Schema-Aware Deep Graph Convolutional Networks for Heterogeneous Graphs

Saurav Manchanda
University of Minnesota
Twin Cities, USA
manch043@umn.edu

Da Zheng
AWS AI Labs
Palo Alto, USA
dzzhen@amazon.com

George Karypis
AWS AI Labs
Palo Alto, USA
gkarypis@amazon.com

***Abstract*—Graph convolutional network (GCN) based approaches have achieved significant progress for solving complex, graph-structured problems. GCNs incorporate the graph structure information and the node (or edge) features through message passing and computes 'deep' node representations. Despite significant progress in the field, designing GCN architectures for heterogeneous graphs still remains an open challenge. Due to the schema of a heterogeneous graph, useful information may reside multiple hops away. A key question is how to perform message passing to incorporate information of neighbors multiple hops away while avoiding the well-known over-smoothing problem in GCNs. To address this question, we propose our GCN framework *Deep Heterogeneous Graph Convolutional Network (DHGCN)*, which takes advantage of the schema of a heterogeneous graph and uses a hierarchical approach to effectively utilize information many hops away. It first computes representations of the target nodes based on their *schema-derived ego-network* (SEN). It then links the nodes of the same type with various pre-defined metapaths and performs message passing along these links to compute final node representations. Our design choices naturally capture the way a heterogeneous graph is generated from the schema. The experimental results on real and synthetic datasets corroborate the design choice and illustrate the performance gains relative to competing alternatives.**

***Index Terms*—graph convolutional networks, graph neural networks, metapath, heterogeneous graph, relational graph, over-smoothing, network schema.**

I. Introduction

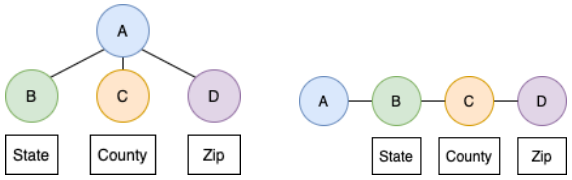
Many real-world problems are naturally represented as graphs. Inspired by the success of Convolutional Neural Networks (CNNs) [1] in computer vision, Graph Convolution Networks (GCNs) [2, 3, 4, 5, 6, 7] defined on graphs have emerged as one of the most powerful tools for solving graph problems. GCNs apply a convolutional layer in the neighborhood of a node to generate node representation, where the neighbors generally correspond to directly connected nodes. By stacking multiple such convolutional layers, GCNs compute *deep* node representations by using information from nodes multiple hops away.

While there is significant progress in the field, designing GCN architectures for heterogeneous graphs still remains an open challenge. Heterogeneous graphs consist of multiple node types and relation types and can be viewed as graphs with pre-defined network schema, where some nodes are entity nodes and other nodes are attribute nodes describing these entity nodes. The same entity can be represented using different subgraphs of attributes. For example, Figure 1 shows two network schemas that model the same information. Depending on the schema, some attribute nodes may be far away from entity nodes. This gives rise to the question: can we design a GCN architecture for heterogeneous graphs such that it can effectively get information from attribute nodes many hops away as well as entity nodes of the same type.

Some heterogeneous GNN models, such as Relational Graph Convolution Network (RGCN) [8], performs message passing with direct neighbors. One may argue that in order to get information from nodes multiple hops away we just need to put more RGCN layers. However, there are two limitations with such an approach: (i) Since important information can be present many hops away, the number of RGCN layers can grow very large. However, by increasing the number of RGCN layers, we also bring irrelevant information to the model. (ii) GNNs with multiple layers suffer from an over-smoothing problem [9, 10, 11], in which all nodes will converge to the same representation. Another line of research on heterogeneous graphs takes advantage of metapaths [12, 13]. By using metapaths, these models can reach the information on the nodes multiple hops away. For example, HAN [13] performs message passing between nodes of the same type that are linked with pre-defined metapaths. However, HAN only uses the information on the endpoint nodes of the metapath and ignores all nodes in the ego-network of these endpoint nodes as well as the nodes along with the metapath. Thus, HAN can potentially miss much information in a heterogeneous graph. MAGNN [12] improves HAN by incorporating the information of the nodes along the metapath but still misses the information in the ego-network of the endpoint nodes.

We introduce the *Deep Heterogeneous Graph Convolutional Network* (DHGCN), a purposefully designed class of GCN models for heterogeneous graphs. DHGCN uses a two-step schema-aware hierarchical approach. Without loss of generality, assume that we are interested in making predictions for a node of node type $\mathcal{A}$. In the first step, DHGCN aggregates information from the immediate ego-network that is derived from the schema. Second, DHGCN aggregates information from the neighbors of type $\mathcal{A}$ using metapath-based convolutions, i.e., graph convolutions on the metapath-guided neighbors. In a

Work was done when the first author was an intern at AWS.



(a) (b)

Fig. 1: The same entity 'address' can be represented in multiple ways, depending upon the design of the network schema.

heterogeneous graph, we believe that our proposed framework is the right way to aggregate the information because it captures the way the graph was generated using the schema.

We present extensive experiments using various real and synthetic datasets to explore DHGCN's design space and assess its performance. Though DHGCN can be used for many problems that involve learning over graphs, we evaluate our approaches on node classification. We show that as the complexity of the graph increases, DHGCN outperforms the competing approaches. On various datasets, DHGCN achieves up to 10.4% improvement over the competing approaches.

The remainder of the paper is organized as follows. Section II corresponds to the definitions, notations and the background used in the paper. The paper discusses the proposed methods in Section IV followed by the experiments in Section V. Section VI discusses the results. Finally, Section VII provides some concluding remarks.

II. Notation and Background

A. Heterogeneous graphs

A heterogeneous graph is a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, where each node $v \in \mathcal{V}$ and each edge $e \in \mathcal{E}$ are associated with their type mapping functions $\tau(v) : \mathcal{V} \rightarrow \mathcal{A}$ and $\phi(e) : \mathcal{E} \rightarrow \mathcal{R}$, respectively. A metapath P on the heterogeneous graph $\mathcal{G}$ is defined as a path on $\mathcal{G}$'s network schema, of the form $A_1 \xrightarrow{R_{1,2}} A_2 \xrightarrow{R_{2,3}} \dots \xrightarrow{R_{n-1,n}} A_n$, which describes a composite relation $R = R_{1,2} \circ R_{2,3} \circ \dots \circ R_{n-1,n}$ between node types A_1 and A_n. Given a metapath P of a heterogeneous graph, a metapath instance p of P is defined as a node sequence in the graph following the schema defined by P.

B. Metapath-guided neighbors

Given a metapath P, where P is of the form $A_1 \xrightarrow{R_{1,2}} A_2 \xrightarrow{R_{2,3}} \dots \xrightarrow{R_{n-1,n}} A_n$, the metapath-guided neighbors of a node $v \in A_1$ is defined as the set of all the nodes, belonging to the same type A_n, that are reachable from v, following the metapath P.

When the metapath P of the form $A_1 \xrightarrow{R_{1,2}} A_2 \xrightarrow{R_{2,3}} \dots \xrightarrow{R_{n-1,n}} A_n$ has $A_1 = A_n$, the metapath-guided neighbors are of the same type A_1 (or A_n). We call these metapath-guided neighbors as the same-type metapath-guided neighbors, and these neighbors carry a special semantic meaning. For example, in a scholar network with authors, papers, and venues, Author-Paper-Author (APA) and Author-Paper-Venue-Paper-Author (APVPA) are symmetric-metapaths describing two different relations among authors. The APA metapath associates two co-authors, while the APVPA metapath associates two authors who published papers in the same venue.

TABLE I: Notation used throughout the paper.

Symbol	Description
$\mathcal{G}$	A heterogeneous graph.
$\mathcal{V}$	Set of nodes in the graph $\mathcal{G}$.
$\mathcal{E}$	Set of edges in the graph $\mathcal{G}$.
$\mathcal{A}$	Set of node types in the graph $\mathcal{G}$.
$\mathcal{R}$	Set of edge types in the graph $\mathcal{G}$.
$\tau(v)$	Type mapping function $\tau(v) : \mathcal{V} \rightarrow \mathcal{A}$, that maps a node to its corresponding type in the graph $\mathcal{G}$.
$\phi(e)$	Type mapping function $\tau(e) : \mathcal{E} \rightarrow \mathcal{R}$, that maps an edge to its corresponding type in the graph $\mathcal{G}$.
$\psi(p)$	Type mapping function that maps a metapath instance to its corresponding metapath in the graph $\mathcal{G}$.
P	A metapath P on the heterogeneous graph $\mathcal{G}$ is defined as a path on $\mathcal{G}$'s network schema, of the form $A_1 \xrightarrow{R_{1,2}} A_2 \xrightarrow{R_{2,3}} \dots \xrightarrow{R_{n-1,n}} A_n$.
E_a	The *schema-derived ego-network* (SEN) of node a of graph $\mathcal{G}$
z_a	Aggregated information of the SEN E_a
$N^l(v)$	A function that defines the neighbor nodes of a node $v \in \mathcal{V}$ for lth layer in a GCN framework.
$N^l_m(v)$	A function that defines the metapath m guided neighbor nodes of a node $v \in \mathcal{V}$ for lth layer in a GCN framework.
$q^l_{m,\hat{v}\rightarrow v}$	Path-aware attention-weight for the message from node $\hat{v}$ to the node v through an instance of the metapath m, for layer l of metapath-based convolutions.
$u^l_{m,\hat{v}\rightarrow v}$	Unnormalized attention-weight corresponding to $q^l_{m,\hat{v}\rightarrow v}$, estimated as a function of source node features, destination node features, in addition to metapath-features.
W^b_a	Projection matrix, the subscripts and superscripts are used to differentiate between the different projection matrices.
f_v	Features associated with a node v.
h_v	Representation of a node v.
h^l_v	Representation of a node v at layer l.
σ	Non-linear operation.

C. Message passing

Graph neural network models can be formulated in the form of message passing [14], in which each node of a graph broadcasts messages to its neighbors and compute its node representation with its own features as well as the messages from its neighbors. A general form of the message passing mechanism has four functions:

Neighbor identification: A function $N^l(v)$ that defines neighbor nodes of a node $v \in \mathcal{V}$ for lth layer.

Message construction: A function that maps the representation $h_{\hat{v}}$ of a neighbor node $\hat{v} \in N^l(v)$ of a target node v to form a message:

$$m^l_{\hat{v}\rightarrow v} = \text{MESSAGE}^l(h^{l-1}_{\hat{v}}, h^{l-1}_v, c(\hat{v}, v)),$$

where $c(\hat{v}, v)$ is the connectivity information between $\hat{v}$ and v. For example, when $N^l(v)$ corresponds to directly connected

nodes of v, $c(\hat{v}, v)$ can be the features on the edge connecting v and $\hat{v}$, i.e., $e^l_{\hat{v}\to v}$.

Message aggregation: A function that gathers the messages from a node v's neighbors:

$$m^l_v = \text{AGGREGATE}^l(\{m_{\hat{v}\to v} | \hat{v} \in N^l(v)\})$$

Node update: A function that updates the node v's representation by the aggregated message m^l_v and its representation at previous layer h^{l-1}_v

$$h^l_v = \text{UPDATE}^l(m^l_v, h^{l-1}_v)$$

All of these functions can be defined with neural networks and can be learned for a downstream task.

III. Related work

Many research works have been conducted to develop machine learning models for graphs. Earlier works [15, 16, 17, 18] developed unsupervised models to learn representations of nodes in a graph. The representations can be used in many downstream tasks. Although these unsupervised models are powerful, they have some limitations. First, they are not able to incorporate node or edge features in the learned node representations. Secondly, the representations are learned with supervisions that are different from the downstream tasks.

Graph neural networks arose in popularity in the past few years. They apply deep neural networks on graph data and can address the limitations in the earlier representation learning methods. GNN models apply message passing on graphs and compute node representations with input node/edge features and graph structures. These models can be trained with any downstream tasks to learn node representations specifically for these tasks and achieve better performance.

The earlier GNN models [2, 4, 5, 7] apply message passing on homogeneous graphs. The initial GCN network, proposed by [5], uses all one-hop direct nodes to v, in addition to v (i.e., self-edge) as $N^l(v)$. The MESSAGE function is modeled as a projection of $h^{l-1}_{\hat{v}}$ through a weight matrix W^l. The AGGREGATE function is modeled by taking sum of the incoming messages and normalizing it. The UPDATE function simply returns the aggregated message m^l_v followed by a non-linearity σ. Formally,

$$h^l_v = \sigma\left(\sum_{\hat{v}\in N^l(v)} \frac{1}{d_{v\hat{v}}} W^l h^{l-1}_{\hat{v}}\right),$$

where $d_{v\hat{v}} = \sqrt{|N(v)||N(\hat{v})|}$ is a normalization constant based on graph structure. Graph attention network (GAT) [7] extends GCN by introducing the attention mechanism as a substitute for the statically normalized convolution operation. The attention weights are used to do a weighted combination of the messages in the AGGREGATE function, as compared to the vanilla sum used in GCN. We can apply message passing on a graph multiple times to have a node to gather information from nodes multiple hops away, instead of just direct neighbors.

The message passing formulation of graph neural networks is extended to heterogeneous graphs. Relational Graph Convolution Network (RGCN) [8] uses the type-specific projection matrix for message construction, i.e.,

$$h^l_v = \sigma\left(\sum_{\hat{v}\in N^l(v)} \frac{1}{d_{v\hat{v}}} W^l_{\phi(e(\hat{v},v))} h^{l-1}_{\hat{v}}\right),$$

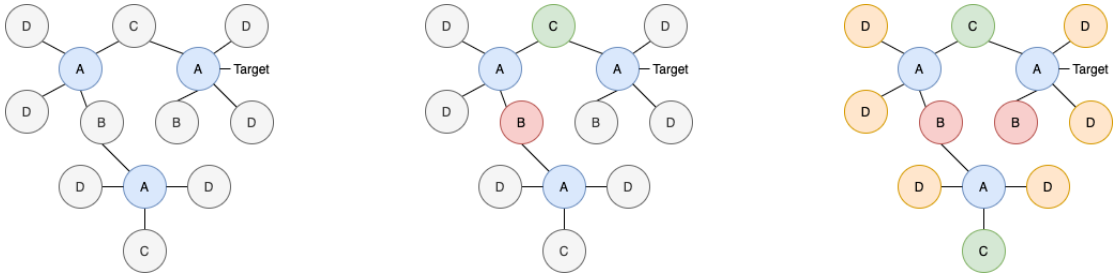
where $W^l_{\phi(e(\hat{v},v))}$ is a type-specific projection matrix and $\phi(e(\hat{v}, v))$ gives the edge-type (relation) between the nodes $\hat{v}$ and v. Heterogeneous graph transformer [19] applies the attention mechanism to heterogeneous graphs. The attention score on each edge is parametrized by the relation type.

A major problem of GNN models is that they suffer from an over-smoothing problem when multiple GNN layers are deployed to access data from neighbors multiple hops away [20]. The problem is even more pronounced in the case of heterogeneous graphs, where the nodes of interest can be present at arbitrary depth, depending upon the underlying network schema.

To address this challenge, several approaches have been developed to use the metapaths to guide neighbor selection, allowing the approaches to get information from distant nodes. The approaches that use metapath-guided neighbor selection include Heterogeneous Graph Attention Network (HAN) [13] and Metapath Aggregated Graph Neural Network (MAGNN) [12]. These approaches differ in their neighbor-identification approach. While HAN only uses the metapath-guided neighbors as the neighbor nodes, MAGNN extends HAN to compute the representation with the information of the nodes that appear on the metapath. The ego-networks explored by HAN and MAGNN are shown in Figures 2a and 2b, respectively. Both HAN and MAGNN are prone to missing out some nodes, which might provide important signals for the task at hand.

Another line of work that is related to the approach proposed in the paper estimates node representations that are aware of the network schema. Examples include (i) Network Schema Preserving Heterogeneous Information Network Embedding (NSHE) [21], which estimates node-representations that are aware of the subgraph imposed by the schema, and not just aware of the locally connected neighborhood; and (ii) Tree structure-aware Graph Neural Network (TGNN) [22], which decomposes the schema into different trees, which can denote different organizational forms of neighbors, and hierarchically aggregates the information from these derived trees. By design, the neighborhood over the the information is aggregated in the above approaches is simply hard-coded based on the network-schema, thus, these approaches lack the notion of GCN "layers". Therefore, these approaches could potentially miss out important information that is present beyond the schema-derived neighborhood.

In this paper, we develop novel approaches to address the limitations of prior approaches.



(a) HAN (b) MAGNN (c) DHGCN

Fig. 2: Ego-networks explored by various approaches for a given target-node belonging to type A (greyed-out nodes are the ones not explored). HAN only explores the metapath-guided neighbors of the same-type A. MAGNN also explores other nodes that appear on the metapath $A...A$, thus not exploring the neighbors which are not part of the metapath. The proposed approaches explore the complete ego-network, without suffering through the oversmoothing problems faced by RGCN.

IV. Deep Heterogenous Graph Convolutional Networks

We introduce the *Deep Heterogeneous Graph Convolutional Network* (DHGCN), a purposefully designed class of GCN models for heterogeneous graphs. DHGCN uses a two-step schema-aware hierarchical approach. Without loss of generality, assume that we are interested in making predictions for a node of type $\mathcal{A}$. In the first step, DHGCN aggregates information from the immediate ego-network that is derived from the schema. Second, DHGCN aggregates information from the neighbors of type $\mathcal{A}$ using metapath-based convolutions, i.e., graph convolutions on the same-type metapath-guided neighbors. Figure 3 gives an overview of the two-step aggregation strategy of the proposed framework.

In the remaining part of this section, we discuss in detail about, (i) generating schema-derived ego-network, (ii) aggregating information from this schema-derived ego-network, and (iii) aggregating information from the same-type metapath-guided neighbors using metapath-based convolutions.

A. Schema-derived ego-network

We are interested in getting a representation (and hence making predictions) for node a of type $\mathcal{A}$ in graph $\mathcal{G}$. We define the *schema-derived ego-network* (SEN) E_a of node a of graph $\mathcal{G}$ as a directed subgraph of $\mathcal{G}$, which contains all the nodes that describe a. E_a can be derived from the graph schema automatically, but can also be customized by domain experts for the task at hand. In this paper, we take a general approach, and construct E_a from the graph schema as follows: starting from a, we do a depth-first traversal of G, such that we do not visit the same edge type more than once in the same path (there is no loop). We denote the set of children (successors) of a node v as $C(v)$. Note that SEN will contain a loop if and only if, the network schema of $\mathcal{G}$ contains a loop.

B. Step 1: Aggregating information from the schema-derived ego-network (SEN)

The first step involves aggregation of information from the SEN E_a of a, as well as SENs of neighbors $N^l(a)$ of a. Lets denote the aggregated information of the SEN E_a of a as z_a.

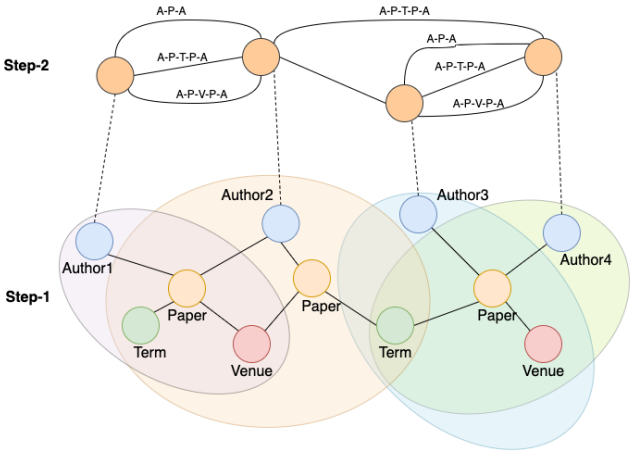
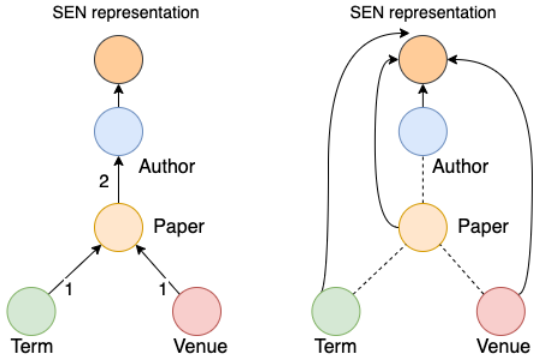


Fig. 3: Overview of the proposed framework. The key characteristic of the framework is a hierarchical structure that first computes representations of the target entity-types (Author) based on their heterogeneous networks, i.e., Step-1 and then combine them using various metapaths into a homogeneous second-level graph, i.e., Step-2.

This step can be modeled as any pooling operation over a subgraph, such as, Tree structure-aware Graph Neural Network (TGNN) [22]. For the sake of simplicity, in this paper, we explore two simple approaches for this step: *Hierarchical-Aggregation* (HA) and *Set-Aggregation* (SA). For HA, we aggregate the information of the SEN by propagating the information from the leaves towards the target node following a Tree-LSTM fashion, while for SA, we relax the hierarchy constraint, and model the node types in SEN as a set. Figure 4 illustrates the HA and SA on the scholar network with author, paper, venue and term node types. Note that, we illustrate relatively simpler aggregation strategies for this step, but we can leverage more expressive aggregation approaches here as well, that can lead to potentially better performance. We discuss these aggregation approaches in detail below:



(a) Hierarchical aggregation (HA)

(b) Set aggregation (SA)

Fig. 4: Two approaches to aggregate information from the *schema-derived ego-network* (SEN).

1) Hierarchical-Aggregation (HA): For HA, we aggregate the information of the SEN by propagating the information from the leaves towards the target node following a Tree-LSTM fashion. Particularly,

$$z_a = W_{\tau(a)}(f_a) + \sum_{v \in C(a)} W_{\phi(e(v,a))}(h_v),$$

where $W_{\tau(a)}$ is a node-type specific projection matrix, $W_{\phi(e(v,a))}$ is an edge-type specific projection matrix, and f_a are the features associated with the node a. h_v is recursively defined as follows:

$$h_v = \sigma(W_{\tau(v)}(f_v) + \sum_{\hat{v} \in C(v)} W_{\phi(e(\hat{v},v))}(h_{\hat{v}})),$$

where σ is a non-linearity such as Tanh or ReLU. Note that the HA is also related to thr hierarchical aggregation strategy leveraged in Tree structure-aware Graph Neural Network (TGNN) [22].

2) Set-Aggregation (SA): For SA, we relax the connectivity information in SEN, and estimate the SEN representation as a sum of the projected representations of the constituent nodes. Each metapath from the target node to the constituent nodes corresponds to a projection matrix. Particularly,

$$z_a = \sum_{v \in C(a)} \sum_{p \in M(a \to v)} W_{\psi(p)}(h_v),$$

where M is the set of metapath instances from a to v, such that the metapath corresponding to the metapath instance does not contain a loop, and $\psi(p)$ is the mapping fuction which maps the metapath instance p to its corresponding metapath. $W_{\psi(p)}$ is a metapath specific projection matrix. Though SA is not that expressive, as by relaxing the connectivity information, it essentially ignores the semantics that are encoded in the underlying schema, it is interesting in the view that it is invariant to schematic changes discussed before. A similar idea has also been leveraged by Network Schema Preserving Heterogeneous Information Network Embedding (NSHE) [21], which estimates node-representations that are aware of the subgraph imposed by the schema, and not just aware of the locally connected neighborhood.

C. Step 2: Aggregating information from the same-type metapath-guided neighbors

The same-type metapath-guided neighbor selection gives rise to another heterogeneous graph, which has only one node type (the target type) and the number of relations is equal to the number of metapaths used to guide neighbor selection. To model the aggregation step over this heterogeneous graph, we investigate a metapath-aware aggregation approach. Our aggregation approach has two components as follows:

1) Step 2(a): Aggregating information individually for each metapath: The second step involves aggregation of information from the representations of the same-type metapath-guided neighbors $N^l(a)$ of a. We propose a metapath-aware attention-based aggregation to aggregate the information from the metapath-neighbors. Particularly, we denote the aggregated information for a metapath m at layer l as $h^l_{m,v}$, and estimate it as

$$h^l_{m,v} = \sum_{\hat{v} \in N^l_m(v)} q^l_{m,\hat{v} \to v} h^{l-1}_{\hat{v}},$$

where $N^l_m(v)$ is the set of metapath m guided neighbors for node v for lth layer, and $q^l_{m,\hat{v} \to v}$ are the path-aware attention-weights calculated as follows:

$$q^l_{m,\hat{v} \to v} = \frac{exp(u^l_{m,\hat{v} \to v})}{\sum_{\bar{v} \in N^l_m(v)} exp(u^l_{m,\bar{v} \to v})},$$

where $u^l_{m,\hat{v} \to v}$ is further a function of source node features, destination node features, in addition to metapath-features. Specifically, given the metapath instance of the metapath m denoted by $\hat{v} \to \bar{v}_1 \to \bar{v}_2 \ldots \bar{v}_{|p|-2} \to v$, we estimate $u^l_{m,\hat{v} \to v}$ as follows:

$$u^l_{m,\hat{v} \to v} = \sigma(\hat{W}^l_{m,u}(h^{l-1}_{\hat{v}} || (||_{i=1}^{|p|-2} f_{\bar{v}_i}) || h^{l-1}_v)),$$

where $||$ is the concatenation operator and $\hat{W}^l_{m,u}$ is a projection matrix. Without loss of generality, we can assume that the size of all the features-vectors of the nodes as well as the latent representations is same and equals d. This makes the size of the projection matrix $\hat{W}^l_{m,u}$ as $(|p| * d, 1)$. Particularly, we concatenate the $l-1$ layer representations of the source and destination nodes, along with the latent representations of the intermediate nodes in the metapath-instance connecting the source and destination node, and pass it through a multilayer-perceptron with one neuron at the output layer with a non-linearity σ to get the un-normalized attention-weights. Note that, we can use more sophisticated networks to calculate the attention-weights, but use the single layer network (single projection matrix) for simplicity. Note that the representations h^0_v are the output representations from step 1, i.e., $h^0_v = z_v$.

2) Step 2(b): Aggregating information from different metapaths: We use a simple one-layer multilayer-perceptron to aggregate the information from different metapaths, followed by a non-linearity. This corresponds to projecting each aggregated metapath representation from the step 2(a) and aggregating the projected representations followed by the non-linearity. Formally, we estimate the l-layer representation of node v as

$$h_v^l = \sigma\left(\sum_{m \in T} W_m^l h_{m,v}^l\right),$$

where T is the set of metapaths used for the same-type metapath-guided neighbor selection and W_m^l is the type-specific projection matrix for lth layer and metapath m. Similar to the step 2(a), we can use more sophisticated networks for this step too, but use the single layer network (single projection matrix) for simplicity.

D. Putting it together

Given the two choices we have for aggregating information from the SEN, we propose two approaches as follows:

DHGCN-H: Use Hierarchical-Aggregation to aggregate information from the SEN, followed by metapath-aware aggregation from the same-type metapath-guided neighbors.

DHGCN-S: Use Set-Aggregation to aggregate information from the SEN, followed by metapath-aware aggregation from the same-type metapath-guided neighbors.

We call a DHGCN network as k-layered, if it aggregates information from same-type metapath-guided neighbors, that are k-traversals away. Thus, a 0-layered DHGCN only aggregates information from the SEN, and does not use any information from the same-type metapath-guided neighbors.

E. Training

Though the proposed approaches can be used for any learning task over graphs (node classification, link-prediction etc.), we present results for the node classification in this paper. As such we minimize the negative log-likelihood between the predicted probabilities and ground truth labels as follows:

$$L(\mathcal{G}, F) = -\frac{1}{N}\sum_{i}^{N} \log(F(i, y_i)), \tag{1}$$

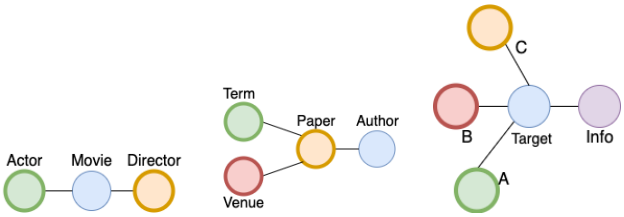
where F is the model we are training, and $F(i, y_i)$ is the probability predicted for the ground truth class y_i of the training instance i.

V. Experimental Methodology

Though the proposed framework can be used for any learning-on-graph problem, we evaluate it on node classification.

A. Datasets

We use the same two heterogeneous datasets as used in node-classification task in MAGNN [12]. Specifically, we use IMDb and DBLP datasets. In addition, we also construct a family of synthetic datasets, with controllable complexity to show the advantages of DHGCN over the competing approaches. The network schema for each of the datasets is shown in Figure 5. Table II shows the statistics of the real-world datasets. We describe each of these datasets as follows:



(a) IMDb (b) DBLP (c) Synthetic

Fig. 5: Network schema of the datasets. The blue colored circles denote the target node-type. The bold margin circles denote the nodes that appear on the metapaths between the target nodes.

IMDb: IMDb[1] is an online database about movies and television programs, including information such as cast, production crew, and plot summaries. We use the same subset of IMDb as used in [12], which contains 4278 movies, 2081 directors, and 5257 actors. The schema for the IMDb dataset is shown in Figure 5a. The Movies are labeled as one of three classes (Action, Comedy, and Drama) based on their genre information. The task is to predict the genre of a movie. For same-type metapath-guided neighbor selection, we consider the metapaths: Movie-Actor-Movie and Movie-Director-Movie.

DBLP: DBLP[2] is a computer science bibliography website. We adopt a subset of DBLP extracted by [23, 24], containing 4057 authors, 14328 papers, 7723 terms, and 20 publication venues. The schema for the DBLP dataset is shown in Figure 5b. The authors are divided into four research areas (Database, Data Mining, Artificial Intelligence, and Information Retrieval). The task here is to predict the research area of an author. Similar to IMDb, DBLP is a good dataset for the node classification task, because authors associated with a particular research area tend to publish in particular venues and the features on the paper nodes are also informative. For same-type metapath-guided neighbor selection, we consider the metapaths: Author-Paper-Author, Author-Paper-Term-Paper-Author and Author-Paper-Venue-Paper-Author.

Synthetic: We created a family of synthetic datasets to mimic the schema of real-world applications. The schema for the synthetic dataset is shown in Figure 5c. The task here is to predict the class of the node type 'target', where the total number of classes is set to three. The nodes features for the nodes of 'target' and 'info' are drawn from a multivariate Gaussian with dimensionality 50. For each class, the coefficients for this mixture follow a multinomial distribution sampled from a Dirichlet distribution. There are three more node types (A, B and C) that we use to create metapaths connecting the 'target' node types. We call them bridge node types. The number of nodes of each bridge node type is drawn from a

[1]https://www.imdb.com/

[2]https://dblp.org/

TABLE II: Dataset statistics.

Dataset	Number of node-types	No. of nodes	No. of edge-types	No. of edges	No. of labels	No. of metapaths
IMDb	3	11,616	2	17,106	3	2
DBLP	4	26,128	3	119,783	4	3

uniform distribution in the range of $[5, 10]$. Each target node is connected to exactly one of the nodes from each of the bridge node type. For each class c and each bridge node type b, we have a multinomial distribution sampled from a Dirichlet distribution. For the target nodes belonging to class c, we sample the node it connects belonging to the bridge-type b from the corresponding multinomial distribution. To predict the class of a 'target' node, we need to estimate the corresponding Gaussian distribution from which the features of the 'target' node and the connected 'info' nodes are sampled. As such, if sufficient 'info' nodes are connected to the target node, the corresponding Gaussian can be estimated. However, if the number of 'info' nodes connected to the 'target' node is not large enough, we need information from the metapath-neighbors to make accurate predictions. To this extent, we experiment with varying number of 'info' nodes connected to the 'target' node-type as presented in Section VI. For same-type metapath-guided neighbor selection, we consider the metapaths: Target-A-Target, Target-B-Target and Target-C-Target.

B. Competing approaches

RGCN: Relational Graph Convolution Network (RGCN) [8] extends the idea of GCN for heterogeneous graphs, and uses the type-specific projection matrix for message construction.
HAN: Heterogeneous Attention Network (HAN) [13] estimates metapath-specific node embeddings from different metapath-based homogeneous graphs, and leverages the attention mechanism to combine them into one vector representation for each node.
MAGNN For each user-specified meta-path, Metapath Aggregated Graph Neural Network (MAGNN) [12] first performs intra-metapath aggregation by encoding all the object features along a metapath instance, and then performs inter-metapath aggregation by attention mechanism.

C. Metrics

For the node classification task, we consider three commonly used metrics: Negative log likelihood, Micro F1 score and Macro F1 score. We discuss each of these metrics below:

F1 score: For a given instance, F1 score is the harmonic mean of the precision and recall based on the predicted classes and the observed classes. It is a popular metric used in binary classification and ranking problems. F1 score reaches its best value at 1 and worst at 0. Since, we use multiclass datasets on the node-classification task, we look into two extensions of F1 score for multiclass problems as defined below:
Micro F1 score: For the micro F1 setting, we calculate the F1 score by globally counting the total true positives, false negatives and false positives, in order to calculate precision and recall.
Macro F1 score: For the macro F1 setting, we report the mean of F1 score over all the classes.
Negative log likelihood (NLL): NLL can be interpreted as the 'unhappiness' of the network with respect to its parameters. The higher the NLL, the higher the unhappiness. It is exactly the loss function that we minimize as in Equation 1.

D. Methodology and Parameter selection

We implemented the proposed framework and the competing approaches in the efficient Deep Graph Learning (DGL)[3][25] library with PyTorch[4] backend.
Network details: We used 64 dimensional representations/embeddings for all the approaches and methods. We train the models with ADAM [26] optimizer, with the learning rate of 0.001. For regularization, we used an $L2$ regularization (weight decay) of 0.0001. In each epoch, we construct mini-batches with neighborhood sampling, and sample at maximum 20 random neighbors for each relation. The size of a mini-batch is set to 1024. We perform random walk to sample metapath-guided neighbors for each metapath. For each node and each metapath, we perform 20 random-walks to sample the metapath-guided neighbors.

For the node classification task, we report results based on a five-fold cross validation. In each fold, we apply early stopping with a patience of five, based on the performance on the NLL metric on the validation set, i.e., the validation loss. We try multiple number of layers (upto 4) for various approaches, and report results for the layer based on the performance on the NLL metric on the validation set.

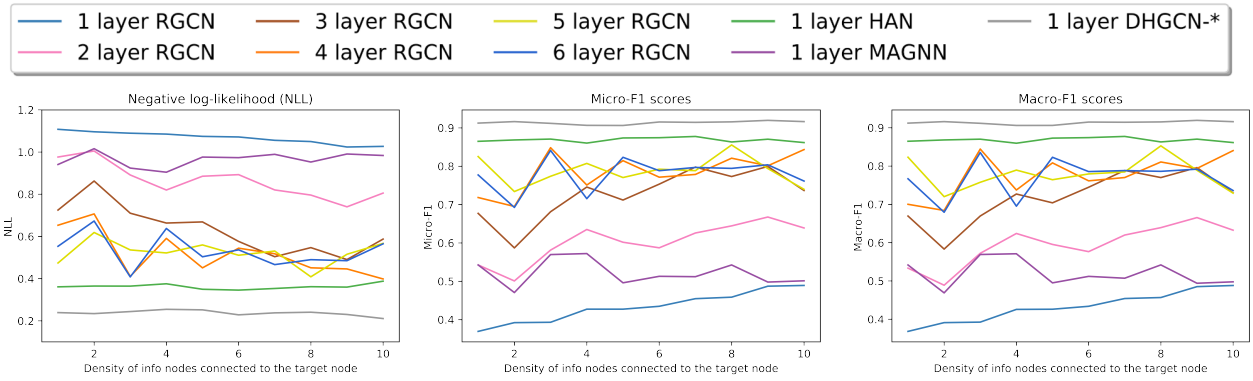
VI. Results

In this section, we present experiments to demonstrate the accuracy of our approaches for heterogeneous node classification. First, we provide our extensive analysis on the synthetic heterogeneous dataset that we created. After that, we provide and discuss the results for the node classification on the real-world datasets.

A. Results on the synthetic dataset:

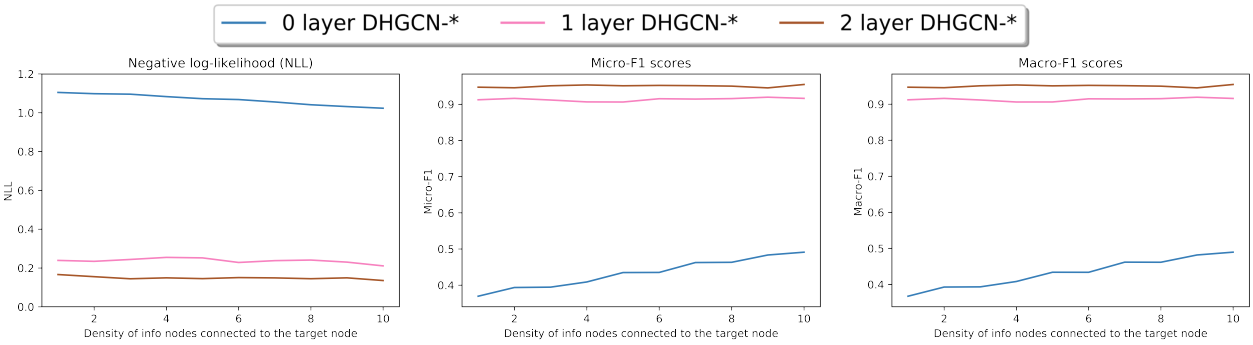
We investigate how well different approaches are able to perform when the classification requires information from multiple hops away to make accurate predictions. Specifically, we vary the number of 'info' nodes connected to the 'target' node, and see if the various approaches are able to get information from the further nodes in order to make accurate predictions. The more 'info' nodes connected to the 'target'

[3]https://www.dgl.ai/
[4]https://pytorch.org/



(b) (c) (d)

Fig. 6: Performance on the synthetic dataset.



(b) (c) (d)

Fig. 7: DHGCN performance variation with number of layers.

node, the *richer* is the immediate ego-network. Thus, for fewer connected 'info' nodes, we need information from the same-type metapath-guided neighbors to make more accurate predictions. We construct 10 datasets, such that in the ith dataset ($1 \leq i \leq 10$), for each 'target' node, we connect j 'info' nodes, where j is sampled from a Gaussian distribution with mean i (with the obvious constraint that j is an integer and $j > 0$).

Figures 6b, 6c and 6d show the NLL, Micro-F1 and Macro-F1 for various approaches on synthetic datasets with various amount of predictive signals in the immediate ego-net. Note that, both Hierarchical and Set-aggregation are equivalent for the synthetic dataset, so we only show results for the Hierarchical-aggregation based DHGCN. As expected, as the density of the 'info' nodes connected to the 'target' node increases, the performance of RGGCN and DHGCN approaches improves. By design, MAGNN and HAN do not leverage any information from the 'info' nodes, and as such, their performance does not show any trend of change as the number of 'info' nodes is varied. Further, we observe that RGCN is unable to leverage information from multiple hops away, and the performance even starts degrading as the number of layers increases due to the over-smoothing problem [9, 10, 11]. Interestingly, HAN yields a significantly better result than MAGNN. We conjecture that sequential modeling in the MAGNN results in a loss of information. In contrast, DHGCN effectively uses the information from multiple hops away and outperforms the competing approaches. We do not see any distinct patterns among the performances of the various DHGCN approaches.

Figure 7 shows the variations of DHGCN as the number of layers increases. DHGCN-* with 0 layer means we use Hiererchical aggregation to aggregate information over the SEN, but do not use any information from the same-type metapath-guided neighbors. The performance numbers increase along with the number of layers. Clearly, DHGCN based approaches are capable of leveraging from multiple hops away. Note that, for the synthetic dataset, a one-layer DHGCN actually gets information from three hops away from the underlying graph. A two layer DHGCN is able to get information from five hops away.

B. Results on the node-classification

Table III shows the performance of different methods on the IMDb dataset. All the DHGCN approaches outperform the competing approaches achieving up to 0.3%, 1.0% and 1.2% improvement over the next best approach, which is RGCN for all three metrics. Note that both Hierarchical-Aggregation and Set-Aggregation are equivalent in the case of IMDb, as there

TABLE III: Results on the node classification task: IMDb.

Method	NLL	Macro F1	Micro F1
RGCN	0.731 ± 0.013	0.690 ± 0.009	0.689 ± 0.010
MAGNN	0.747 ± 0.016	0.680 ± 0.005	0.679 ± 0.006
HAN	0.809 ± 0.006	0.648 ± 0.007	0.649 ± 0.007
DHGCN-S	$\mathbf{0.729 \pm 0.014}$	$\mathbf{0.697 \pm 0.005}$	$\mathbf{0.697 \pm 0.006}$
DHGCN-H	$\mathbf{0.729 \pm 0.014}$	$\mathbf{0.697 \pm 0.005}$	$\mathbf{0.697 \pm 0.006}$

Boldfaced entries correspond to the overall best-performing scheme. Underlined entries correspond to DHGCN variants whose performance is better than all previously developed methods (RGCN, MAGNN, and HAN).

TABLE IV: Results on the node classification task: DBLP.

Metric	NLL	Macro F1	Micro F1
RGCN	0.173 ± 0.018	0.938 ± 0.007	0.942 ± 0.006
MAGNN	0.203 ± 0.031	0.930 ± 0.012	0.936 ± 0.010
HAN	0.260 ± 0.023	0.918 ± 0.014	0.924 ± 0.014
DHGCN-S	0.165 ± 0.019	0.938 ± 0.006	0.942 ± 0.004
DHGCN-H	$\mathbf{0.155 \pm 0.009}$	$\mathbf{0.944 \pm 0.005}$	$\mathbf{0.948 \pm 0.005}$

Boldfaced entries correspond to the overall best-performing scheme. Underlined entries correspond to DHGCN variants whose performance is better than all previously developed methods (RGCN, MAGNN, and HAN).

is only one level of information in the IMDb schema. Further, RGCN outperforms HAN and MAGNN.

Table IV shows the performance of the methods on the DBLP dataset. In the same fashion, DHGCN approaches, in general, outperform all of the competing approaches on the DBLP dataset on the NLL metric, achieving up to 10.4% improvement over the next best performing baseline (RGCN). On the Macro F1 and Micro F1, though DHGCN-H outperforms the competing baselines, while DHGCN-S performs at par with RGCN. Specifically, DHGCN approaches outperform RGCN by achieving up to 0.6% on both the Macro F1 and Micro F1 metrics. Similar to IMDb, RGCN and DHGCN approaches outperform HAN and MAGNN.

VII. Conclusion

In this paper, we proposed solutions to two major limitations of GCN approaches for heterogeneous graphs: (i) over-smoothing, leading to low expressive power as a result of shallow models; or (ii) inability to explore the complete ego-network, leading to missing out some nodes, which might provide important signals for the task at hand. To this extent, our solution *Deep Heterogeneous Graph Convolutional Network* (DHGCN), is a purposefully designed class of GCN models for heterogeneous graphs. DHGCN uses a two-step schema-aware hierarchical approach. In the first step, DHGCN aggregates information from the immediate ego-network that is derived from the schema. In the second step, DHGCN aggregates information from the same-type metapath-guided neighbors. On a variety of synthetic and real-world datasets, we show that the proposed approaches outperform the competing baselines.

There are many avenues of optimization of our approaches worth investigating, for example, using stronger models to estimate the attention weights for Step-2 aggregation, such as deeper neural networks, using multi-head attention etc. It is also important to perform further benchmarking on heterogeneous datasets generated from more complex schemes. This constitutes our future work. Code accompanying with this paper is available at Github[5].

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[3] J. Zhang, X. Shi, S. Zhao, and I. King, "Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems," *arXiv preprint arXiv:1905.13129*, 2019.

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

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[6] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "Gaan: Gated attention networks for learning on large and spatiotemporal graphs," *arXiv preprint arXiv:1803.07294*, 2018.

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *6th International Conference on Learning Representations*,

[5]https://github.com/gurdaspuriya/dhgcn

2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[8] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[9] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] S. Luan, M. Zhao, X.-W. Chang, and D. Precup, "Break the ceiling: Stronger multi-scale deep graph convolutional networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 943–10 953.

[11] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *arXiv preprint arXiv:1902.07153*, 2019.

[12] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," in *Proceedings of The Web Conference 2020*, 2020, pp. 2331–2341.

[13] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference*, 2019, pp. 2022–2032.

[14] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," 2017.

[15] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk," *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, 2014. [Online]. Available: http://dx.doi.org/10.1145/2623330.2623732

[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," 2016.

[17] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line," *Proceedings of the 24th International Conference on World Wide Web - WWW '15*, 2015. [Online]. Available: http://dx.doi.org/10.1145/2736277.2741093

[18] Y. Dong, N. V. Chawla, and A. Swami, "Metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[19] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," 2020.

[20] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," 2019.

[21] J. Zhao, X. Wang, C. Shi, Z. Liu, and Y. Ye, "Network schema preserving heterogeneous information network embedding." IJCAI, 2020.

[22] Z. Qiao, P. Wang, Y. Fu, Y. Du, P. Wang, and Y. Zhou, "Tree structure-aware graph representation learning via integrated hierarchical aggregation and relational metric learning," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 432–441.

[23] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao, "Graph regularized transductive classification on heterogeneous information networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 570–586.

[24] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han, "Graph-based consensus maximization among multiple supervised and unsupervised models," in *Advances in Neural Information Processing Systems*, 2009, pp. 585–593.

[25] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

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