AUT Journal of Electrical Engineering
10.22060/EEJ.2025.23497.5618

Kidney Stones CT Images Classification using Graph Convolutional Network

Amin Asgari[1], Hossein Ebrahimnezhad [2*], Mohammad Hossein Sedaaghi [3]

[1]PhD Candidate, Faculty of Electrical Engineering, Sahand University of Technology, Tabriz, Iran
[2]Professor, Computer Vision Research Lab, Faculty of Electrical Engineering, Sahand University of Technology, Tabriz, Iran
[3]Professor, Faculty of Electrical Engineering, Sahand University of Technology, Tabriz, Iran

Abstract:

Kidney stones are solid crystals made of minerals and salts that form within the kidney, often creating a sharp, hard mass. These stones can block urine flow as they move into the urinary tract, making early detection crucial. Although deep neural networks (DNNs) have been used to diagnose kidney stones with some success, they still face performance and standardization issues. A new approach combines graph convolutional networks (GCNs) with DNNs to address these challenges. This method extracts orb features from images, converts them into graphs, and embeds nodes using a graph convolutional network, which includes a message-passing layer and node feature aggregation. The GCN updates node properties, enhancing efficiency and performance when integrated into a deep network. This approach enables more comprehensive and precise feature extraction from images, improving kidney stone diagnosis. The study highlights GCNs' potential in analyzing medical images for diagnosing kidney stones. The proposed architecture was tested using publicly available CT scan images and demonstrated outstanding accuracy, correctly identifying kidney stones or healthy conditions in 98.6% of cases. It outperformed other advanced techniques, especially in detecting stones of various sizes, including very small ones, proving its effectiveness in medical image analysis.

Keywords:

Graph Convolution Network, Kidney Stones, ORB Feature Extraction, Node Embedding, Deep Neural Network

[*] Corresponding author, Email: ebrahimnezhad@sut.ac.ir

10.22060/EEJ.2025.23497.5618

1. Introduction

Kidney stones are hard mineral and salt deposits that form within the kidneys. They can cause severe pain and potentially block urine flow, leading to kidney damage if left untreated. Traditional methods for diagnosing kidney stones, such as ultrasound, CT scans, and X-rays, rely heavily on medical imaging techniques, but interpreting these images can be complex and time-consuming. Often, the process requires the expertise of specialist radiologists [1]. However, the advent of deep learning has significantly transformed medical image analysis. Convolutional neural networks (CNNs), a type of deep learning model, have proven highly effective in image classification and object recognition, enabling accurate disease diagnosis, including kidney stone detection, by learning intricate patterns from large datasets [2].

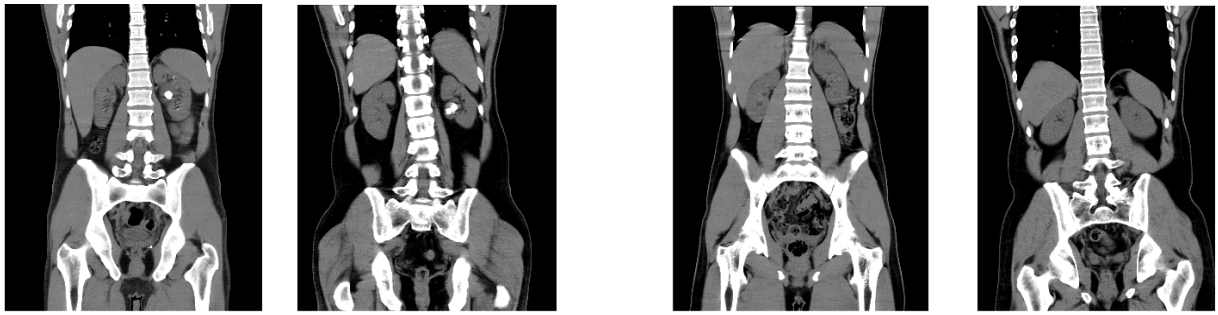
A newer class of neural networks, known as graph convolutional networks (GCNs), operates on graph-structured data and excels in modeling spatial relationships within medical images. This ability to represent complex structures and relationships within the data can enhance diagnostic accuracy, offering a promising alternative or complement to traditional CNNs in medical imaging tasks. By capturing the connections between different kidney segments or stone features, GCNs provide valuable insights into kidney stone formation and distribution. The integration of deep learning models like CNNs and GCNs into kidney stone diagnosis offers the potential to improve both accuracy and speed while reducing radiologists' workload, ultimately expediting patient treatment [3].

As deep learning technologies continue to evolve, GCNs, in particular, hold great promise for advancing the diagnosis of renal pathologies. With their ability to learn from large and diverse datasets, these models are well-positioned to generalize to new data, making them invaluable tools in the future of medical image analysis. This progress is expected to result in improved diagnostic accuracy and better healthcare outcomes for patients [4-6].

The integration of computer vision and deep learning into medical imaging has significantly enhanced the capabilities of diagnostic tools. Deep learning models have shown great success in various tasks, such as image segmentation, classification, and lesion detection, when applied to medical imaging techniques like MRI, CT scans[7], and X-rays[8]. Fig. 1 shows examples of renal CT scans, including normal images and

10.22060/EEJ.2025.23497.5618

images with kidney stones. These advancements have improved diagnostic accuracy and reliability, aiding physicians in diagnosing and managing various medical conditions. Recent progress in artificial intelligence, particularly through deep neural networks (DNNs), has yielded significant achievements in interpreting medical images and biological signals[9]. These sophisticated algorithms have proven valuable across many medical applications, including the urology field, where deep learning is increasingly used for the automated identification of urinary tract stones[10].



(a) (b)

Fig. 1. Sample CT images: (a) kidney stone, and (b) healthy.

This study combines Graph Convolutional Networks (GCNs) for node feature embedding and Deep Neural Networks (DNNs) for final learning. GCNs update nodes by incorporating information from neighboring nodes, integrating both local and neighboring data. This approach applies to various graph-structured data, such as social networks and protein interactions. The refined node features from GCNs are then processed by the DNN to identify complex patterns, making the two-step method more effective at capturing node characteristics.

Several studies have explored different deep learning approaches to improve the classification and detection of urinary tract stones. Nithya et al.[11], in 2020 achieved a 93.45% accuracy rate in classifying stone cases by using GLCM feature extraction models and analyzing 100 samples, despite the dataset being considered small. Wu et al[12]. in 2020 utilized a multi-feature fusion neural network, combined with Inception-V3, to achieve a 94.67% accuracy rate when analyzing ultrasound data. However, the complexity of the model presented challenges for practical application. Other studies, such as those by in 2018 Thein et al[13]. and Cui et al. also focused on kidney stone segmentation and classification,

achieving varying degrees of accuracy. Each study highlighted limitations related to dataset size, model complexity, and generalizability. For instance, Yildirim et al[14]. in 2021 used the XResNet-50 method with a SoftMax classifier to achieve a 96.82% accuracy rate but noted the model's complexity as a potential barrier to practical use. In 2023, Chaohua Yan et al[15]. introduced an optimized Deep Belief Network (DBN) using a fractional coronavirus herd immunity optimizer (FO-CHIO). This method combines deep learning and meta-heuristics to create a customized DBN tailored for kidney stone detection. The approach is based on a fractional version of the coronavirus herd immunity enhancer, aiming to deliver an efficient and reliable detection system. Simulations demonstrate that the proposed DBN/FO-CHIO approach outperforms other studied methods, achieving an accuracy of 97.98%.

The aim of the current research is to leverage Graph Convolutional Networks (GCNs) and deep neural networks to reduce missed kidney stone diagnoses in CT scans and minimize human error, particularly in emergency settings where specialist radiologists may not be available. The increasing availability of data has facilitated the integration of deep learning in medical applications, and graph-based techniques have shown promise in optimizing data usage for image analysis. This study proposes a novel model that combines GCNs with DNNs for the automated classification of kidney stones using CT imaging. The innovative model seeks to enhance the classification accuracy of kidney stones by utilizing features from coronary CT scans, offering a powerful tool for healthcare providers to make precise diagnoses and improve patient outcomes. Experiments were conducted to assess the impact of combining graph convolutional networks (GCN) with deep learning on improving recognition accuracy, specifically by integrating GCN with convolutional neural networks (CNN). This approach leverages the capabilities of GCNs in modeling relationships between elements and the strengths of CNNs in extracting visual features and, resulting in more comprehensive feature extraction and a deeper understanding of the data. The combined method outperforms the use of either technique individually and proves effective in addressing various problems and processing structured data. The goal of this model is to assist radiologists in the precise identification of kidney stones. Key contributions of this study include:

10.22060/EEJ.2025.23497.5618

- ✓ **Dataset Preparation:** A dataset of CT scan images focusing on coronary arteries is compiled. From these images, a region of interest (ROI), specifically targeting kidney areas, is isolated to improve the accuracy of detecting kidney stones.
- ✓ **Feature Extraction:** A feature extraction algorithm processes each image, creating a feature matrix that encapsulates critical image attributes.
- ✓ **Image-to-Graph Conversion:** The feature vectors are transformed into graph nodes, converting images into graph structures based on extracted features.
- ✓ **Graph Convolutional Network (GCN) Processing:** The GCN analyzes the graph nodes to generate embeddings, ensuring accurate feature recognition, including the identification of small kidney stones.
- ✓ **Deep Neural Network Classification:** The deep neural network classifies the embedded features, facilitating the recognition and classification process.
- ✓ **Effectiveness Evaluation:** An in-depth analysis evaluates the performance of the GCN combined with deep learning approaches, verifying its efficacy in enhancing diagnostic precision.

The remainder of the paper is structured as follows: Section 2 outlines the problem statement and details the proposed methodology. Section 3 provides the results along with their analysis. Lastly, Section 4 offers the conclusion.

2. Proposed method

This research explores a method for classifying kidney stones in CT images using a combination of a graph convolutional network (GCN) and a deep neural network (DNN). The process begins with the preprocessing and cropping of CT images to focus on significant areas. The ORB (Oriented FAST and Rotated BRIEF) feature extractor is then used to identify and extract distinct points in each image, which are treated as nodes in a graph, resulting in a separate graph for each image. A GCN is employed to embed these nodes, refining each node's representation based on its neighbors, to capture spatial relationships and dependencies within the image. The refined graphs, enriched with contextual

10.22060/EEJ.2025.23497.5618

information from the GCN, are then input into a DNN to complete the training process. This integrated approach of GCN and DNN allows the model to effectively learn and classify the presence of kidney stones in CT images by combining graph-based and neural network techniques.

2-1- Pre-processing

This research aims to develop a framework to reduce the oversight of kidney stone cases by physicians during CT scan evaluations. The proposed model utilizes Graph Convolutional Networks (GCNs) to detect kidney stones in low-contrast coronary CT scans, combining advanced deep learning with radiographic image processing. As shown in Fig. 2, The methodology involves five key phases: data acquisition, data preparation, image-to-graph conversion, application of the GCN, and image classification.

10.22060/EEJ.2025.23497.5618

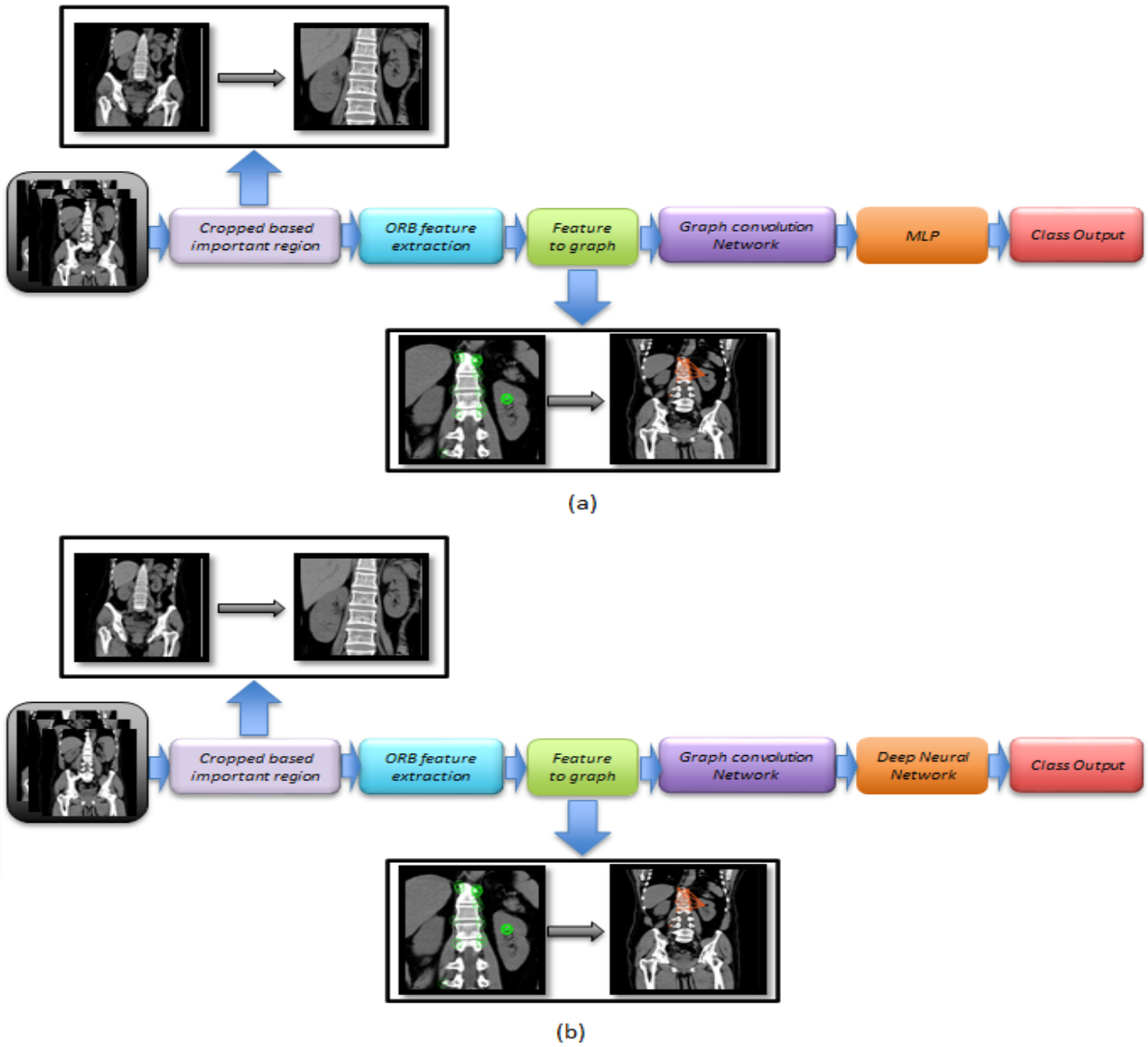
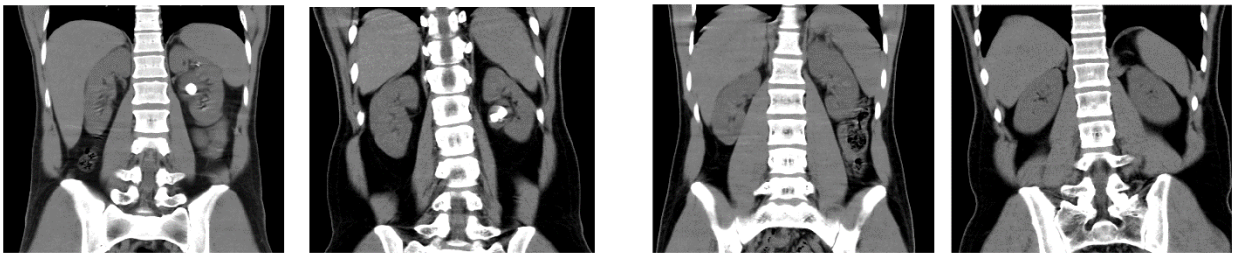


Fig. 2. Overall steps of the proposed method for kidney stone CT image classification.
a) standard model of GCN. b) Enhanced model of GCN.

The dataset is compiled by sourcing images and standardizing their dimensions to an average size. To increase data variability and improve machine learning model performance, the images are synthetically modified through rotations (5 and 10 degrees) and translations (horizontal and vertical shifts). These augmentation techniques help the model recognize features from different angles and reduce orientation bias. The images are then preprocessed through resizing, rotation, translation, and trimming. Trimming removes border areas to focus on the kidney region, enhancing the precision and depth of analysis. This targeted approach reduces noise, improves algorithm effectiveness, and increases the accuracy and

10.22060/EEJ.2025.23497.5618

reliability of image analysis, particularly crucial in medical imaging. Fig. 3 depicts the pre-processing stage, highlighting the kidney region.



(a) (b)

Fig. 3. Image preprocessing reduces noise by using image cropping, which removes excess information by focusing on the kidney region. (a) kidney stone images cropped, (b) normal images cropped.

2-2- ORB Feature Extraction

In medical image processing, algorithms like ORB (Oriented FAST and Rotated BRIEF) [16] improve diagnostic efficiency by focusing on key image points, such as edges, where brightness and contrast change significantly. This targeted approach reduces unnecessary pixel processing, enhancing detection accuracy for critical areas like kidney stones and speeding up analysis.

The study did not use deep feature extraction due to challenges in interpreting complex image-derived concepts. Although deep learning can identify intricate patterns, the abstract nature of deep features makes it difficult to correlate them with specific image elements, limiting their practical application in analyzing CT images of kidney stones. For the present study, the use of ORB feature extractor is as follow, after locating the kidney region in the images, we employ the ORB feature extractor, a robust image processing and machine vision technique, to extract features. ORB identifies and describes key points in images, which are distinct points characterized by significant changes in color intensity or texture. These key points, along with their descriptors, are crucial for image analysis and matching.

In Eq. (1), The FAST algorithm, which detects points with substantial color intensity differences from their surroundings, is used to identify these key points. If the number of points within a circular area around a potential key point P with intensity changes greater than a threshold t compared to P's intensity exceeds a certain number, P is recognized as a key point.

10.22060/EEJ.2025.23497.5618

$$t \text{ for at least neighbors } < |Intensity(n) - True \text{ if } | Intensity(p) = FAST(p) \quad (1)$$

In this equation, $Intensity(p)$ represents the color intensity at point p, and n denotes one of its neighboring points. The variable t is the threshold for color intensity change, and k indicates the number of neighboring points. Descriptors are characteristic attributes that offer supplementary details regarding salient points within an image. The ORB (Oriented FAST and Rotated BRIEF) algorithm enhances the BRIEF (Binary Robust Independent Elementary Features) descriptors to ensure they are invariant to rotational transformations. BRIEF encodes the characteristics of these salient points in a binary format. These descriptors are derived by evaluating the color intensity differences at various sampled locations surrounding the salient points. According to Eq. (2), The outcome of these evaluations is encoded as a binary vector consisting of 0s and 1s. For each salient point, the color intensity is compared between pairs of sampled points in its vicinity. If the color intensity at the first sampled point ${}_1p$ exceeds that of the second sampled point ${}_2p$, a value of 1 is assigned to the corresponding position in the binary vector; conversely, if the intensity at ${}_1p$ is less than or equal to that at ${}_2p$, a value of 0 is assigned.

$$\left.\begin{matrix} if\ I(p_i) & 1 \\ otherwise & 0 \end{matrix}\right\} = {}_i Binary\ Feature\ Vector\ where\ each\ bit\ b = BRIEF(p) \quad (2)$$

In this equation, $I(p_i)$ is the color intensity of point ${}_i p$ and ${}_j p$ of another point in the sampled area.

BRIEF descriptors are not particularly robust against image rotation; therefore, ORB incorporates an additional step to enhance feature rotation invariance. This step involves computing an orientation for each key point to ensure that the descriptors remain consistent despite image rotations. An angle is determined for each key point, which defines the orientation of the descriptor. This angle is then applied to the BRIEF descriptors to make them resistant to rotational transformations.

The process of feature extraction using ORB is as follows: initially, the FAST algorithm is employed to detect key points within the image. Subsequently, the orientation angle for each key point is computed. Following this, BRIEF is utilized to generate a binary descriptor for each key point. These descriptors are constructed based on the intensity values at sampled locations around each key point. Finally, the

10.22060/EEJ.2025.23497.5618

descriptors are rotated in accordance with the computed orientation angle to achieve rotational invariance. The optimal threshold for identifying key points with the FAST algorithm was determined through various experiments, in such a way that it provided the highest accuracy in identifying features and small kidney stones, and the overall performance of the algorithm remained resistant to threshold changes

2-3- Image to graph With Features

In this study, ORB features are extracted from the image using the FAST algorithm to detect key points with sharp intensity changes, followed by the BRIEF algorithm to describe these points. ORB is effective under varying lighting conditions and image rotation. Each key point is treated as a node in a graph, and connections are formed based on the distances between nodes within a certain area, modeling spatial relationships and enhancing the understanding of the image's structure. After extracting features from images using the ORB algorithm, a 200×32 feature matrix is obtained for each image. Each 1×32 feature vector is treated as a node, and a graph with 200 nodes is constructed for each image by connecting nodes based on a similarity threshold set at 200. This process is applied to all images, resulting in a unique graph for each, as illustrated in Fig. 4. The graph-based representation preserves local and global image information, enables flexible comparison, and captures complex spatial relationships, facilitating the application of graph algorithms and analysis techniques to visual data.

(a) (b)

Fig. 4. (a) ORB Feature Extraction, (b) Feature to graph

Empirical testing established a similarity threshold of 200 as the most effective option after evaluating various values. This threshold was selected based on tests across multiple datasets, optimizing graph density and connectivity, which enhanced the Graph Convolutional Network (GCN) model's learning and

10.22060/EEJ.2025.23497.5618

prediction accuracy. While adaptive thresholds were considered, they were excluded due to higher computational complexity and instability. Thus, a threshold of 200 was confirmed as optimal for improving graph structure and GCN performance.

2-4- Graph convolution network for node embedding

Graph Convolutional Networks (GCNs) are neural networks tailored for graph-structured data, excelling in tasks like node classification, link prediction, and graph classification. They work by embedding nodes to capture their properties and inter-node relationships. GCNs extend the convolution operation from grid-like structures to graphs through message passing, where nodes share attribute information with neighbors, followed by aggregation and update steps to refine node representations. As shown in Eq. (3), The effectiveness of a GCN layer can be described mathematically, typically through a formulation that captures how node representations are transformed through these steps. This mathematical expression encapsulates the process of information propagation and aggregation across the graph, leading to the learning of meaningful node embeddings.

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \tag{3}$$

where $\mathbf{H}^{(l)}$ is the matrix of node features at layer l, with dimensions $N \times F_l$, where N is the number of nodes, and F_l is the number of features at layer l. $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ Is the adjacency matrix of the graph with added self-loops, where A is the original adjacency matrix and I is the identity matrix. $\tilde{\mathbf{D}}$ is the diagonal degree matrix of$\tilde{A}$, with entries $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$. $\mathbf{W}^{(l)}$ the learnable weight matrix at layer l, with dimensions $F_l \times F_{l+1}$. σ is an activation function, such as ReLU or a sigmoid function. The formula is derived from spectral graph theory, where the convolution operation on a graph within the spectral domain is defined using the graph Laplacian. The normalization term $\tilde{\mathbf{D}}^{(-\frac{1}{2})}(\tilde{\mathbf{A}})\tilde{\mathbf{D}}^{(-\frac{1}{2})}$, ensures that the node features are normalized, preventing the unbounded growth of features across layers. The stages of the graph convolution network (GCN) layer in the study are:

10.22060/EEJ.2025.23497.5618

a) Message Passing: In the message passing phase, which is specified by Eq. (4), each node i sends its current feature vector $\mathbf{h}_i^{(l)}$ to its immediate neighbors. This process can be conceptualized as a node disseminating its information throughout the graph. By including self-loops in the adjacency matrix $\tilde{A}$, each node also takes into account its own features.

$$\mathbf{M}_i^{(l)} = \sum_{j \in \aleph(i) U \{i\}} \mathbf{h}_j^{(l)} \tag{4}$$

where $\aleph(i)$ denotes the set of neighbors of node i.

b) Aggregation Phase: During the aggregation phase, according to Eq. (5), each node collects and combines the messages it has received from its neighboring nodes. This process typically involves summing or averaging the messages. To ensure that nodes with varying numbers of connections (degrees) contribute fairly to the overall sum, a normalization by $\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is applied. This normalization step is crucial for maintaining the integrity of the aggregated information, regardless of the node's degree of connectivity within the network.

$$\mathbf{A}_i^{(l)} = \tilde{\mathbf{D}}_{ii}^{-\frac{1}{2}} \sum_{j \in \aleph(i) U \{i\}} \tilde{\mathbf{A}}_{ij} \mathbf{h}_j^{(l)} \tilde{\mathbf{D}}_{jj}^{-\frac{1}{2}} \tag{5}$$

This step captures the local structure of the graph by combining information from neighboring nodes.

c) Update Phase: In the update phase, according to Eq. (6), the aggregated information is transformed using a learnable weight matrix $\mathbf{W}^{(l)}$. This transformation is followed by the application of a nonlinear activation function σ, which generates updated node features $\mathbf{h}_i^{(l+1)}$. This process allows the model to learn complex patterns and relationships within the graph structure.

$$\mathbf{h}_i^{(l+1)} = \sigma(\mathbf{A}_i^{(l)} \mathbf{W}^{(l)}) \tag{6}$$

This transformation allows the network to learn complex patterns in the data by applying nonlinearities and tuning the parameters of $\mathbf{W}^{(l)}$. Graph Convolutional Networks (GCNs)

10.22060/EEJ.2025.23497.5618

typically consist of several layers, with the output of each layer serving as the input for the subsequent layer. This hierarchical structure enables the network to learn representations of the graph that capture both local and global structural information.

In the study, Graph Convolutional Networks (GCNs) are utilized to enhance image analysis by transforming image features into graph nodes. Initially, image features are extracted using the ORB feature extractor, producing high-dimensional vectors that capture local patterns. These features are then used to construct a graph where each feature point is a node, and edges are established based on spatial proximity or feature similarity, forming an adjacency matrix. The GCNs process this graph through multiple layers, refining the node features by integrating information from neighboring nodes. This process enhances the representation of each node by considering the broader image context.

Through the iterative process of message passing, aggregation, and non-linear activation in GCNs, the embeddings of the nodes are continually refined, incorporating both local and global contextual information. This refined representation significantly improves the performance of image-related tasks such as segmentation, classification, and object recognition. GCNs thus offer a robust framework for capturing intricate relationships within image data, making them highly effective for complex image analysis tasks. Fig. 5 shows the graph convolution network used in embedding nodes. This procedure is conducted over multiple iterations to enhance the positioning of nodes within the graph, taking into account the broader context of their neighborhood. Subsequent to the update of the graph nodes, a 200×32 matrix is generated for each image, which serves as a representation of the image features extracted through the graph convolution network.

10.22060/EEJ.2025.23497.5618

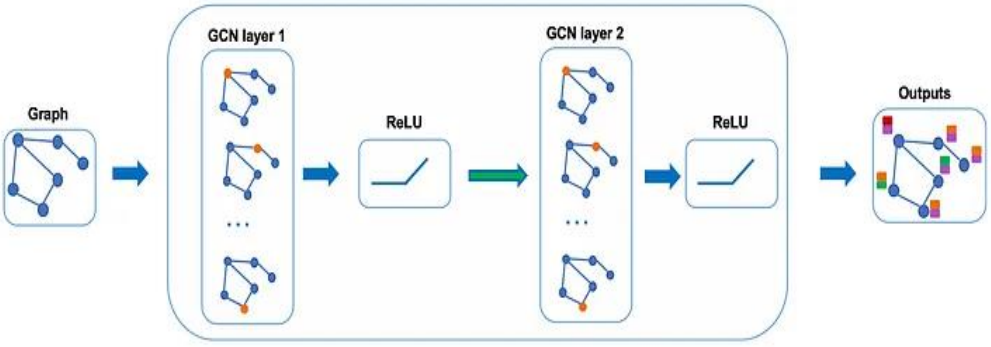


Fig. 5. features of each node are updated in every message-passing layers.

In graph convolutional networks (GCNs), selecting the right adjacency matrix, including inner loops, is crucial for effective feature propagation and model performance. This choice allows nodes to update their attributes using both their own and neighbors' characteristics, enhancing pattern recognition accuracy. Normalization helps prevent feature accumulation across layers, but in sparse graphs, it can lead to information loss, while in dense graphs, it may overemphasize central nodes. Thus, choosing suitable normalization methods and adjacency matrices based on the graph's structure is essential for optimizing GCN performance.

In the standard Graph Convolutional Network (GCN) workflow, after the node embedding stage, the learned embeddings are typically passed to a downstream model for task-specific learning. Initially, a Multilayer Perceptron (MLP) with three fully connected layers of 256, 128, and 64 neurons is used. The MLP processes these embeddings to perform classification or regression tasks, with training done using backpropagation and an appropriate loss function. However, the results from this approach often fall short of expectations. This underperformance is likely due to the MLP’s inability to leverage the rich structural and topological information embedded in the node features, as it treats features independently and lacks the capacity to capture complex patterns inherent in the graph data. To improve performance, the MLP is replaced with a convolutional network, which is better equipped to process the spatial and hierarchical relationships in the embeddings. Convolutional layers, by design, are adept at identifying local patterns and maintaining feature correlations, making them more effective in extracting meaningful

representations from the node embeddings. This approach leads to significantly better results, as the convolutional network can exploit the structural properties of the embeddings, enhancing task-specific learning. This shift highlights the importance of selecting the right downstream architecture to complement the embeddings produced by GCNs, especially for tasks where the graph's inherent structure plays a crucial role in determining the outcome.

2-5- Deep Neural Network

In the subsequent phase, the extracted features from each image are fed into a Deep Neural Network for the feature training process. This network consists of several key components designed for efficient learning and classification, including convolutional layers with Leaky ReLU activation, normalization layers, max-pooling layers, fully connected layers, and dropout layers. The initial six convolutional layers are crucial for extracting and refining high-level features from the images, with batch normalization applied after each layer to stabilize and accelerate training. MaxPooling layers then reduce the spatial dimensions of the feature maps, helping to decrease computational load and control overfitting.

To further enhance generalization and prevent overfitting, dropout layers are used, randomly omitting a portion of input units during training. The fully connected layers follow, with one dense layer using Leaky ReLU for high-level reasoning, and another final fully connected layer responsible for generating class scores. These scores are passed through a SoftMax layer, which normalizes them into probabilities for classification. The classification layer then interprets these probabilities to predict the final class, assigning labels to the input data and playing a critical role in loss calculation during training. Fig. 6 illustrates the architecture of the deep network used for training, incorporating features extracted from the graph convolutional network. As shown in Fig. 6, a convolutional neural network (CNN) is defined for classification. It begins with an input layer designed to process the feature matrix obtained from the graph convolutional network. The network comprises four convolutional blocks, each with convolutional layer that use 3*3 filters, followed by batch normalization, a Leaky ReLU activation (0.01), and a max-pooling layer that halves the spatial dimensions. These blocks progressively increase the number of filters from 64

10.22060/EEJ.2025.23497.5618

to 512, allowing the network to extract more complex features at each stage. Dropout layers, with a dropout rate of 20 percent, are included after each block to mitigate overfitting. After feature extraction, the network transitions to fully connected layers with sizes gradually reducing from 256 to 64 neurons, each followed by Leaky ReLU activation and dropout layers. The final fully connected layer maps the features to the number of output classes. A SoftMax layer converts the outputs into probabilities, and a classification layer computes the loss for classification. This architecture effectively combines feature extraction, nonlinearity, dimensionality reduction, and regularization to build a robust model for image classification tasks.

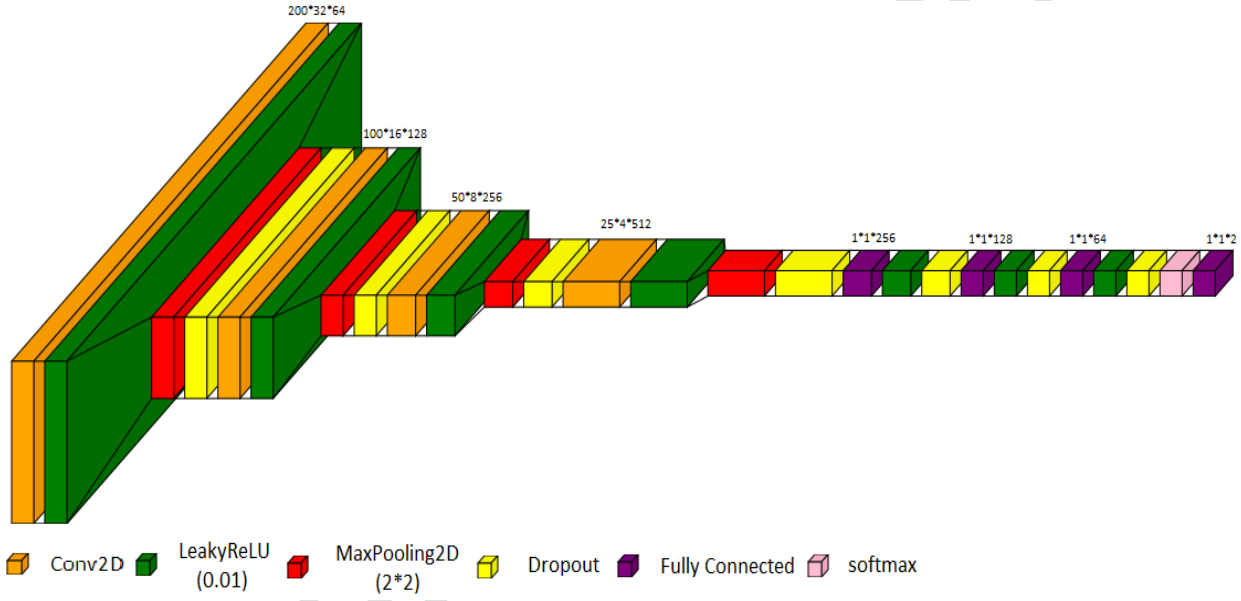


Fig. 6. architecture of the proposed CNN network

The deep learning model's training process involves feeding image features through convolutional, normalization, pooling, and fully connected layers, with dropout layers to prevent overfitting. The network's weights are adjusted using the Adam optimizer to minimize the loss, typically cross-entropy, which measures the difference between predicted probabilities and actual labels. This architecture effectively processes and refines image features, enabling the network to learn complex patterns and achieve accurate image classification, with the final SoftMax layer generating class probabilities for classification.

10.22060/EEJ.2025.23497.5618

The combination of Graph Convolutional Networks (GCNs) and Deep Neural Networks (DNNs) significantly enhances the model's ability to process complex data, especially in contexts where both relational graph data and image-based patterns need to be analyzed together. GCNs are designed to efficiently extract and propagate information from graph-structured data by learning from the connectivity between nodes (i.e., entities in the graph) and their neighbors. This allows GCNs to capture both the inherent structure of the graph and the interactions between entities, providing a comprehensive feature matrix that encodes essential structural relationships and node-specific information. When these graph-derived features are passed as inputs into a Convolutional Neural Network (CNN), the fusion of GCN and CNN capabilities is powerful. The CNN's hierarchical layers excel in recognizing spatial patterns and capturing complex, multi-level features in images or other structured data. By feeding the feature-rich output of a GCN into the CNN, the model benefits from the GCN's ability to preserve the graph's topological information, which is crucial for tasks like node classification, graph classification, or any task involving data with inherent structure.

Moreover, the synergy between GCN and CNN plays a pivotal role in improving the model's capacity to understand both the structural dependencies and the local patterns in the data. The GCN provides a solid foundation by encoding the relationships between entities in a graph, while the CNN further enhances this information through its powerful feature extraction and abstraction capabilities. The end result is a model that can simultaneously capture relational patterns (from the graph) and local, spatial patterns (from the CNN), which is essential for tasks requiring a deeper understanding of both graph structures and complex, spatial relationships.

As specified in Algorithm 1, a convolutional neural network (CNN) model is designed to classify images into two classes. First, the input data, including features obtained from the convolutional graph network and the training and test labels, is loaded and preprocessed. This process involves randomizing the data, normalizing the pixel values to the interval [0, 1], and converting the labels into a categorized format. Next, the training data is split into two parts: training and validation. The CNN model is constructed with

convolutional layers for feature extraction, normalization layers for stability, activation layers for nonlinear learning, and aggregation and dropout layers to reduce complexity and prevent overfitting. The training configuration includes the Adam optimizer, a fine-tuned learning rate, and continuous evaluation on the validation data. Finally, the trained model is capable of classifying new data with high accuracy.

Algorithm1. Preprocessing, Training, and Validation of a CNN Model for Classification

Step	Description
Input	Training and test Features and labels.
Output	Trained CNN model.
1. Initialization	Clear workspace and load data files.
2. Preprocessing	Shuffle data,input Features [200 × 32], normalize to [0, 1], and convert labels to categories.
3. Data Split	Split training data into 80% training and 20% validation sets.
4. CNN Definition	Define CNN with input, convolutional, pooling, dropout, and fully connected layers.
5. Training Setup	Config. optimizer (Adam), learning rate, epochs (30), and validation monitoring.
6. Training	Train the CNN using train Network with prepared data and options.

3- Results & Discussions

We developed the proposed model using the MATLAB 2023b programming environment. The simulation ran on a system with an Intel i7 processor, 48 GB of DDR4 RAM, and an NVIDIA RTX 3080 GPU. The dataset contains 1799 CT scans of kidneys: 790 scans with kidney stones and 1009 scans of healthy kidneys. We divided the data into two subsets for testing: 90% for training and 10% for validation. Additionally, the experimental dataset included 346 CT scans, with 165 images showing kidney stones and 181 depicting healthy kidneys.

3-1- Dataset

The dataset was prepared by Yildirim et al [14] following approval from the Ethics Committee of Firat University, Turkey. It includes 500 non-contrast computed tomography (NCCT) scans from 433 patients aged 18 to 80, divided into two groups, 278 with kidney stones and 165 healthy individuals. The CT scans were independently reviewed and labeled by a radiologist and a urologist to confirm the presence or

10.22060/EEJ.2025.23497.5618

absence of stones. The scans were taken using a 120 kV CT protocol with an automatic current range of 100-200 mA and a slice thickness of 5 mm. The dataset comprises a total of 1,799 images, with 790 depicting kidney stones and 1,009 showing healthy kidney tissue. To maintain a balanced dataset, data augmentation techniques were used to adjust the ratio of stone-containing to normal images. The data was then split into training and testing sets, with the training directory containing 1,453 images (625 with stones and 828 without stones) and the testing directory containing 346 images (165 with stones and 181 without stones). To minimize bias, participants in the training set were not included in the testing set, ensuring distinct groups for each phase. The images were saved in PNG format and used for evaluating the performance of the graph convolutional network (GCN) algorithm combined with a deep neural network (DNN).

3-2- Setting hyperparameters

During the training and validation phases of a deep learning model, hyperparameters are fine-tuned to enhance performance. Initially, the model is trained on a training dataset, and its performance is evaluated on a validation dataset. Hyperparameters such as learning rate (η), batch size, and number of epochs are adjusted iteratively until optimal values are found. Finally, the model's performance is tested on a separate test set to ensure it does not overfit the validation data. The proposed architecture achieved 98.6% accuracy after 50 iterations, demonstrating improved performance over existing models for kidney stone classification.

The network is trained using the Adam optimizer, with a batch size of 64 and an initial learning rate of 0.0001. The learning rate decreases by a factor of 0.1 every 30 epochs. Validation is conducted on a separate dataset every 100 iterations, and the process includes logging details and visualizing progress to monitor performance. The runtime environment is configured to automatically choose between CPU and GPU for optimal efficiency. These settings facilitate efficient and effective neural network training, promoting robust learning and generalization. Fig. 7 presents the training and validation accuracy for each epoch of the proposed GCN-DNN model, providing insight into its performance.

10.22060/EEJ.2025.23497.5618

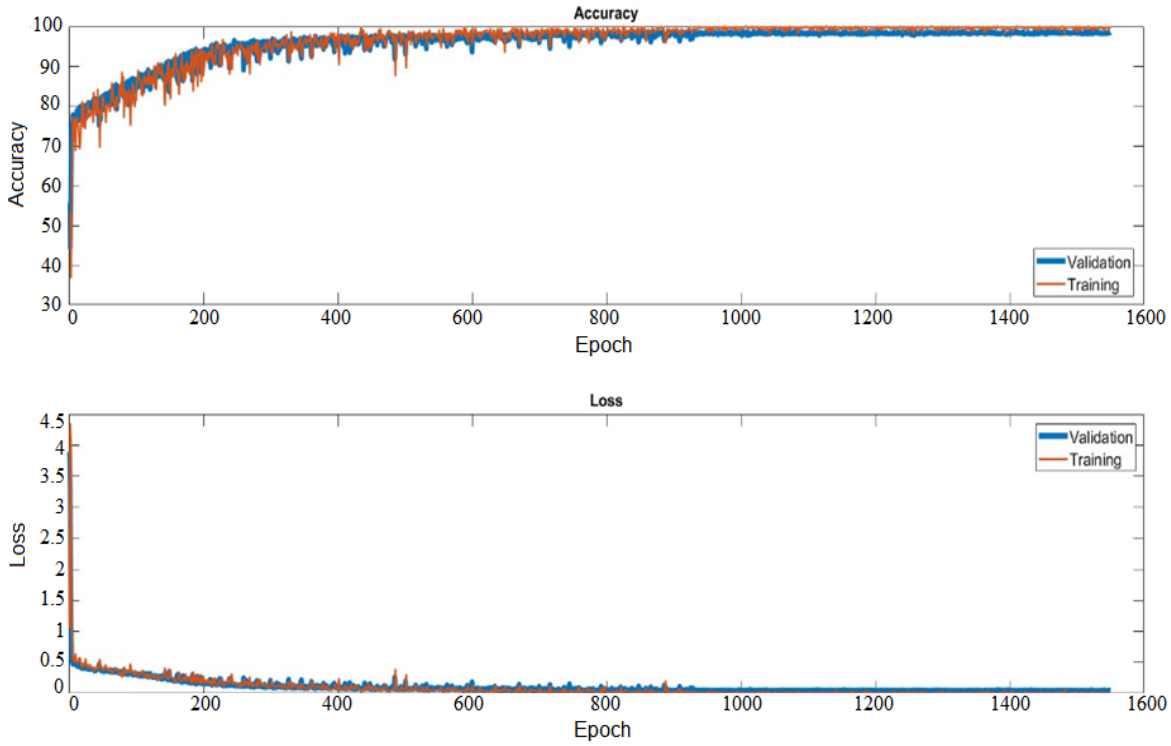


Fig. 7. Training, validation accuracy and loss of the proposed model.

The confusion matrix in Fig. 8 illustrates the performance of the proposed GCN-DNN model for kidney stone detection, achieving an impressive accuracy of 98.6% using a Graph Convolutional Network (GCN). This highlights the model's exceptional capability to deliver highly accurate results.

10.22060/EEJ.2025.23497.5618

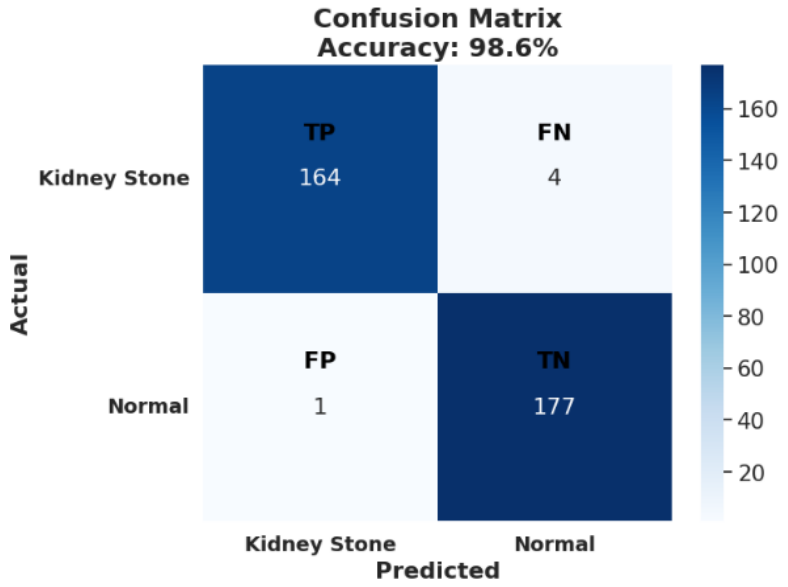
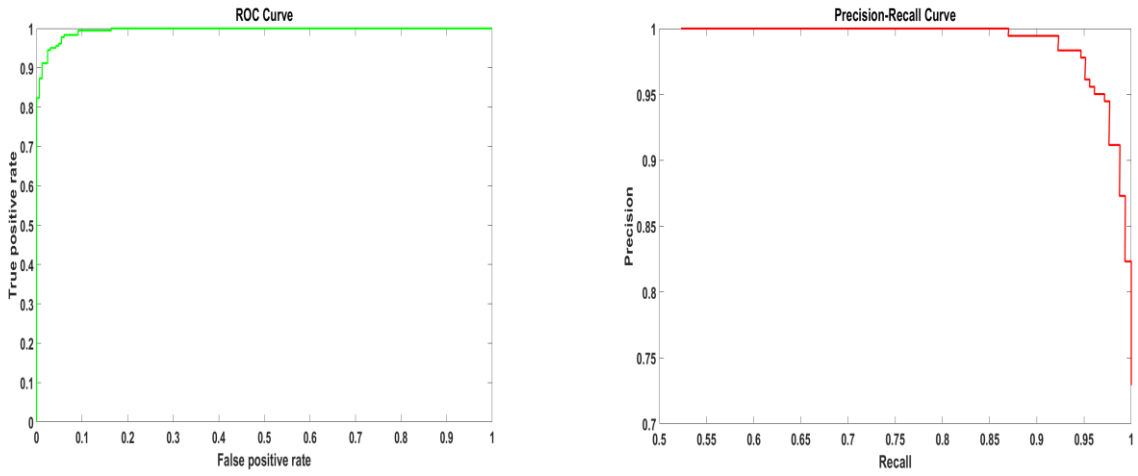


Fig. 8. confusion matrix derived from the test data

The confusion matrix[17] reveals that the model accurately predicts kidney stone images as true positives (TP) and correctly identifies normal images as true negatives (TN). However, it occasionally misclassifies kidney stone images as normal (false negatives, FN) and normal images as kidney stones (false positives, FP). To evaluate the model's performance, we use several metrics: precision (TP / (TP + FP)), recall (TP / (TP + FN)), and the F1 score (2 × precision × recall / (precision + recall)). Note that the formula for accuracy in the original text is incorrect; it should be (TP + TN) / (TP + TN + FP + FN). These metrics are crucial for assessing the model's effectiveness. Accuracy measures the overall rate at which the model correctly predicts both normal and kidney stone cases, while recall evaluates the model's ability to identify all true positive cases. The F1 score, being the harmonic mean of precision and recall, balances these two measures.

In our evaluation with 346 test cases, the model achieved 98.6% accuracy and 99% sensitivity. We tested the model on an unbalanced dataset comprising 790 kidney stone scans and 1009 normal scans. Fig. 9 presents both the receiver operating characteristic (ROC) curve[18] and the precision-recall (PR) curve[18]. The PR curve is generally more informative for datasets with an unbalanced distribution, as it better reflects the model's performance on the minority class. Conversely, ROC curves are more suitable for balanced datasets.

10.22060/EEJ.2025.23497.5618



(a) (b)

Fig. 9. (a) ROC curve and (b) Precision-Recall (PR) curve for classification performance.

3-3- Comparison with Standard Methods

In this section, we evaluate the performance of our proposed GCN-DNN model against traditional deep learning models, including xResNet-50 [14], DELM [20], Urinary [19], and DKN [21], as well as the standard graph convolutional network (GCN) model with MLP. Table 1 presents a comparative analysis based on metrics such as accuracy, sensitivity, specificity, precision, F1 score, and testing time. The standard GCN model, designed for graph data classification, achieved an accuracy of 80%. While this accuracy may suffice for some applications, it is considered relatively low for this particular problem, which involves a complex and challenging dataset. This suboptimal performance highlights the limitations of the standard GCN model in extracting deeper features or establishing more intricate connections between data points. To enhance accuracy and overall performance, we employed a GCN-DNN hybrid model. This approach combines the strengths of GCNs for extracting graph-structured features and deep neural networks (DNNs) for learning more complex nonlinear patterns. By replacing the MLP with the GCN-DNN architecture, the model achieves better generalization and accuracy, providing a more robust evaluation on complex datasets. This combination leverages the advantages of both GCN and DNN architectures, addressing the limitations of the standard model.

10.22060/EEJ.2025.23497.5618

As fully specified in Table 1, The GCN-DNN model achieves the highest accuracy of 98.6%, surpassing all other models. It converges more rapidly, requiring only 50 runs, and demonstrates the highest sensitivity at 99% and an F1 score of 98.7%. Although its specificity of 97.5% is slightly lower than DKN's 98.9%, the GCN-DNN model's accuracy is comparable to DKN's 99%. Notably, the GCN-DNN model also boasts the shortest testing time of 0.93 seconds, showcasing its computational efficiency. Overall, the GCN-DNN model outperforms benchmark models in terms of accuracy, F1 score, and computational efficiency, making it a promising choice for real-time applications in biomedical engineering.

Table 1. Comparison of results (%) obtained by our proposed GCN_DNN model with deep learning-based methods.

Model	Accuracy	Epochs	Sensitivity	Specificity	Precision	F1-Score	Test time
Urinary[19]	88%	-	86%	90%	89.6%	86.5%	-
DELM[20]	94.4%	50	94.6%	93.3%	92.5%	93.9%	-
xResNet-50 [14]	96.8%	200	95%	97%	97%	96%	1.3 Sec
DKN[21]	98.5%	150	98.1%	**98.9%**	**99%**	98.6%	1.5 Sec
GCN-Standard (proposed)	80.06%	200	80%	81%	82%	81%	1.06 Sec
GCN-Enhanced (proposed)	**98.6%**	**50**	**99%**	97.5%	97.8%	**98.7%**	**0.93 Sec**

This section discusses the Ablation Study [22] and time complexity [23] analysis, which are essential for understanding and improving the model's performance. The Ablation Study involves modifying or removing components of a model to evaluate their impact on performance, helping to identify critical elements and optimize the model by eliminating unnecessary parts. Various modifications were made, and the model's accuracy was assessed at each stage. Table 2 compares the baseline model's performance with these modifications, which included removing layers (convolutional, dropout, batch normalization),

10.22060/EEJ.2025.23497.5618

reducing neuron units in fully connected layers, and adjusting hyperparameters like dropout rate, epochs, optimizers, and learning rate.

Table 2. Ablation Study for the proposed method

Changes Made	Model Accuracy (%)	Change from Base Model	Description
Removed one convolution layer	95.3	-3.3	Reduced number of filters, accuracy drop.
Removed dropout layers	97.2	-1.4	Slight increase in overfitting.
Removed normalization layers	93.8	-4.8	Negative impact on convergence and accuracy.
Reduced number of neurons	96.1	-2.5	Reduced model capacity, accuracy drop.
Changed dropout rate to 0.5	97.8	-0.8	Slight improvement in preventing overfitting.
Changed dropout rate to 0.1	96.8	-1.8	Slight increase in overfitting.
Used SGD optimizer	96.4	-2.2	Slower convergence and accuracy drop.
Learning rate set to 1e-3	97.5	-1.1	Learning rate 1e-3

The results indicated that removing normalization and convolutional layers significantly reduced accuracy, highlighting their critical role in network convergence. The Adam optimizer with a learning rate of 1e-4 proved to be the most effective. Increasing epochs to 50 slightly improved accuracy but also raised training time. Removing dropout layers caused overfitting, emphasizing their importance in prevention. These insights aid in optimizing network architecture and assessing model strengths and weaknesses.

Next, the time complexity of the learning algorithm is analyzed based on layers, neurons, input size, and other parameters. Training time complexity depends on iterations, dataset size, layer count, and parameters, with deeper networks requiring more calculations. In CNNs, time complexity is influenced by convolution operations, filter sizes, and computation methods for each layer. Table 3 outlines the time

complexity for convolutional, fully connected, and max-pooling layers, demonstrating that increased layer numbers and complexity lead to longer processing times. This table illustrates the impact of each network component on overall performance.

Table 3. Time Complexity of CNN Layers

Layer	Output Size	Time Complexity	Layer	Output Size	Time Complexity
Image Input Layer	200x32x1	O(1)	Conv2D (3, 512)	25x4x512	O(3*3*256*512*25*4)
Conv2D (3, 64)	200x32x64	O(3*3*1*64*200*32)	Conv2D (3, 512)	25x4x512	O(3*3*512*512*25*4)
Conv2D (3, 64)	200x32x64	O(3*3*64*64*200*32)	Max Pooling (2x2)	12x2x512	O(12*2*512)
Max Pooling (2x2)	100x16x64	O(100*16*64)	Dropout	12x2x512	O(1)
Dropout	100x16x64	O(1)	Fully Connected (256)	256	O(12*2*512*256)
Conv2D (3, 128)	100x16x128	O(3*3*64*128*100*16)	Dropout	256	O(1)
Conv2D (3, 128)	100x16x128	O(3*3*128*128*100*16)	Fully Connected (128)	128	O(256*128)
Max Pooling (2x2)	50x8x128	O(50*8*128)	Dropout	128	O(1)
Dropout	50x8x128	O(1)	Fully Connected (64)	64	O(128*64)
Conv2D (3, 256)	50x8x256	O(3*3*128*256*50*8)	Dropout	64	O(1)
Conv2D (3, 256)	50x8x256	O(3*3*256*256*50*8)	Fully Connected (2)	2	O(64*2)
Max Pooling (2x2)	25x4x256	O(25*4*256)	SoftMax	2	O(2)
Dropout	25x4x256	O(1)	Classification Layer	2	O(1)

4- Conclusion

This research introduces a deep learning model based on graph convolutional networks (GCNs) to enhance image feature extraction. By converting image feature vectors into graph nodes and applying GCNs with a message-passing algorithm, this approach captures more detailed and comprehensive features from images. This technique significantly improves the traditional methods of image analysis by effectively handling complex details crucial for accurate interpretation. The model was tested on publicly

available CT scans for kidney stone detection, achieving an impressive accuracy of 98.6%. This performance surpasses existing state-of-the-art methods, especially in detecting stones of varying sizes, including small ones. Such precision is vital in medical diagnostics, where accurate and timely detection can significantly impact patient outcomes. The primary advantage of using GCNs lies in their ability to perform convolution operations on graph nodes, capturing both local and global structural information. This not only enhances feature extraction but also preserves intricate details and variations within images. The capability to identify fine details is particularly beneficial in medical imaging, where such precision can be crucial for diagnosis.

References:

1. Alelign, T. and B. Petros, Kidney stone disease: an update on current concepts. Advances in urology, 2018. **2018**(1): p. 3068365.
2. Kolhe, M., et al., Advances in data and information sciences. Lecture Notes in Networks and Systems, 2017. **39**.
3. Zhang, S., et al., Graph convolutional networks: a comprehensive review. Computational Social Networks, 2019. **6**(1): p. 1-23.
4. Ren, H., et al., Graph convolutional networks in language and vision: A survey. Knowledge-Based Systems, 2022. **251**: p. 109250.
5. Rahman, M.M. and R. Marculescu. G-CASCADE: Efficient cascaded graph convolutional decoding for 2D medical image segmentation. in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2024.
6. Song, T.-A., et al. Graph convolutional neural networks for Alzheimer's disease classification. in 2019 IEEE 16th international symposium on biomedical imaging (ISBI 2019). 2019. IEEE.
7. Brisbane, W., M.R. Bailey, and M.D. Sorensen, An overview of kidney stone imaging techniques. Nature Reviews Urology, 2016. **13**(11): p. 654-662.
8. Hu, S., et al. Towards quantification of kidney stones using X-ray dark-field tomography. in 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017). 2017. IEEE.
9. Lee, J.-G., et al., Deep learning in medical imaging: general overview. Korean journal of radiology, 2017. **18**(4): p. 570-584.
10. Suzuki, K., Overview of deep learning in medical imaging. Radiological physics and technology, 2017. **10**(3): p. 257-273.
11. Nithya, A., et al., Kidney disease detection and segmentation using artificial neural network and multi-kernel k-means clustering for ultrasound images. Measurement, 2020. **149**: p. 106952.
12. Wu, Y. and Z. Yi, Automated detection of kidney abnormalities using multi-feature fusion convolutional neural networks. Knowledge-Based Systems, 2020. **200**: p. 105873.
13. Thein, N., et al. A comparison of three preprocessing techniques for kidney stone segmentation in CT scan images. in 2018 11th Biomedical Engineering International Conference (BMEiCON). 2018. IEEE.
14. Yildirim, K., et al., Deep learning model for automated kidney stone detection using coronal CT images. Computers in biology and medicine, 2021. **135**: p. 104569.

15. Yan, C. and N. Razmjooy, Kidney stone detection using an optimized Deep Believe network by fractional coronavirus herd immunity optimizer. Biomedical Signal Processing and Control, 2023. **86**: p. 104951.
16. Weberruss, J., L. Kleeman, and T. Drummond. ORB feature extraction and matching in hardware. in Australasian conference on robotics and automation. 2015.
17. Patro, V.M. and M.R. Patra, Augmenting weighted average with confusion matrix to enhance classification accuracy. Transactions on Machine Learning and Artificial Intelligence, 2014. **2**(4): p. 77-91.
18. Fan, J., S. Upadhye, and A. Worster, Understanding receiver operating characteristic (ROC) curves. Canadian Journal of Emergency Medicine, 2006. **8**(1): p. 19-20.
19. Parakh, A., et al., Urinary stone detection on CT images using deep convolutional neural networks: evaluation of model performance and generalization. Radiology: Artificial Intelligence, 2019. **1**(4): p. e180066.
20. Rehman, A., et al., HCDP-DELM: Heterogeneous chronic disease prediction with temporal perspective enabled deep extreme learning machine. Knowledge-Based Systems, 2024. **284**: p. 111316.
21. Patro, K.K., et al., Application of Kronecker convolutions in deep learning technique for automated detection of kidney stones with coronal CT images. Information Sciences, 2023. **640**: p. 119005.
22. Meyes, R., et al., Ablation studies in artificial neural networks. arXiv preprint arXiv:1901.08644, 2019.
23. Shah, B. and H. Bhavsar, Time complexity in deep learning models. Procedia Computer Science, 2022. **215**: p. 202-210.