IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 23 September 2024, accepted 31 October 2024, date of publication 11 November 2024, date of current version 20 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3495531

RESEARCH ARTICLE

Mahalanobis Distance-Based Graph Attention Networks

KONSTANTINA MARDANI, NICHOLAS VRETOS, (Senior Member, IEEE), AND PETROS DARAS, (Senior Member, IEEE)
Information Technologies Institute, Visual Computing Lab, Centre for Research and Technology Hellas, 57001 Thessaloniki, Greece
Corresponding author: Konstantina Mardani (kons_mardani@outlook.com)

This work was supported by European Commission under the Horizon 2020 Program through the FALCON Project, Fight Against Large-Scale Corruption and Organized Crime Networks under Grant GA 101121281 (https://www.falcon-horizon.eu/).

ABSTRACT In this paper, a Mahalanobis Distance-based Graph Attention Network for graph classification, is proposed. In contrast to traditional Graph Attention Networks, the proposed approach learns the covariances between node features so as to determine the attention between nodes, instead of directly learning the attention coefficients using learnable parameter matrices. During training, the network learns the covariance matrix that is essential component of the Mahalanobis distance, and thus adjusts the covariances between node features based on the specific characteristics of the graph under examination. Leveraging Mahalanobis distance, the model manages to capture complex features correlations leading to better graph representations. Additionally, the proposed method combines the concepts of multi-head and multi-view to achieve enhanced performance and generalization ability. Multi-head attention enables the model to focus on diverse aspects of the data, whereas multi-view attention provides different perspectives on node relationships. Extensive experiments on benchmark datasets demonstrate that the proposed method either outperforms or is on par with the state-of-the-art methods. The study also examines the impact of the number of heads and views for the multi-head and multi-view concepts respectively on the proposed method.

INDEX TERMS Graph attention networks, mahalanobis distance, attention mechanism, graph classification.

I. INTRODUCTION

Graph-structured data play a pivotal role in various tasks, requiring effective handling of information, related with the dependencies between graph elements. From simulating complex physics systems [1] and learning molecular fingerprints to predicting protein interfaces [2] and classifying diseases [3], there is a great demand for powerful models capable of learning from graph inputs [4]. Graph Neural Networks (GNNs) have been a leading type of neural networks that capture dependencies in graphs by efficiently performing message passing between nodes and they have demonstrated remarkable capabilities in addressing diverse challenges [4].

Three categories of GNNs can be identified according to the type of prediction: 1) Node-level predictions, which include node classification, regression, clustering, among others. Node classification involves categorizing nodes into predefined classes, whereas node regression predicts continuous values associated with each node. Node clustering seeks to partition nodes into disjoint groups, where nodes with similar properties are grouped together. 2) Edge-level predictions, which include edge classification and link prediction, which require the model to classify edge predefined types or predict whether an edge exists between two given nodes. 3) Graph-level tasks that include graph classification, regression, and matching. All the above-mentioned graph-level tasks require the model to learn graph representations [4].

Recently, multiple variants of GNNs, such as Graph Convolutional Networks (GCNs) [5], Graph Attention Networks (GATs) [6], and Graph Recurrent Networks (GRNs) [7], have attracted significant attention for their remarkable performances in deep learning tasks. GCNs leverage graph

The associate editor coordinating the review of this manuscript and approving it for publication was Zhan-Li Sun.

convolutions to perform node-level predictions efficiently, whereas GATs utilize attention mechanisms to capture node dependencies more effectively. GRNs, on the other hand, capture temporal dependencies in dynamic graph-structured data.

The proposed method is inspired by Graph Attention Networks (GATs) and addresses the graph classification task. Graph classification plays a significant role in various real-world applications, including bioinformatics, social network analysis, and recommendation systems [8]. The ability of GATs to provide discriminative representations by calculating the significance of neighboring nodes, as well as their ability to adapt to different graph structures, along with their shared parameters, forms a set of advantageous features for tackling graph classification tasks.

In this paper, a modified version of the GAT network is proposed that specifies the attention between nodes using the Mahalanobis distance [9]. This distance measures the dissimilarity between vectors in a multivariate space, considering the covariance of the node features. In the proposed approach the covariance matrix is learnable. This enables the proposed Mahalanobis Distance-based Graph Attention Network (MD-GAT) to capture complex correlations and adaptively determine the significance of all neighboring nodes. The main contributions of this paper are:

Mahalanobis Distance-Based Attention: A novel attention mechanism is introduced that utilizes the Mahalanobis distance and its learnable covariance matrix to capture node dependencies in graph-structured data more efficiently.

*Extension of the GAT Network With the **multi-view** Concept:* The proposed MD-GAT extends the GAT network by introducing the *multi-view* concept. Each *view* constructs a different complete graph derived from the input graph and thus, extracts different relations between nodes.

Enhanced Generalization Ability: MD-GAT model enhances the generalization ability of graph classification by learning meaningful representations, using Mahalanobis distance. Extensive experiments were conducted on diverse benchmark datasets for graph classification, demonstrating the superior performance of MD-GAT compared to the state-of-the-art graph classification methods.

II. RELATED WORK

Graph Convolutional Networks can be divided in two categories: spectral and spatial convolution methods. In spectral convolution, for the understanding of the graph's underlying structure, the Laplacian matrix of the graph is used. In the Fourier domain, eigen decomposition on the Laplacian matrix is performed to obtain its eigenvalues and eigenvectors and allows the transformation of the graph data into a more informative representation in the spectral domain. A representative model of this domain is the GCN model [5]. In spatial convolution, instead of using the graph's Laplacian matrix, it directly considers the features of neighboring nodes to understand the node's properties, thus, it operates on the local neighborhood of the node in the graph. Spatial convolutions are computationally simpler and faster compared to spectral convolutions. A representative model of this type of convolution methods is the GAT network [6] and consequently, the proposed MD-GAT method falls within this category of GNNs.

Graph classification defined as the task that aims to associate a label with an entire graph. Except for the message passing over nodes/edges, it also needs to retrieve graph-level representation to achieve accurate predictions. In spatial domain, PSCN [10] proposes a novel framework for learning convolutional neural networks suitable for arbitrary graph data, with various graph types and node/edge attributes, that extracts locally connected regions from graphs, producing efficient feature representations. Similarly, DGCNN [11], which is an end-to-end architecture, attempts to extract information from arbitrary graph structures directly and sorts the vertices of a graph in a consistent order by using a SortPooling layer, enabling traditional neural networks to be trained. Compared to PSCN, DGCNN improves expressibility by allowing gradient backpropagation through SortPooling, dynamically sorts nodes to reduce overfitting to specific orderings, and integrates preprocessing directly into the neural network. On the other hand, KGCNN [12], addresses the challenge of applying CNNs to graph data by integrating graph kernels with classical neural networks, and extracting patches from graphs, creating feature embeddings using graph kernels and feeding them to classical CNNs. In GIN [13], the authors prove that many popular GNN variants cannot learn to distinguish certain simple graph structures and thus, they introduce a most expressive GNN architecture, comparable in power to the Weisfeiler-Lehman (WL) [14] graph isomorphism test and capable to capture different graph structures. Finally, the authors of DGK [15] propose a unified framework that leverages the dependency information between sub-structures to learn latent representations for graphs, inspired by recent advancements in language modeling and deep learning.

In the spectral domain, Defferrard et al. [16] propose the generalization of convolutional neural networks from low-dimensional regular grids (image/video/speech), to high-dimensional irregular grids like graphs, achieving local, stationary and compositional feature learning. DiffPool [17] proposes a differentiable graph pooling module that produces hierarchical graph representations that maps nodes to a set of clusters in each GNN layer, in order to pass this coarsened output to the next GNN layer. Its main limitation is its quadratic space complexity due to the need to maintain an entire assignment matrix that relates nodes from the original graph to the pooled graph, which becomes prohibitive for large graphs. In [18], Cangea et al., propose differentiable graph coarsening, that reduces graph's size adaptively, similarly to an image downsampling. Graph U-net [19] introduces a graph pooling (gPool) operation that reduces graph nodes based on scalar projections and a graph unpooling operation (gUnpool) to reconstruct initial structure using node positions. CapsGNN [20] extracts node features in capsule form and utilizes a routing mechanism to generate expressive high-level capsules and thus, to capture graph-level information. However, CapsGNN fails to distinguish between neighbor and self-connected edges, losing self-information, and introduces noise from multi-hop neighbors, which, with more layers, leads to over-smoothing and difficulty distinguishing node features across classes [21]. Finally, GFN [22] is introduced as a neural set function defined on a set of graph augmented features. These features include both graph structural/topological features and graph propagated features. In contrary to other methods, GFN focuses on the simplicity and efficiency instead of the expressiveness or trainability of GNNs.

PPGN [23] presents a model that utilizes matrix multiplication and Multilayer Perceptrons (MLPs) to achieve provably stronger 3-WL expressiveness compared to message passing models. However, it requires quadratic memory and cubic time complexity, making the model prohibitive for large graphs [24]. To address the problem of pre-set persistence feature weights in existing methods, [25] developed WKPI, which learns an optimal weight-function from labeled data and applies it to persistence summaries for improved graph classification performance. WKPI introduces a learnable positive semi-definite weighted-kernel for computing distances between persistence diagrams. QGNN [26] aims to learn quaternion embeddings for graph-structured data, extending the traditional GCNs to operate within the Quaternion space, which offers highly expressive computations through the Hamilton product.

Moreover, given the success of attention models in deep learning [27], [28], [29], [30], researchers introduced the attention mechanism in graph-based models as well. Over the recent years, numerous approaches [6], [31], [32], [33], [34], [35], [36] have been proposed, each presenting distinct definition of the attention and employing it for a range of valuable objectives. However, all of them have the same objective: the exploitation of attention to effectively focus on the most task-relevant parts for decision making [34]. Notably, attention mechanism has significantly enhanced graph neural networks by accomplishing tasks such as information aggregation for nodes [6], model integration [33], node selection via pooling methods [32] and guiding random-walks through essential nodes [31].

The proposed method, is a modified version of the GAT networks and it is inspired by the MV-AGC [37] approach. MV-AGC is a spectral-based method, which uses the notion of *views*, where in each *view* a different complete graph is constructed from the pairwise Mahalanobis distances of the vertices in the feature space. Each *view* encapsulates a different relation between vertices in a distance metric learning context and this is achieved by learning a different covariance matrix for each *view*, which defines the interdependencies between nodes.

III. METHODOLOGY

In this section, the proposed method for graph classification is thoroughly detailed. For the easy comprehension of the proposed MD-GAT network, a brief overview of the traditional GAT networks is provided. In contrast to the traditional GAT networks, in the proposed method the Mahalanobis distances [9] between node pairs are calculated to represent attention coefficients. The concepts of *multi-head* from GAT [6] and *multi-view* from MV-AGC [37] are combined, as can be seen in Fig. 1, to capture complex correlations and more meaningful node feature dependencies.

A. GRAPH ATTENTION NETWORKS (GATS)

In this subsection, a single graph attention (GAT) layer from [6] is presented. It uses as input a set of node features, $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^d$, where N is the total number of nodes, and d is the number of features in each node. This GAT layer produces a new set of node features, $\{\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_N\}$, $\mathbf{x}'_i \in \mathbb{R}^{d'}$.

In order to transform the initial input features into high-level features, a shared linear transformation parametrized by a weight matrix $\mathbf{W} \in \mathbb{R}^{d' \times d}$, is applied to each node. Subsequently, the attention mechanism computes attention coefficients α_{ij} for each pair of nodes i and j -as long as they are neighbors (*masked attention*)-, determining the importance of node j to node i based on their feature similarities. In this case, the neighborhood $\mathcal{N}_i$ contains all the first-order neighbors of node i. The attention coefficients are computed by the attention mechanism as follows:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_j]\right)\right)}{\sum_{n\in\mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_n]\right)\right)} \tag{1}$$

where α is a single-layer feedforward neural network parametrized by the weight vector $\mathbf{a} \in \mathbb{R}^{2d'}$, $\mathbf{x}_i$ represents the feature vector of node i, $\mathbf{x}_j$ represents the feature vector of node j, $\mathbf{x}_n$ represents the feature vector of node n that belongs to the neighborhood $\mathcal{N}_i$ of node i. $\mathbf{W}$ represents a shared learnable linear transformation that modifies the input graph node features from size d to d'. The output features are then aggregated by applying the concatenation operation $\|$, which requires the weight vector $\mathbf{a}$ that stores the learnable attention coefficients to have size equal to $2d'$ ($\mathbf{a} \in \mathbb{R}^{2d'}$). The LeakyReLU activation introduces non-linearity to the attention mechanism. For easy comparison across different nodes the Softmax function is applied to the attention scores, in order to normalize coefficients across all choices of j.

After normalizing the attention coefficients, they are used to determine the impact of each neighboring node $j \in \mathcal{N}_i$ on the central one i. This is calculated as a weighted combination of the neighboring features, with the weights determined by the attention coefficients. After applying the activation function σ to introduce nonlinearity, the updated, enriched

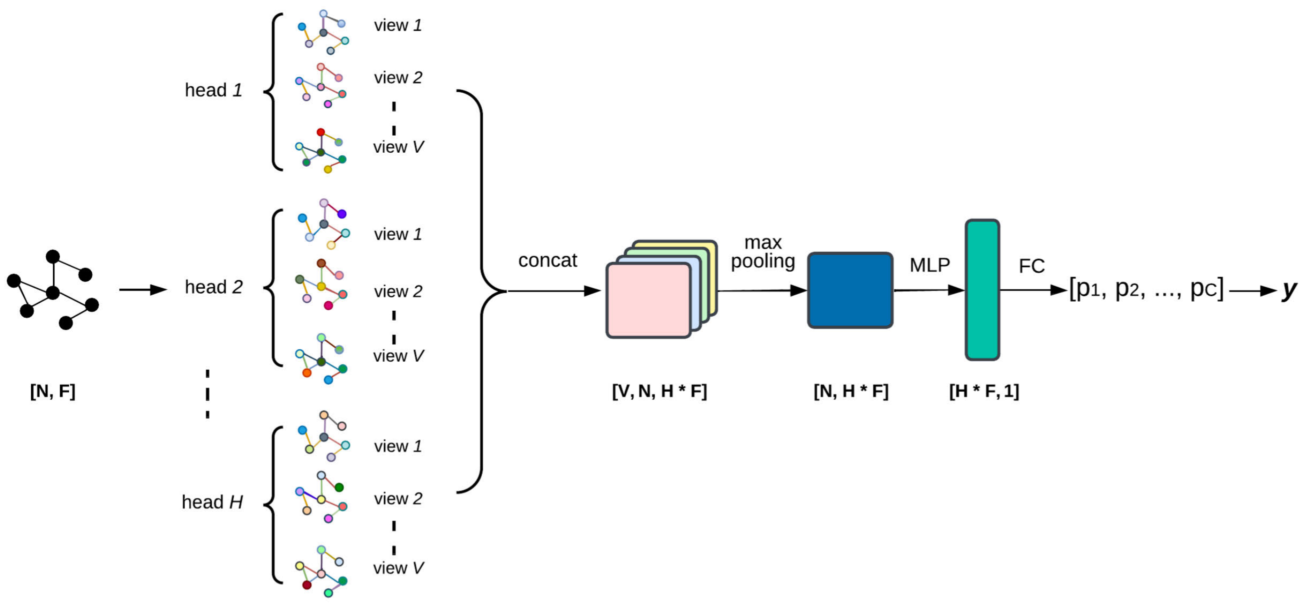


FIGURE 1. **Overview of the proposed MD-GAT network: This figure depicts a single-layer MD-GAT network architecture. Graph-structured data pass through the network, H *heads* are initialized using $\mathbf{W}^h$ where $h \in [1, \ldots, H]$ and in each *head*, V *views* are initialized by using V number of covariance matrices $\mathbf{M}_v$, where $v \in [1, \ldots, V]$. The output of the MD-GAT layer is concatenated, and a max-pooling layer returns the most significant task-related *view*. After passing through an MLP network and two fully-connected layers, the log Softmax gives the final prediction.**

node features are obtained according to (2).

$$\mathbf{x}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}\mathbf{x}_j\right) \tag{2}$$

To enhance the model's performance and stabilize learning process [6], GAT employs multiple attention *heads*, each with its own set of learnable parameters. The attention *heads* $h = \{1, 2, \ldots, H\}$, where H represents the total number of *heads* in a single GAT layer, operate independently on the node features, capturing different patterns and relationships within the graph. The output of each attention *head* h is concatenated and further aggregated to obtain the final node representations:

$$\mathbf{x}'^H_i = \|_{h=1}^H \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha^h_{ij}\mathbf{W}^h\mathbf{x}_j\right) \tag{3}$$

where $\|$ represents concatenation operation, α^h_{ij} are the normalized attention coefficients computed by the h-th attention mechanism (α^h) and $\mathbf{W}^h$ is the weight matrix of head h. The final size of each node feature using *multi-head* attention will be Hd', and not d' (attention with 1 *head*). The attention mechanism in [6] allows the model to dynamically adjust the importance of neighboring nodes based on the learned attention coefficients. The introduction of multiple attention *heads* enables the model to capture diverse patterns and improves the node representations.

B. MAHALANOBIS DISTANCE-BASED GRAPH ATTENTION NETWORKS (MD-GATS)

In this subsection, a modified version of GAT, called Mahalanobis Distance-based Graph Attention Network (MD-GAT) is described. The proposed method is inspired by [37], which introduces *views* to encapsulate different dependencies between vertices, and thus it constructs different complete graphs according to *views*.

Mahalanobis distance can measure the dissimilarity between two feature vectors $\mathbf{x}, \mathbf{y}$ and can be expressed by (4):

$$dist(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{y})} \quad \forall \mathbf{x}, \mathbf{y} \tag{4}$$

where, $\mathbf{C}$ is the covariance matrix of the feature space to which x and y belong, and the terms $^{-1}$ and ^{T} denote the matrix inverse and transpose operations, respectively.

In the proposed approach, where a single MD-GAT layer takes as input a set of nodes $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^d$, where N is the total number of nodes, and d is the number of features in each node, the Mahalanobis distance is computed, after transforming the input features by applying a linear transformation using weight matrix $\mathbf{W} \in \mathbb{R}^{d'\times d}$, which modifies the feature dimensionality from d to d'.

In the *multi-view* concept, each *view* can be seen as a unique perspective or interpretation of the data. Specifically, for each *view* $v \in [1, \ldots, V]$, where V is the total number of *views* in a single MD-GAT layer, a positive semi-definite matrix $\mathbf{M}_v \in \mathbb{R}^{d'\times d'}$ is associated, where $\mathbf{M} = \mathbf{C}^{-1}$, which is a transformation matrix or the inverse of the covariance matrix of the feature space of the nodes. These matrices are learnable and each of them belongs to a single *view*. According to (5), the Mahalanobis distance between the feature vectors of nodes i and j, for the *view* $v \in [1, \dots, V]$ is:

$$dist_v(i,j) = \sqrt{(\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j)^T \mathbf{M}_v (\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j)}\ \forall \mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d \tag{5}$$

$\mathbf{M}_v \in \mathbb{R}^{d' \times d'}$, where d' is the feature dimensionality after applying the linear transformation $\mathbf{W}$, is defined as $\mathbf{M}_v = \mathbf{Q}_v \mathbf{Q}_v^T$, where $\mathbf{Q}_v$ is randomly initialized at the beginning of the learning process.

Furthermore, a feature difference matrix $\mathbf{F} \in \mathbb{R}^{c \times d'}$ is defined, where $c = \frac{1}{2}N(N-1)$ and N is the total number of nodes in the graph. This matrix $\mathbf{F}$ contains all feature differences between unique pairs of nodes i, j, without considering their connectivity and thus, the graph structural information.

Following [37], for the optimization of memory consumption and achieving end-to-end training, unique distance pairs are computed as follows:

$$\begin{aligned}\mathbf{d}_v &= \operatorname{diag}\left(\left((\mathbf{F}\mathbf{M}_v\mathbf{F}^T) \odot \mathbf{I}_c\right)^{1/2}\right) \\ &= ((\mathbf{F}\mathbf{M}_v \odot \mathbf{F})\mathbf{1}_d)^{1/2} \in \mathbb{R}^c \end{aligned}\tag{6}$$

where *diag* is the operator that maps the main diagonal of a $c \times c$ matrix into a vector $\in R_c$ and $1d'$ is a d' element vector of ones. $\mathbf{I}_c$ is the identity matrix of size $c \times c$ and $\odot$ denotes the Hadamard operation. The unique distance pairs in $\mathbf{d}_v$ are then stored in a *view* distance matrix $\mathbf{D}_v \in \mathbb{R}^{N \times N}$. In order to integrate graph structure information into the network, the $\mathbf{D}_v$ is modified to store only distances between connected nodes (*masked attention*).

In MD-GAT, the attention coefficients are computed by integrating this distance matrix $\mathbf{D}_v$ in (1):

$$\alpha_{ij}^v = \exp(\mathbf{D}_v(i,j)) \Big/ \sum_{n \in \mathcal{N}_i} \exp(\mathbf{D}_v(i,n)) \tag{7}$$

where α_{ij}^v stores the normalized attention coefficients between nodes i and j, computed by using the v-th transformation matrix $\mathbf{M}_v$ and $\mathcal{N}_i$ denotes the neighbors of node i.

To enhance the expressive power of the model and capture diverse relationships between nodes, the concept of *multi-head* attention from the traditional Graph Attention Networks (GAT) is introduced to the MD-GAT framework. The *multi-head* mechanism allows the model to focus on different aspects of the data, aiding in better representation learning.

Thus, the MD-GAT framework is extended to include multiple *heads* or perspectives, each characterized by its unique weight matrix $\mathbf{W}^h$, where $h \in [1, \dots, H]$ and H is the total number of *heads* in a single MD-GAT layer. Thus, (7) is modified to:

$$\alpha_{ij}^{vh} = \exp(\mathbf{D}_v^h(i,j)) \Big/ \sum_{n \in \mathcal{N}_i} \exp(\mathbf{D}_v^h(i,n)) \tag{8}$$

where α_{ij}^{vh} denotes the normalized attention coefficients between nodes i and j, computed by using the transformation matrix $\mathbf{M}_v$ from the v-th *view* and the weight matrix $\mathbf{W}^h$ from the h-th *head*. $\mathcal{N}_i$ represents the neighbors of node i. After computing the Mahalanobis distances for all *heads* and *views* and for all nodes, they are concatenated and pass through a max-pooling layer to keep the most significant *view* for the task. The max-pooling layer is used to fuse the different *views* and it is performed in the third dimension (view dimension) as can be seen in Fig. 2.

1) INCORPORATING MAHALANOBIS DISTANCE FOR FEATURE CORRELATION IN ATTENTION MECHANISM

As can be seen in (1), the operation $\mathbf{a}^T[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_j]$ is used to compute the attention scores between nodes. This operation combines concatenation operation and dot product operations to compute the attention coefficients. This implies that GATs assume that all features are totally independent, which is incorrect for complex datasets where features are correlated. By integrating the Mahalanobis distance into the attention mechanism, the model computes attention scores based on a more sophisticated metric that considers the underlying data distribution and this is achieved with the learnable covariance matrix into the Mahalanobis distance calculation. This matrix, allows the model to learn the dependency between node features during training. The existence of the covariance matrix in the Mahalanobis equation is the reason this distance measure has been chosen over alternatives. For instance, if it is assumed that the node features are totally independent, this would mean that $\mathbf{C}^{-1} = \mathbf{I}$ for (4), where $\mathbf{I}$ is the identity matrix. This special case of Mahalanobis distance defines the Euclidean distance (9), which measures the straight-line distance between two feature vectors $\mathbf{x}$, $\mathbf{y}$ in a feature space and is defined as:

$$\begin{aligned} dist_E(\mathbf{x}, \mathbf{y}) &= \sqrt{(\mathbf{x}-\mathbf{y})^T \mathbf{I} (\mathbf{x}-\mathbf{y})} \\ &= \sqrt{(\mathbf{x}-\mathbf{y})^T (\mathbf{x}-\mathbf{y})} \end{aligned}\tag{9}$$

Euclidean distance assumes that features are independent and that each feature contributes equally to the distance calculation. This assumption is often not valid in real-world data where features may be correlated or vary significantly in scale. It computes distance based purely on the raw differences between feature values, which can lead to inaccurate similarity measures if features are not independent. On the other hand, Mahalanobis distance explicitly considers feature correlations by incorporating the covariance matrix $\mathbf{M}_v$. For instance, if two features are highly correlated, the Mahalanobis distance will appropriately reduce their combined influence on the distance calculation. This way, it ensures that features are weighted according to their actual statistical distribution, leading to a more balanced and fair comparison. In section IV. there is an experimental analysis on this topic.

2) COMPUTATIONAL COMPLEXITY OPTIMIZATION

The most computationally intensive operation in MD-GAT is represented by (5), which through linear algebra operations results to (6) to optimize computations by exploiting the symmetry in the distance matrix. This transformation is crucial for efficiently handling pairwise distances within the MD-GAT network.

In order to compute the Mahalanobis distance between the feature vectors of nodes i and j for a specific view v the feature difference matrix $\mathbf{F}$ is introduced. This matrix $\mathbf{F}$ captures the difference between the feature vectors of all possible unique pairs of nodes. Specifically, each entry $\mathbf{F}_{ij}$ in matrix $\mathbf{F}$ is given by:

$$\mathbf{F}_{ij} = \mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j \tag{10}$$

Here, $\mathbf{F} \in \mathbb{R}^{c \times d'}$, where $c = \frac{1}{2}N(N-1)$ represents the total number of unique node pairs and d' denotes the feature dimensionality after applying $\mathbf{W}$.

Next, the matrix product $\mathbf{F}\mathbf{M}_v\mathbf{F}^T$ is computed, where $\mathbf{F}^T$ is the transpose of $\mathbf{F}$. This product results in a $c \times c$ matrix in which each entry (i, j) corresponds to the squared Mahalanobis distance between the features of node pairs i and j:

$$(\mathbf{F}\mathbf{M}_v\mathbf{F}^T)_{ij} = (\mathbf{W}x_i - \mathbf{W}x_j)\mathbf{M}_v(\mathbf{W}x_i - \mathbf{W}x_j)^T \tag{11}$$

Next, the Hadamard product is applied with the identity matrix $\mathbf{I}_c$ on (11), ensuring that only the diagonal elements are considered:

$$(\mathbf{F}\mathbf{M}_v\mathbf{F}^T) \odot \mathbf{I}_c \tag{12}$$

Finally, to obtain the actual Mahalanobis distances, the square root of the diagonal vector is computed by (6).

The resulting vector $\mathbf{d}_v$ contains the Mahalanobis distances for each unique pair of nodes. This vector is then used to construct the distance matrix $\mathbf{D}_v$, which incorporates the graph structure by storing distances only between connected nodes.

These optimizations reduce the time complexity from $O(n^2d^2)$ to 0.5 $O(n^2d^2)$, where n is the number of nodes and d is the dimension of the node feature vectors. For instance, for graphs with $n = 80$ and $d = 128$ the operations are reduced from $104,857,600$ to $52,428,800$. Similarly, the complexity of Hadamard multiplication decreases from $O(n^2d)$ to $0.5O(n^2d)$ per view, resulting in an overall time complexity reduction by a factor of 4. Although the time complexity remains quadratic, the layer's scalability now scales linearly with the number of views. Additionally, regarding space complexity, it is reduced from $O(n^4)$ to $0.5O(n^2d)$. For instance, for graphs with $n = 80$ and $d = 128$ the space complexity is reduced from $40,960,000$ to $819,200$ bytes.The learning space complexity is $O(d^2)$ per view [37].

IV. EXPERIMENTS

In this section, the evaluation benchmarks are described and their statistical information is presented in 2. The experimental setup and the evaluation protocols are thoroughly described and the experimental results in 3 to 7 and in Fig. 3 to 5 are extensively discussed.

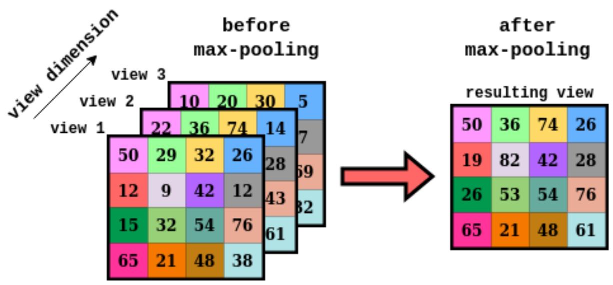


FIGURE 2. This figure depicts the max-pooling process when applied to the concatenated *views*. In this example the number of views is equal to 3.

A. DATASETS

The performance of MD-GAT is evaluated on standard benchmark graph classification datasets, by measuring classification accuracy and standard deviation. These datasets are: 1) MUTAG [38], which consists of edges that correspond to atom bonds and vertices to atom properties, and the goal is to predict the mutaginicity on Salmonella typhimurium, 2) PTC-MR [39], which consists of 344 molecules compounds represented as graphs, where classes indicate carcinogenicity in rats, 3) PROTEINS dataset [40], which consists of 1113 graphs classified as enzymes or non-enzymes. Nodes represent amino acids and edges between nodes exist if they are less than 6 Angstroms apart, 4) IMDB-BINARY [15] and 5) IMDB-MULTI [15], which both of them consist of ego networks of 1000 actors, where nodes represent actors, and edge exists between them if they appear in the same movie and the objective is to find the movie genre, and finally 6) COLLAB [41] dataset, which consists of 5K researchers' ego networks from three fields of Physics. In 2 a statistical summary for each benchmark is described. Additionally, MD-GAT is compared against the state-of-the-art graph classification methods to demonstrate its superiority in capturing complex feature dependencies.

Some of the biological datasets have both discrete vertex labels and continuous-valued vertex attributes. Following all previous works presented here, for fair comparison, only the discrete vertex labels are used for training. Moreover, the social network datasets (IMDB-BINARY, IMDB-MULTI, COLLAB) do not provide any vertex information, thus the one hot encoding of each vertex degree is used as feature vector.

B. EXPERIMENTAL SETUP

For a fair comparison of the MD-GAT network with the state-of-the-art methods, two different evaluation protocols are followed. The comparison methods include both kernel-based and deep learning approaches. The implementation of the MD-GAT network is based on the official codes of [42],

TABLE 1. Abbreviation table for mathematical elements.

Symbol	Meaning	Description
N	Number of nodes	The total number of nodes in the graph
d	Feature dimension	The feature dimension of each node in the graph
$\mathbf{W}$	Weight matrix	A learnable matrix used to transform node features dimensions from d to d'
d'	Transformed feature dimension	The dimension of the node features after applying the weight matrix $\mathbf{W}$
$\mathbf{x}_i$	Node feature vector	The feature vector of node i in the graph
$\mathbf{x}'_i$	Transformed node feature vector	The feature vector of node i in the graph after applying $\mathbf{W}$
$\mathcal{N}_i$	Neighborhood of node i	The set of neighboring nodes directly connected to node i
$\mathbf{a}$	Attention weight vector	A vector used in the attention mechanism to calculate attention scores
$\|$	Concatenation operator	The operation of concatenation
σ	Activation function	A non-linear function (e.g., LeakyReLU) applied to introduce non-linearity in the model
H	Number of attention heads	The total number of attention heads in a GAT layer
V	Number of viewss	The total number of views in a GAT layer
α_{ij}	Attention coefficient	The attention coefficient between nodes i and j
α_{ij}^{vh}	Attention coefficient for view v and head h	The attention coefficient between nodes i and j for view v and attention head h
$\mathbf{M}_v$	Inverse covariance matrix	A learnable positive semi-definite matrix which is the inverse of $\mathbf{C}$
$\mathbf{C}$	Covariance matrix	A matrix representing the covariances of node features
$dist(\mathbf{x}, \mathbf{y})$	Mahalanobis distance	A distance metric that measures the dissimilarity between the feature vectors $\mathbf{x}$ and $\mathbf{y}$
$\mathbf{Q}_v$	Construction matrix for view v	A randomly initialized matrix used to construct the $\mathbf{M}_v$
$\mathbf{F}$	Feature difference matrix	A matrix containing the feature differences between unique pairs of nodes
$\mathbf{d}_v$	Unique distance pairs vector	A vector that stores the unique Mahalanobis distances between connected node pairs in view v
$\mathbf{D}_v$	View distance matrix	A matrix that stores the Mahalanobis distances between nodes for view v
$\odot$	Hadamard product	Hadamard products of two matrices
$dist_E(\mathbf{x}, \mathbf{y})$	Euclidean distance	The Euclidean distance between nodes i and j
$\mathbf{I}_c$	Identity matrix	The identity matrix of size $c \times c$
diag	Diagonal operator	An operator that extracts a diagonal matrix from a square matrix

TABLE 2. Description summary of benchmarks.

Datasets	# Graphs	# Classes	# Nodes(Avg.)	# Edges (Avg.)	Node Labels
MUTAG	188	2	17.93	19.79	Yes
PTC-MR	344	2	14.29	14.69	Yes
PROTEINS	1113	2	39.06	72.82	Yes
IMDB-B	1000	2	19.77	96.53	No
IMDB-M	1500	3	13.00	65.94	No
COLLAB	5000	3	74.49	2457.78	No

TABLE 3. This is a comparison table of mean accuracies along with their standard deviations for the evaluation protocol 1, between the MD-GAT network and the state-of-the-art methods. The bold fonts denote the best mean accuracy for each dataset.

Method	MUTAG	IMDB-B	IMDB-M	PTC-MR	COLLAB	PROTEINS
WL	86.0 ± 1.7	-	-	61.3 ± 1.4	-	75.6 ± 0.4
WL-OA	84.5 ± 1.7	-	-	63.6 ± 1.5	-	76.4 ± 0.4
DGK	87.44 ± 2.72	66.96 ± 0.56	44.55 ± 0.52	60.08 ± 2.55	73.09 ± 0.25	75.68 ± 0.54
PSCN	92.63 ± 4.21	71.00 ± 2.29	45.23 ± 2.84	62.29 ± 5.68	72.60 ± 2.15	75.89 ± 2.76
KGCNN	-	71.45 ± 0.15	47.46 ± 0.21	62.94 ± 1.69	74.93 ± 0.14	75.76 ± 0.28
DiffPool	-	-	-	-	82.13	76.25
CapsGNN	-	71.69 ± 3.40	48.50 ± 4.10	66.01 ± 5.91	77.71 ± 2.51	75.76 ± 0.28
WKPI	88.3 ± 2.6	75.1 ± 1.1	49.05 ± 0.4	-	-	78.5 ± 0.4
GMT	83.44 ± 1.33	73.48 ± 0.76	50.66 ± 0.82	-	80.74 ± 0.54	75.09 ± 0.59
MV-AGC	92.98 ± 5.12	78.20 ± 3.05	53.47 ± 3.62	$\mathbf{74.45 \pm 3.42}$	$\mathbf{82.41 \pm 1.09}$	78.81 ± 3.31
MD-GAT	$\mathbf{93.12 \pm 5.09}$	$\mathbf{78.41 \pm 3.83}$	$\mathbf{53.65 \pm 3.45}$	68.84 ± 6.98	82.36 ± 1.62	$\mathbf{78.9 \pm 2.2}$

TABLE 4. This is a comparison table of mean accuracies along with their standard deviations for the evaluation protocol 2, between the MD-GAT network and the state-of-the-art methods. The bold fonts denote the best mean accuracy for each dataset.

Method	MUTAG	IMDB-B	IMDB-M	PTC-MR	COLLAB	PROTEINS
DGCNN	85.83 ± 1.66	70.03 ± 0.86	47.83 ± 0.85	58.59 ± 2.47	73.76 ± 0.5	76.26 ± 0.24
AttPool	-	-	-	-	79.66	76.50
GFN	90.84 ± 7.22	73.00 ± 4.35	51.80 ± 5.16	-	81.50 ± 2.42	76.46 ± 4.06
PPGN	90.55 ± 8.70	73.00 ± 5.77	50.46 ± 3.59	66.17 ± 6.54	81.38 ± 1.42	77.20 ± 4.73
QGNN	92.59 ± 3.59	77.56 ± 2.45	53.78 ± 3.83	69.92 ± 2.59	81.36 ± 1.31	78.47 ± 3.30
GIN	90.00 ± 8.80	75.10 ± 5.10	52.30 ± 2.80	66.60 ± 6.90	80.60 ± 1.90	76.20 ± 2.60
Graph U-Nets	-	-	-	-	77.56	78.50 ± 0.4
U2GNN	89.97 ± 3.65	77.04 ± 3.45	53.60 ± 3.53	69.63 ± 3.60	77.84 ± 1.48	78.53 ± 4.07
MD-GAT	$\mathbf{93.89 \pm 4.86}$	$\mathbf{78.3 \pm 2.83}$	$\mathbf{54.4 \pm 2.81}$	$\mathbf{70.88 \pm 7.65}$	$\mathbf{81.5 \pm 1.31}$	$\mathbf{78.83 \pm 4.69}$

[43], and [44]. For the experiments, a m-layer MD-GAT network is used. In the proposed method, after applying a dropout layer to the initial node features, they pass to the first MD-GAT layer. The output features from all *heads* and *views* are concatenated to the corresponding dimensions. A max-pooling layer is applied to the *view* dimensions to choose the most significant *view* as can be seen in Fig. 2. The updated features then pass to the next MD-GAT layer and follow the same process. All the MD-GAT layers have hidden dimensions equal to 128. After the last MD-GAT layer and the last max-pooling layer, the features pass through a 2-layer MLP network in order to aggregate all nodes in a single entity. Thus, the MLP network has input dimensions equal to the *number of nodes*, hidden dimensions equal to the *number of heads* $(H) \times$ *hidden dimension of each MD-GAT layer* (128) and output dimensions equal to 1. After the MLP network, a dropout layer is applied. The output features pass to a fully-connected layer with input dimensions equal to the size of the concatenated features after m layers and H heads (*number of MD-GAT layers* $(m) \times$ *number of heads* $(H) \times$ *hidden dimension of each MD-GAT layer* (128)) and the output dimensions are equal to 128. The output of this fully connected layer passes to the Elu layer and then, to the last fully connected layer with output dimensions equal to the *number of classes* for each dataset. Finally, the graph prediction is obtained after applying a log Softmax function. For the experiments GeForce RTX 3090 Ti, RTX A5000, GV102, and GeForce RTX 2070 were used and the code implementation was carried out using PyTorch [45].

C. EVALUATION PROTOCOL 1

The first protocol follows the approach of 10-fold cross-validation similar to [37], and the mean classification accuracy across folds with the standard deviation are calculated and presented in 3. The proposed method consists of $m = 3$ MD-GAT layers with 128 hidden dimensions in each layer, for all datasets. The number of *heads* for each layer ranges from 1 to 4 *heads* and the number of *views* from 1 to 2. All layers have the same number of *heads* and *views*, but different for each dataset. The number of *heads* and *views* was selected after extended experiments for each dataset. In order to prevent overfitting, three dropout layers are used in the MD-GAT network. The first dropout layer in applied directly to the input features and its value ranges from 0.3 for larger datasets to 0.5 for smaller ones. The second dropout layer is applied to the attention coefficients after Softmax layer and its value ranges from 0.0 for larger datasets to 0.5 for the smaller ones. The third dropout layer is applied after MLP layer and before the last two fully connected layers and its value ranges from 0.3 for larger datasets to 0.6 for the smaller ones. For weight initialization, Glorot initialization [46] has being used for $\mathbf{W}$ and uniform initialization with value range (0, 1) for $\mathbf{Q}$. The number of epochs is different for each dataset and it ranges from 50 for COLLAB and PROTEINS datasets to 350 for IMDB-BINARY dataset. For all datasets, the model was trained using Adam optimizer [47], learning rate starting from 0.001 and reducing according to the MultiStepLr scheduler in epochs 20 and 30 with value $\gamma = 0.1$, and weight decay 0.0004. The batch size ranges from 4 for larger datasets to 64 for smaller ones.

D. EVALUATION PROTOCOL 2

The second approach follows the evaluation protocol of [35], where a 10-fold cross-validation is used, and the epoch with the highest accuracy is selected from each fold and the average accuracy over the ten folds is calculated, along with the corresponding standard deviation. This protocol is applied to predefined data splits for fair and clear comparison. For the experiments, $m = 3$ MD-GAT layers with 128 hidden dimensions in each layer were used, for all datasets. The number of *heads* of each layer ranges from 1 to 4 *heads* and the number of *views* from 1 to 3. All layers have the same number of *heads* and *views*, but different for each dataset. The initialization of $\mathbf{W}$ and $\mathbf{Q}$, the values of the three dropout layers, as well as the settings for learning rate and weight decay follow the evaluation protocol 1. The number of epochs is different for each dataset and it ranges from 50 for COLLAB and PROTEINS datasets to 200 for PTC-MR

TABLE 5. This is an experimental results table, that demonstrates the impact of multiple *heads* and *views* in MD-GAT layers for the MUTAG dataset. The bold fonts denote the best combination of number of *heads* and *views* according to the mean accuracies.

MUTAG	V=1	V=2	V=3	V=4	V=5
H=1	91.67 ± 6.55	92.78 ± 5.89	90.00 ± 5.74	90.00 ± 5.11	90.00 ± 7.31
H=2	$\mathbf{93.89} \pm \mathbf{4.86}$	92.78 ± 5.89	91.66 ± 5.40	91.11 ± 7.03	89.44 ± 5.52
H=3	92.22 ± 4.68	90.55 ± 7.43	89.44 ± 6.65	91.11 ± 7.5	91.11 ± 8.36
H=4	92.78 ± 3.75	90.00 ± 4.38	89.44 ± 6.65	92.78 ± 5.27	91.67 ± 5.99

TABLE 6. This is an experimental results table, that demonstrates the impact of multiple *heads* and *views* in MD-GAT layers for the PTC-MR dataset. The bold fonts denote the best combination of number of *heads* and *views* according to the mean accuracies.

PTC-MR	V=1	V=2	V=3	V=4	V=5
H=1	65.88 ± 9.42	67.94 ± 6.99	68.82 ± 7.10	69.71 ± 7.60	68.53 ± 6.51
H=2	70.59 ± 7.96	69.41 ± 7.23	$\mathbf{70.88} \pm \mathbf{7.65}$	68.83 ± 6.08	68.82 ± 6.38
H=3	68.24 ± 4.56	68.82 ± 8.46	66.77 ± 7.85	67.65 ± 6.79	67.06 ± 6.91
H=4	68.53 ± 7.60	67.94 ± 6.99	68.82 ± 7.36	67.94 ± 7.90	68.82 ± 6.38

dataset. The batch size ranges from 4 for larger datasets to 128 for smaller ones.

E. EXPERIMENTAL RESULTS

According to the experimental results presented in 3, which correspond to the evaluation protocol 1, the proposed method outperforms the competitive methods in 4 out of 6 datasets in terms of accuracy. As can be seen in 3, MD-GAT fails to surpass the MV-AGC network in 3 out of 6 datasets in terms of standard deviation. Notably, 2 of these datasets lack label information for vertices, and thus, the one hot encoding of each vertex degree is used as feature vector. Although these feature vectors pass connectivity information into the model, they may lack task-related information about the vertices and this might negatively affect model performance. Additionally, the results on the PTC-MR dataset surpass all previous works except for MV-AGC, which maintains a lead over all other methods by a large margin. The authors of MV-AGC, attribute this success to the critical importance of the provided vertex labels in the dataset for classification. This characteristic does not apply to the proposed method. Generally, the MD-GAT network combines high accuracy and generalization ability yielding either superior or comparable results.

According to the experimental results of 4 that correspond to the evaluation protocol 2, the proposed method outperforms all competitive methods in all datasets. It is clear that MD-GAT performs well on multiple datasets, and this demonstrates a consistent behavior. Furthermore, it achieves lower standard deviations compared to the second-best methods for most datasets, indicating an improved ability of generalization. This clear comparison across all 10 folds in standard splits, as shown in 4, underscores the benefits of using Mahalanobis distances to determine attention coefficients and combining the *multi-head* and *multi-view* concepts. This combination leads to the creation of high-level feature representations, resulting in superior performance in terms of accuracy and enhanced generalization ability.

F. IMPACT OF MULTI-HEADS AND MULTI-VIEWS

In order to highlight the benefits of *heads* and *views* independently, as well as, their combination, experiments were conducted on small datasets (MUTAG, PTC-MR, IMDB-MULTI) without any hyperparameter tuning for multiple number of *heads* and *views*. In 5, 6, and 7, the mean accuracies along with their standard deviations are presented for $V = 1, \ldots, 5$ and $H = 1, \ldots, 4$, where H is the total number of *heads* and V is the total number of *views* in a single layer.

1) DISCUSSION ON ABLATION STUDY FOR MULTI-HEADS AND MULTI-VIEWS

In 5, the mean accuracies and their standard deviations for the MUTAG dataset are presented. As can be seen, this dataset achieves the best performance with $V = 1$ and $H = 2$. Several observations can be made: although in most cases *heads* do not impact the accuracies for $H > 2$, they can improve the generalization ability of the model. On the other hand, high number of *views* do not positively affect the mean accuracies for this dataset at all, but in some cases it may improves the generalization ability of the model. This dataset is the smaller one, therefore it is reasonable to conclude that high number of *heads* and *views* and consequently higher number of trainable parameters will not improve the model performance on this small dataset.

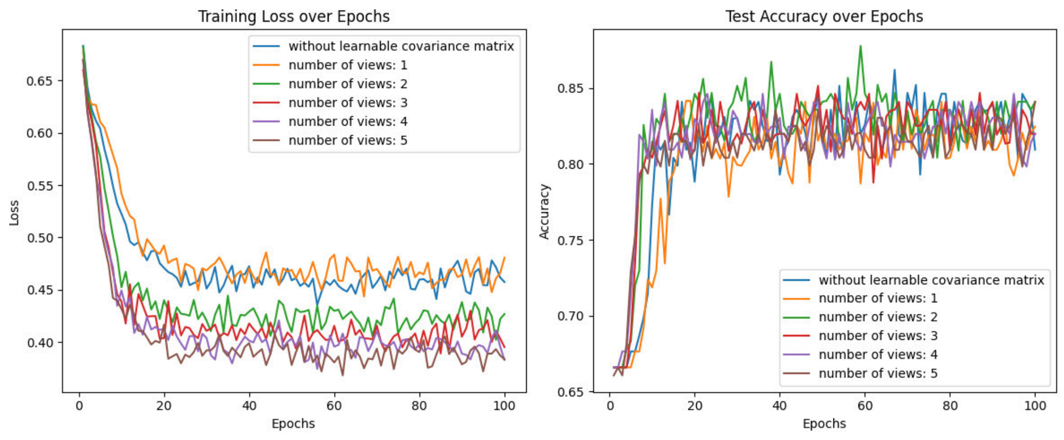


FIGURE 3. **Training loss and test accuracy for MUTAG dataset.**

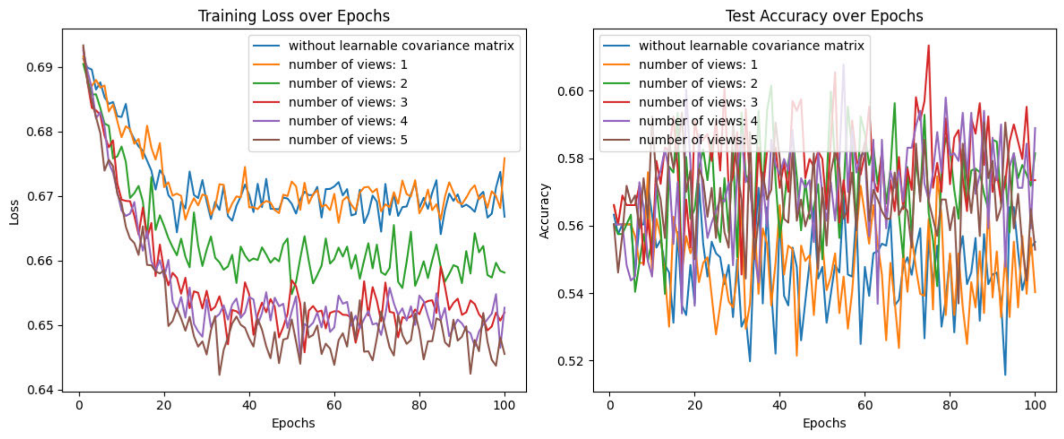


FIGURE 4. **Training loss and test accuracy for PTC-MR dataset.**

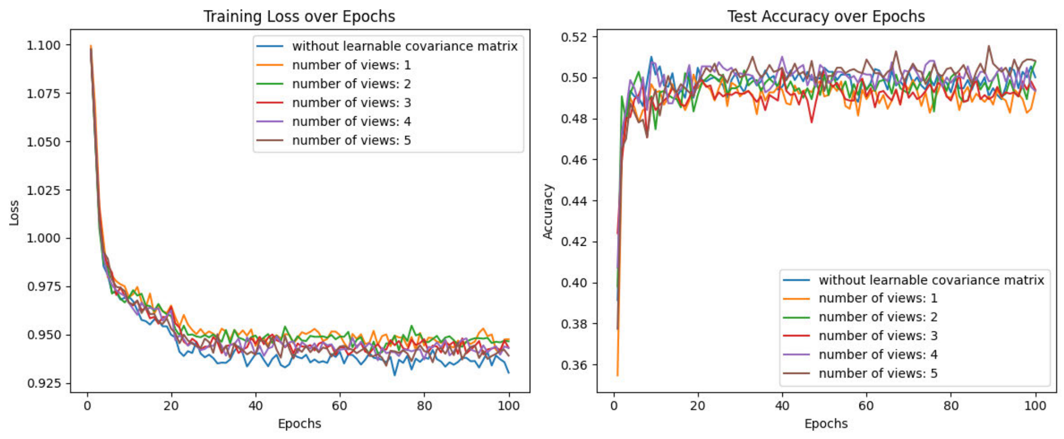


FIGURE 5. **Training loss and test accuracy for IMDB-MULTI dataset.**

In 6, the mean accuracies along with their standard deviations are presented for the PTC-MR dataset. As can be seen, the best accuracy is achieved for $V = 3$ and $H = 2$. This is a small dataset as well, however it is larger than MUTAG. Again, it can be observed that for $H > 2$, *heads* neither improve mean accuracies nor standard deviations. However, *views*, in most cases, improve both mean accuracies and standard deviations. Thus, in this dataset *views* offer higher impact than *heads*.

TABLE 7. This is an experimental results table, that demonstrates the impact of multiple *heads* and *views* in MD-GAT layers for IMDB-MULTI dataset. The bold fonts denote the best combination of number of *heads* and *views* according to the mean accuracies.

IMDB-MULTI	V=1	V=2	V=3	V=4	V=5
H=1	53.8 ± 3.25	53.27 ± 3.12	53.53 ± 3.69	53.33 ± 3.78	53.67 ± 2.52
H=2	54.2 ± 3.10	54.33 ± 2.99	$\mathbf{54.40 \pm 2.81}$	54.33 ± 2.95	53.80 ± 2.83
H=3	54.2 ± 3.51	53.40 ± 2.85	54.33 ± 3.00	53.80 ± 2.52	53.87 ± 3.34
H=4	54.0 ± 3.54	54.40 ± 3.43	53.87 ± 2.98	54.40 ± 3.54	53.53 ± 2.71

In 7, the mean accuracies along with their standard deviations are presented for the IMDB-MULTI dataset. Similarly to the PTC-MR dataset, the top accuracy is achieved with 2 *heads* and 3 *views*. In contrast to both previous datasets, on these experiments for the IMDB-MULTI dataset, both *heads* and *views*, offer improved performance and enhanced generalization ability.

2) CONCLUSIONS FOR MULTI-HEADS

The addition of multiple heads in MD-GAT layers is primarily intended to enhance feature extraction capabilities by allowing the model to focus on different parts of the graph data simultaneously. For the MUTAG dataset, increasing the number of heads beyond two does not lead to substantial improvements in accuracy. This is indicative of the dataset's size limitations, where the benefits of additional heads are overshadowed by the risk of overfitting. However, in larger datasets like IMDB-MULTI, the presence of more heads contributes positively to the model's performance, suggesting that the model can leverage multiple heads to extract richer and more varied features effectively. Moreover, in terms of generalization ability, as can been from the standard deviations, all the datasets show to benefit from multiple heads, in some cases more than others. For instance, in the MUTAG dataset, it can be seen that the standard deviation is reduced as more heads are added to the model when $V = 1$. On the other hand, in the PTC-MR dataset $V = 2$ shows that additional heads do not improve the model's generalizability.

3) CONCLUSION FOR MULTI-VIEWS

The impact of varying the number of views is more pronounced. Increasing the number of views allows the model to learn from multiple graph representations, each capturing different structural aspects of the data. This is particularly beneficial in larger datasets. For instance, for MUTAG dataset which is the smaller one, additional views do not yield significant performance improvements but may still contribute to the generalization ability by providing varied perspectives. In contrast, the PTC-MR dataset which is larger that MUTAG, demonstrates that additional views improve both accuracy and standard deviation, indicating that multiple views help the model to generalize better. Additional experiments were conducted to show the influence of multiple views in the convergence rate and stability during training. The multiple views imply the initialization of multiple covariance matrices as mentioned in section III. The experiments in Fig. 3 to 5 examine the case where the covariance matrix is a fixed matrix, specifically the identity matrix (Euclidean distance) and the cases where the covariance matrices are learnable and their number varies from 1 to 5 (*mutli-view* concept). All the experiments assume that $H = 1$ and all the settings for each dataset (MUTAG, PTC-MR, IMDB-MULTI) are the same except for the number of views V, that will be examined. As can be seen, for all datasets especially for MUTAG and PTC-MR, the learnable covariance matrices contribute positively to the converge rate, especially when their number increases (*multi-views*). On the contrary, the stability during the training process presents a minor deterioration.

V. CONCLUSION

MD-GAT offers a novel approach to graph classification, leveraging the Mahalanobis distance to improve the attentiveness of nodes and capture more meaningful relationships in graph-structured data. After extensive experiments, the choice of the number of *heads* and *views* depends on the dataset size and the information that is stored on vertex level. Larger datasets might benefit from multiple sets of *heads* and *views*, while smaller datasets may achieve their best performance with fewer *heads* and *views*. It is essential to perform hyperparameter tuning to find the optimal configuration for each specific dataset. For future research, it would be interesting to explore alternatives to the max-pooling layer, aiming to make a more informed selection of the most significant *view*. Additionally, conducting experiments with an increased number of GAT layers could be beneficial to enhance both performance and generalization capabilities. Moreover, by considering higher-order neighboring nodes as part of the neighborhood, the proposed method may capture more complex dependencies between nodes. Lastly, the reduction of computational complexity would be valuable for the MD-GAT model to run more efficiently, making it more suitable for large-scale applications.

REFERENCES

[1] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 119, H. Daumé III and A. Singh, Eds., 2020, pp. 8459–8468.

[2] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.

[3] Z. Sun, H. Yin, H. Chen, T. Chen, L. Cui, and F. Yang, ''Disease prediction via graph neural networks,'' *IEEE J. Biomed. Health Informat.*, vol. 25, no. 3, pp. 818–826, Mar. 2021.

[4] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, ''Graph neural networks: A review of methods and applications,'' *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[5] T. N. Kipf and M. Welling, ''Semi-supervised classification with graph convolutional networks,'' 2016, *arXiv:1609.02907*.

[6] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, ''Graph attention networks,'' *Stat*, vol. 1050, no. 20, p. 48550, 2017.

[7] L. Yujia, T. Daniel, B. Marc, and Z. Richard, ''Gated graph sequence neural networks,'' in *Proc. Int. Conf. Learn. Represent.*, vol. 22, 2016, pp. 1–20.

[8] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, ''A comprehensive survey on graph neural networks,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[9] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, ''The Mahalanobis distance,'' *Chemometrics Intell. Lab. Syst.*, vol. 50, no. 1, pp. 1–18, 2000.

[10] M. Niepert, M. Ahmed, and K. Kutzkov, ''Learning convolutional neural networks for graphs,'' in *Proc. Int. Conf. Mach. Learn. (ICML)*, M. F. Balcan and K. Q. Weinberger, Eds., New York, NY, USA, 2016, pp. 2014–2023.

[11] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, ''An end-to-end deep learning architecture for graph classification,'' in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 1–8.

[12] G. Nikolentzos, P. Meladianos, A. J.-P. Tixier, K. Skianis, and M. Vazirgiannis, ''Kernel graph convolutional neural networks,'' in *Proc. 27th Int. Conf. Artif. Neural Netw.*, Rhodes, Greece. Berlin, Germany: Springer, Oct. 2018, pp. 22–32.

[13] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, ''How powerful are graph neural networks?'' 2018, *arXiv:1810.00826*.

[14] N. Shervashidze, P. Schweitzer, E. Jan Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, ''Weisfeiler-Lehman graph kernels,'' *J. Mach. Learn. Res.*, vol. 12, no. 9, pp. 2539–2561, 2011.

[15] P. Yanardag and S. V. N. Vishwanathan, ''Deep graph kernels,'' in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 1365–1374.

[16] M. Defferrard, X. Bresson, and P. Vandergheynst, ''Convolutional neural networks on graphs with fast localized spectral filtering,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.

[17] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, ''Hierarchical graph representation learning with differentiable pooling,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.

[18] C. Cangea, P. Veličković, N. Jovanović, T. Kipf, and P. Lió, ''Towards sparse hierarchical graph classifiers,'' 2018, *arXiv:1811.01287*.

[19] H. Gao and S. Ji, ''Graph U-Net,'' in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2083–2092.

[20] Z. Xinyi and L. Chen, ''Capsule graph neural network,'' in *Int. Conf. Learn. Represent.*, 2019, pp. 1–16.

[21] X. Zuo, H. Yuan, B. Yang, H. Wang, and Y. Wang, ''Exploring graph capsual network and graphormer for graph classification,'' *Inf. Sci.*, vol. 640, Sep. 2023, Art. no. 119045.

[22] T. Chen, S. Bian, and Y. Sun, ''Are powerful graph neural nets necessary? A dissection on graph classification,'' 2019, *arXiv:1905.04579*.

[23] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman, ''Provably powerful graph networks,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.

[24] M. Balcilar, P. Héroux, B. Gauzere, P. Vasseur, S. Adam, and P. Honeine, ''Breaking the limits of message passing graph neural networks,'' in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 599–608.

[25] Q. Zhao and Y. Wang, ''Learning metrics for persistence-based summaries and applications for graph classification,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Cambridge, MA, USA: Curran Associates, 2019, pp. 1–12.

[26] T. D. Nguyen and D. Phung, ''Quaternion graph neural networks,'' in *Proc. Asian Conf. Mach. Learn.*, 2021, pp. 236–251.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, ''Attention is all you need,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ''BERT: Pre-training of deep bidirectional transformers for language understanding,'' 2018, *arXiv:1810.04805*.

[29] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, ''An image is worth 16×16 words: Transformers for image recognition at scale,'' 2020, *arXiv:2010.11929*.

[30] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, ''End-to-end object detection with transformers,'' in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2020, pp. 213–229.

[31] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, and A. A. Alemi, ''Watch your step: Learning node embeddings via graph attention,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.

[32] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, ''Attention-based graph neural network for semi-supervised learning,'' 2018, *arXiv:1803.03735*.

[33] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, ''Gated graph sequence neural networks,'' 2015, *arXiv:1511.05493*.

[34] J. Huang, Z. Li, N. Li, S. Liu, and G. Li, ''AttPool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism,'' in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6479–6488.

[35] D. Q. Nguyen, T. D. Nguyen, and D. Phung, ''Universal graph transformer self-attention networks,'' 2019, *arXiv:1909.11855*.

[36] J. Baek, M. Kang, and S. Ju Hwang, ''Accurate learning of graph representations with graph multiset pooling,'' 2021, *arXiv:2102.11533*.

[37] N. Adaloglou, N. Vretos, and P. Daras, ''Multi-view adaptive graph convolutions for graph classification,'' in *Proc. 16th Eur. Conf.*, Glasgow, U.K. Berlin, Germany: Springer, Aug. 2020, pp. 398–414.

[38] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, ''Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity,'' *J. Medicinal Chem.*, vol. 34, no. 2, pp. 786–797, Feb. 1991.

[39] H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma, ''Statistical evaluation of the predictive toxicology challenge 2000–2001,'' *Bioinformatics*, vol. 19, no. 10, pp. 1183–1193, Jul. 2003.

[40] K. M. Borgwardt, C. S. Ong, S. Schonauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel, ''Protein function prediction via graph kernels,'' *Bioinformatics*, vol. 21, pp. 47–56, Jun. 2005.

[41] P. Yanardag and S. V. N. Vishwanathan, ''A structural smoothing framework for robust graph comparison,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.

[42] B. Knyazev, ''Graph classification with graph convolutional networks in PyTorch,'' Inst. Artif. Intell., Univ. Guelph, Vector, Guelph, ON, Canada, 2018.

[43] Yong (Qbxlvnf11). (2023). *Graph-Neural-Networks-for-Graph-Classification*. Accessed: 28-April-2023. [Online]. Available: https://github.com/qbxlvnf11/graph-neural-networks-for-graph-classification

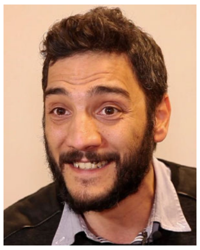
[44] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and Y. Bengio, ''Graph attention networks,'' in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and L. Antiga, ''PyTorch: An imperative style, high-performance deep learning library,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 32, 2019, pp. 1–12.

[46] X. Glorot and Y. Bengio, ''Understanding the difficulty of training deep feedforward neural networks,'' in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[47] D. P. Kingma and J. Ba, ''Adam: A method for stochastic optimization,'' 2014, *arXiv:1412.6980*.

KONSTANTINA MARDANI received the Diploma degree in electrical and computer engineering from the Democritus University of Thrace (D.U.Th.), Xanthi, Greece, in 2019. Since January 2020, she has been a Research Associate with the Information Technologies Institute (ITI), Centre for Research and Technology Hellas (CERTH), Thessaloniki, Greece. She has participated in three European projects. Her research interests include computer vision, natural language processing, and graph networks.



NICHOLAS VRETOS (Senior Member, IEEE) received the B.Sc. degree in computer science from University Pierre et Marie Curie (Paris VI), in 2002, and the Ph.D. degree from the Aristotle University of Thessaloniki, in 2012. He is currently a Researcher with VCL/ITI-CERTH. He worked in 12 European projects as a Technical Manager/WP Leader/Researcher. He has published more than 70 articles in scientific journals and conference proceedings and a book chapter. He has committed as a reviewer for several journals and conferences in the field of image and video processing. His main research interests include image and video processing, semantic analysis, neural networks, and 3-D data processing.

PETROS DARAS (Senior Member, IEEE) received the Diploma degree in electrical and computer engineering and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Aristotle University of Thessaloniki, Greece, in 1999, 2002, and 2005, respectively. He is currently a Research Director and the Chair of the Visual Computing Lab, Information Technologies Institute. He has been involved in more than 50 projects, funded by the EC and the Greek Ministry of Research and Technology. His main research interests include visual content processing, multimedia indexing, search engines, recommendation algorithms, and relevance feedback. His involvement with those research areas has led to the coauthoring of more than 300 articles in refereed journals and international conferences.