Learning and Understanding User Interface Semantics from Heterogeneous Networks with Multimodal and Positional Attributes

12

GARY ANG and EE-PENG LIM, Singapore Management University

User interfaces (UI) of desktop, web, and mobile applications involve a hierarchy of objects (e.g., applications, screens, view class, and other types of design objects) with multimodal (e.g., textual and visual) and positional (e.g., spatial location, sequence order, and hierarchy level) attributes. We can therefore represent a set of application UIs as a heterogeneous network with multimodal and positional attributes. Such a network not only represents how users understand the visual layout of UIs but also influences how users would interact with applications through these UIs. To model the UI semantics well for different UI annotation, search, and evaluation tasks, this article proposes the novel Heterogeneous Attention-based Multimodal Positional (HAMP) graph neural network model. HAMP combines graph neural networks with the scaled dot-product attention used in transformers to learn the embeddings of heterogeneous nodes and associated multimodal and positional attributes in a unified manner. HAMP is evaluated with classification and regression tasks conducted on three distinct real-world datasets. Our experiments demonstrate that HAMP significantly outperforms other state-of-the-art models on such tasks. To further provide interpretations of the contribution of heterogeneous network information for understanding the relationships between the UI structure and prediction tasks, we propose Adaptive HAMP (AHAMP), which adaptively learns the importance of different edges linking different UI objects. Our experiments demonstrate AHAMP's superior performance over HAMP on a number of tasks, and its ability to provide interpretations of the contribution of multimodal and positional attributes, as well as heterogeneous network information to different tasks.

CCS Concepts: • **Computing methodologies** → **Neural networks** • **Human-centered computing** → **User interface management systems** • **Computing methodologies** → **Artificial intelligence** • **Information systems** → **Multimedia information systems**;

Additional Key Words and Phrases: Graph neural networks, transformers, attention mechanism, heterogeneous networks, multimodal, mobile application user interface, supervised learning

The reviewing of this article was managed by special issue associate editors Simone Stumpf, Tuukka Ruotsalo, Krzysztof Gajos.

This research is supported by the National Research Foundation, Singapore under its Strategic Capabilities Research Centres Funding Initiative. Gary Ang is supported by a Monetary Authority of Singapore Postgraduate Scholarship. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore, nor the Monetary Authority of Singapore.

Authors' address: G. Ang (corresponding author) and E.-P. Lim, Singapore Management University, 80, Stamford Road, Singapore 178902, Singapore; emails: gary.ang.2019@phdcs.smu.edu.sg, eplim@smu.edu.sg.

2160-6455/2023/09-ART12 $15.00
https://doi.org/10.1145/3578522

ACM Reference format:
Gary Ang and Ee-Peng Lim. 2023. Learning and Understanding User Interface Semantics from Heterogeneous Networks with Multimodal and Positional Attributes. *ACM Trans. Interact. Intell. Syst.* 13, 3, Article 12 (September 2023), 31 pages.
https://doi.org/10.1145/3578522

1 INTRODUCTION

The pervasiveness of mobile applications and availability of mobile application **user interface** (**UI**) repositories containing rich multimodal information have made the mining of mobile application design knowledge [34] an important research topic which benefits retrieval and annotation of UI objects, design of **user interactions/experiences** (**UI/UX**), and evaluation of UI designs. Mobile application UI data can be viewed as a hierarchy of design objects—mobile applications associated with multiple UI screens, code classes (e.g., Java code classes from the Android API), and elements. As these different types of design objects are linked with one another, they form a *heterogeneous network*. Moreover, they are often associated with *multimodal* and *positional* attributes. Examples of multimodal attributes include visual information (e.g., UI screen and element images) and textual information (e.g., code and description of design objects). The positional attributes include spatial locations of design objects (e.g., locations of UI elements in UI screens), sequential positions (e.g., order of UI screens during user interactions), and hierarchical positions (e.g., hierarchical levels of design objects). Heterogeneous networks formed from a hierarchy of design objects with multimodal and positional attributes are also common in many other UI-related applications—web, print, tangible.

Psychological evidence shows that humans parse images into part-whole hierarchies and model the viewpoint-invariant spatial relationships between a part and a whole as they process a piece of visual information [21]. Intuitively, such hierarchical representations are even more important for UIs since the part-whole hierarchies not only allow a user to understand the visual layout of UIs but also influence user interactions and experiences. For example, the number of levels in a UI design hierarchy could influence the navigation experience of the user; the spatial, sequential, and hierarchical positions of a UI object could affect the way the user perceives its functional role and importance. The above intuition thus motivates our research objective to design a model for heterogeneous networks with multimodal and positional attributes to capture the semantics of UI objects. Such a model would enable a semantic representation vector to be learned for every UI object which can be used in downstream tasks such as UI search, evaluation, annotation, and organization.

In addition to the above modeling requirement, it is also important to be able to interpret the contribution of modeled information to the different downstream tasks. This allows end-users to understand the underlying relationships between the design of the UI structure and model predictions. With this understanding, end users of downstream tasks can gain greater confidence and trust in the model. UI designers and developers can also understand which parts of the UI network structure to focus on to improve the overall UI design.

In this article, we use the RICO repository [9, 34] and an enhanced version of RICO, ENRICO [29], as exemplar datasets. RICO is a real-world mobile UI dataset that covers more than 9,000 Android applications and their rich design information. RICO can be used to support novel UI/UX applications, such as automatic categorization and annotation of UIs for designers to learn design patterns and trends; and prediction of user ratings of design objects to help designers evaluate new designs. ENRICO is an enhanced subset of the RICO dataset that includes 1,460 UI screens manually annotated with design topics, e.g., a UI screen with a *dialer*, *tutorial*, or *news* topic. The

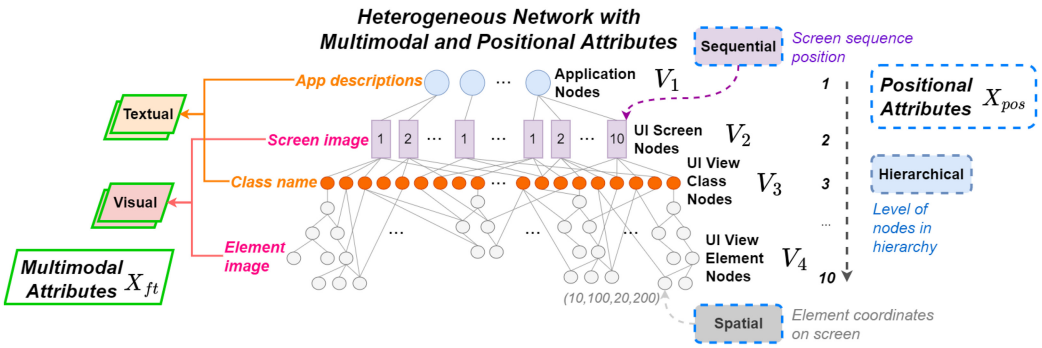

Fig. 1. We show how a hierarchy of UI objects and their multimodal and positional attributes can be captured in an attributed heterogeneous network. The heterogeneous network comprises four types of nodes—mobile application nodes, UI screen nodes, UI view class nodes, and UI view element nodes, with attributes from different modalities, as well as different positional attributes. On the left side, we show application and UI view class nodes associated with textual application description and UI view class name attributes; and UI screen and UI view element nodes associated with visual screen image and element image attributes. On the right side, we show the different types of positional attributes. Each node-type may be associated with one or more of these positional attributes.

ENRICO dataset is intended to be utilized for a range of UI design applications: semantic UI captioning, automatic UI tagging and annotation, explainable UI designs, and search and retrieval.

Figure 1 depicts mobile application UIs in RICO and ENRICO as a heterogeneous network with multimodal and positional attributes. There are four types of nodes in this heterogeneous network—mobile applications, UI screens, UI view classes, and UI elements. Multimodal attributes are associated with these four types of nodes—mobile application node has a description attribute, both UI screen node and UI element node have image attributes, and UI view class object has class name attribute. The positional attributes include spatial locations of UI elements on the screen, sequential positions of UI screens in user interaction traces, and the hierarchical level of the nodes in the original design hierarchy. The hierarchy of nodes in this network structure is captured by the hierarchical level attribute.

Figure 2 provides an overview of the proposed framework in this article, and examples of predictive application tasks that may be performed using a heterogeneous attributed network model, namely to (i) *predict element annotation*: automatic annotation of elements' component type, which could be used as an accessibility metadata attribute in mobile applications; (ii) *predict genre category*: automatic classification of UIs' genres, which could be used for organization, search and retrieval of a UI repository; (iii) *predict UI rating*: which could be used by UI designers to perform an initial evaluation of their designs; and (iv) *predict UI topic*: automatic tagging of UIs with topics for semantic UI captioning, UI tagging, and annotation [29].

Despite the large volume of multimedia research, there is very little work on heterogeneous networks with multimodal and positional attributes, particularly the non-Euclidean nature of network structures associated with multimodal attributes. In recent years, **graph neural network** (**GNN**) models have been developed to capture both network structures and node attributes. However, most GNN works do not attempt to capture multiple modalities of node attributes in heterogeneous networks. They also do not cover spatial and sequential node positional information.

In previous works that capture UI multimodal information [16, 30], item embeddings are generated for recommendation tasks from user interaction information and metadata text without capturing multimodal and positional attributes of a heterogeneous network of UI design objects. Ang and Lim [2] propose multimodal GNN to model mobile UI-related tasks but the model does

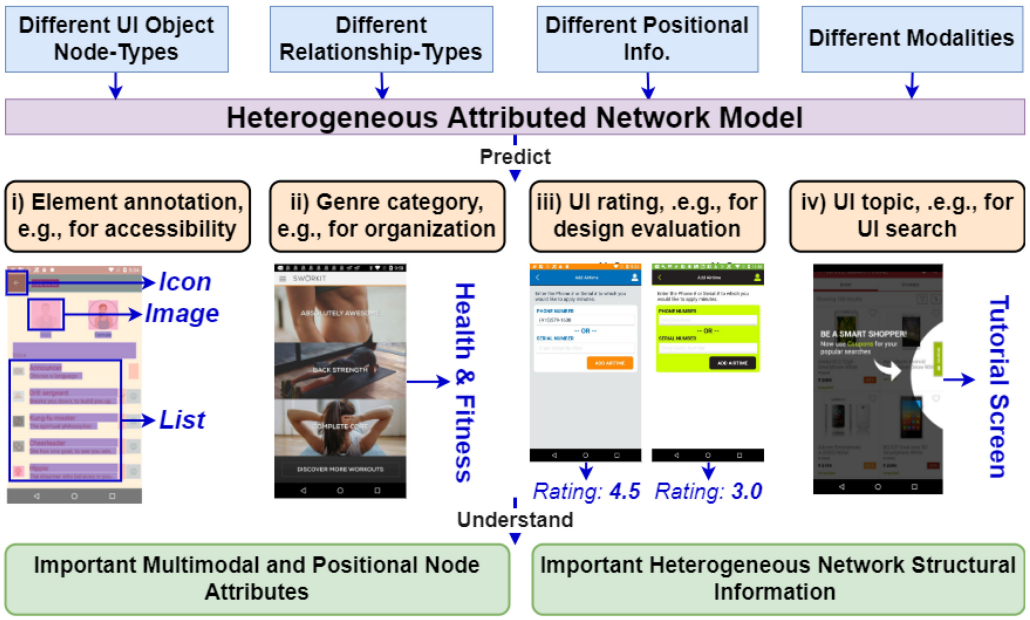

Fig. 2. Overview of framework in this article. A heterogeneous attributed network model captures different UI object node-types with different inter-UI object relationships and multimodal and positional attributes for four different predictive tasks. The heterogeneous attributed network model also allows interpretability of the contributions of multimodal and positional node attributes, as well as network structural information to these predictive tasks, enabling users to better understand model predictions.

not incorporate spatial, sequential, and hierarchical information, and is limited to bipartite networks, a special type of heterogeneous network. Screen2Vec [30] is another model that captures multimodal and sequential UI information, but not the structural network and positional information. The above works also do not interpret the contributions of network structural information to the different tasks.

Hence, in this article, we propose two models that address the limitations of existing models: the **Heterogeneous Attention-based Multimodal Positional** (**HAMP**) model and **Adaptive HAMP** (**AHAMP**) model. Specifically, HAMP aims at: (1) capturing information from different modalities for different types of nodes at different levels of a design hierarchy, along with other positional information such as spatial location and sequence order; (2) ensuring that low dimensional positional attributes are captured effectively alongside high dimensional multimodal attributes; and (3) self-discovering the relative importance of multimodal attributes and positional attributes. In the AHAMP model, inspired by GNNExplainer [55], we propose a novel UI graph discovery and interpretability method to interpret the contribution of the network structural information to different prediction tasks. GNNExplainer [55] introduces a post-hoc method designed for homogeneous networks that provide explanations for selected nodes (i.e., a local explanation). The work, however, is unable to directly provide an explanation for a set of selected task instances, e.g., multiple instances for a UI genre classification task. It instead proposes a graph prototype-based approach to aggregate local explanations across selected task instances.

Our proposed approach in AHAMP differs from GNNExplainer [55] as it: (i) addresses heterogeneous networks; (ii) is integrated within the model; and (iii) directly provides an explanation across multiple instances for a task. AHAMP is able to discover important heterogeneous network

edges between UI objects and interpret the contribution of such heterogeneous network structural information to different prediction tasks across multiple instances.

Both HAMP and AHAMP are based on a novel attention-based GNN inspired by transformers.[1] Our key contributions are as follows:

- To our knowledge, this is the first work to propose an interpretable approach that captures heterogeneous networks and their associated multimodal and positional attributes in a unified manner for UI-related tasks;
- HAMP is also the first work that incorporates a positional vectorizer (PosVect) with different functional forms to extract important information from different positional (spatial, sequential and hierarchical level) attributes within a GNN framework. The module also ensures proper capture of lower dimensional spatial, sequential, and hierarchical level attributes by expanding their dimensions to match higher dimensional multimodal attributes;
- We propose the use of attention fusion in HAMP to self-discover the relative importance of multimodal and spatial, sequential, and hierarchical level attributes for different nodes, and enable the relative importance of the different attributes to be extracted for interpretability;
- We combine the proposed positional vectorizer and attention fusion modules with a scaled dot-product attention-based GNN message propagation method that is designed for heterogeneous networks with different node and edge-types;
- We further propose AHAMP, an extension of HAMP, which includes a novel adaptive graph discovery and interpretability method for heterogeneous network settings to discover important network edges between UI objects for different prediction tasks. AHAMP also supports interpretation of the contribution of heterogeneous network structural information to different tasks.
- We show that HAMP and AHAMP consistently out-perform several state-of-the-art models on UI screen genre classification, UI element component type classification, mobile application ratings prediction, and UI screen topic classification tasks which are highly relevant to real-world applications.

2 RELATED WORK

Key related works in the areas of UI representational learning and network embeddings are outlined in this section.

2.1 UI Representation Learning

[4, 7, 23, 41, 52] are examples of recent representation learning work that also capture UI semantics for a range of tasks, but they use information from just one or two modalities, and do not utilize the structural network information present in the linkages between different UI objects. Huang et al., [23] retrieve UI screen images based on UI sketch images by using image embeddings for visual similarity comparisons but do not capture the structural network information present in the linkages between UI objects. Xie et al., [52] capture structural network information of UI objects to support retrieval applications but do not capture multimodal attributes. Ang and Lim [2] capture structural network and multimodal information but do not incorporate spatial, sequential, and hierarchical information, and is designed for a bipartite network, a special kind of heterogeneous network. Screen2Vec [30], a recent work, generates representations of UI screens and components that capture the multimodal information of UIs, UI layouts, and sequential information of UIs in user interactions. However, it does not capture the structural network information present in a heterogeneous network of UI objects.

[1]Source code available at: https://github.com/playgrdstar/AHAMP.

2.2 Network Embeddings

There are several related works on network embedding approaches which could be applied to a network of UI design objects. The **Graph Variational Autoencoder** (**GVAE**) [26] approach applies a variational autoencoder [25] framework to learn the node embeddings of homogeneous networks. The **Co-Embedding Attributed Network** (**CAN**) [36] uses two VAE channels to jointly encode and decode the node adjacency matrix and another node feature matrix. Semi-supervised Co-embedding Attributed Network [37] extends CAN to co-embed both attributes and nodes of partially labeled networks. Multinomial VAE [32] is a VAE-based approach that generates embeddings of heterogeneous networks by using a multinomial distribution instead of the Bernoulli distribution used in GVAE.

GNN is another approach which composes messages based on network features and propagates them to update the representation vectors of nodes and/or edges over multiple neural network layers [3, 13]. Several GNN-based models have been developed. In particular, **Graph Convolutional Network** (**GCN**) [27] aggregates features of neighboring nodes and normalizes the aggregated representations by the node degrees. GraphSAGE [18] further considers mean, LSTM, or pooling aggregation methods. Unlike GCN, GraphSAGE samples only a fixed number of neighbors for representation aggregation. **Graph Attention Network** (**GAT**) [46] assigns neighboring nodes with different importance weights during aggregation using additive attention. Messages passed between each layer in most GNNs go through non-linear layers such as rectified linear activation units. **Simplifying Graph Convolutional** (**SGC**) Network [51] adopts linear layers to process messages as they are passed to neighboring nodes. **Hard Graph Attention Operator** (**hGAO**) [12] applies hard attention, requiring each node to only attend to a subset of neighboring nodes to improve performance and reduce computational costs.

GNNs have also been applied to heterogeneous networks. Relational Graph Convolutional Networks [42] and Graph Convolutional Matrix Completion [44] use multiple GCNs to encode embeddings of multiple adjacency matrices, one for each edge type, before aggregating them. Neural Graph Collaborative Filtering [48] and LightGCN [19] encode embeddings for different numbers of hops before aggregating them. BiANE [24] captures indirect proximity between the same node types in bipartite networks. **Heterogeneous Graph Attention Network** (**HAN**) [49] and General Attributed Multiplex Heterogeneous Network [5] use multiple GNN-based layers to encode networks formed from different metapaths [10] before using an attention mechanism to aggregate the embeddings.

Other than VAE and GNN-based approaches, transformers [45], initially designed for modeling sequential information, have also been generalized for networks/graphs [11]. While these graph transformers are not designed to capture multimodal, spatial, sequential, and hierarchical information, we are inspired by the scaled dot-product attention mechanisms in such graph transformer works [22, 54].

There are very few works on modeling networks with multiple modalities as well as spatial, sequential, or hierarchical information. With regards to sequential information, there are GNN-related works [15, 17, 33, 40] designed for sequences of network snapshots with their associated timestamps. However, such works are designed for snapshots of simple networks with overlapping sets of nodes and edges, and are not suitable for the hierarchical network described in this article. This article thus proposes the use of a module to accommodate different functional forms (the positional vectorizer) when extracting important lower dimensional spatial, sequential and hierarchical features (representing linear and non-linear patterns) and expanding their dimensions to match higher dimensional textual and visual information. This modeling approach has thus far not featured in GNN-related works. We also use an attention mechanism to self-discover the

Table 1. Summary of Key Notations

Symbol	Description
V	Nodes in graph G comprising Q disjoint sets of nodes of different node-types
E	Edges in graph G formed based on R relationship-types
$X_m; m \in \{ft, pos\}$	Node features, comprising multimodal features ft; as well as positional features pos. Positional features may include spatial locations sp; sequential positions sq; and hierarchical levels hi
$N(v)$	Neighboring nodes of node $v \in V$
X'	Hidden representations after fusion of multimodal, spatial, sequential, hierarchical level information
$e = \langle v_s, r, v_t \rangle$	Canonical triplet formed based on edge e consisting of relationship r between a source node v_s of one node-type and a target node v_t of the same or different node-type
H	Node embeddings/representations

relative importance of multimodal and spatial, sequence and hierarchical level information for different nodes. Finally, instead of the usual message passing methods employed in most GNNs, we adapt the scaled dot-product attention mechanism inspired by graph transformers to undertake the composition, aggregation and update steps in a GNN message passing framework.

Recently, new approaches to explain and interpret the predictions of GNNs have also been proposed and they include GNNExplainer [55], PGExplainer [35], and PGMExplainer [47], which focus on providing single instance or multi-instance explanations for homogeneous networks. A detailed review of such works can be found in Hao et al., [57] and Li et al., [31]. Such works are however neither designed for global explanations across multiple instances for heterogeneous networks nor for the interpretation of GNN predictions for UI-related tasks.

3 HETEROGENEOUS ATTENTION-BASED MULTIMODAL POSITIONAL GRAPH NEURAL NETWORK

In this section, we give a detailed description of our two proposed models, the HAMP model and AHAMP model. We will first cover HAMP as it forms the base model for AHAMP. A summary of the key notations is provided in Table 1.

In both models, we represent different types of objects (e.g., mobile application, UI screen, UI view class, and UI view element nodes in the case of the RICO dataset) as nodes at different levels in a heterogeneous network. We denote the network as

$$G = (V, E, X), \tag{1}$$

where V includes Q disjoint sets of nodes of different types $V = V_1 \cup \ldots \cup V_Q$. Similarly, E consists of edges of R types, i.e., $E = E_1 \cup \ldots \cup E_R$. In the case of the RICO dataset, each edge type represents a specific part-whole relationship as shown in Figure 1, specifically, UI view element nodes *instantiated by* UI view class nodes, UI view class nodes that are *part of* UI screen nodes, and UI screen nodes that are *part of* application nodes. Every edge connecting two nodes is then represented as a *canonical triplet* $\langle v_s, r, v_t \rangle \in E$, where $v_s, v_t \in V$ and $r \in \{1, \ldots, R\}$.

HAMP is designed to model multimodal attributes which can be textual, visual, categorical or numerical, and positional attributes which can be spatial locations, sequential positions, and hierarchical level numbers. For each node-type q, we define a matrix to represent the values of each attribute associated with nodes of the type q. For each textual, visual, categorical, or numerical attribute, we use a X^q_{ft} matrix to represent the node to attribute value mapping, where ft refers to

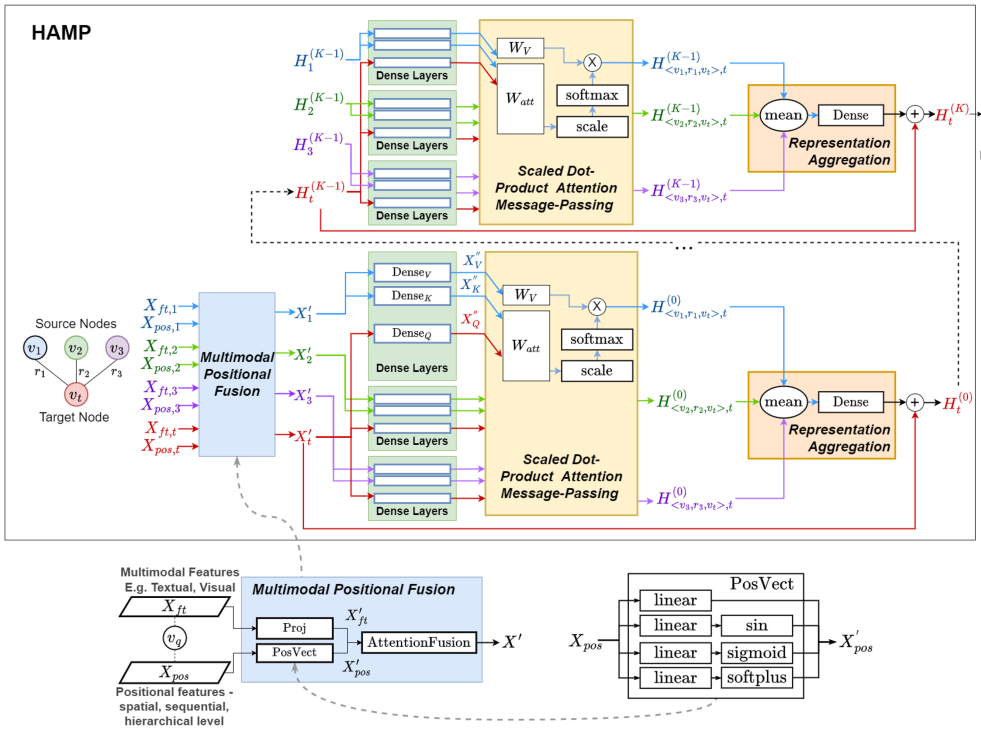

Fig. 3. Architecture of HAMP model, which comprises three main components: positional vectorizer (PosVect) that captures linear, non-linear and periodic relationships between positional attributes and different tasks; multimodal positional fusion module that projects multimodal and positional features to a common latent representational space and fuses them with attention mechanisms; and network-encoding layers that capture structural heterogeneous network information. A different module, say a dense neural network layer, is added to HAMP at the end for different tasks, e.g., classification.

the non-positional attributes. X^q_{ft} is of $|V_q| \times N^q_{ft}$ dimension where N^q_{ft} is the dimension size of the multimodal attribute. For each positional attribute, we also use a X^q_{pos} matrix to represent the node to position mapping. The dimension size of X^q_{pos} is $|V_q| \times N^q_{pos}$ where N^q_{pos} is the dimension of the positional attribute. In the case of specific nodes that are missing multimodal attribute values (e.g., missing textual descriptions), we set the missing multimodal attribute values to random values. We set missing attribute values to random values that are different across such nodes instead of ones or zeros so as to prevent the model from considering nodes with missing attribute values to be similar.

As shown in Figure 3, HAMP comprises three key components. First, the *positional vectorizer* (PosVect) captures linear, non-linear, and periodic relationships between positional attributes and different tasks. Second, the *multimodal positional fusion* module projects different multimodal and positional features to a common latent representational space and fuses them with attention mechanisms. Finally, the *attention-based network-encoding layers* comprising *scaled dot-product attention message-passing* and *representation aggregation* steps are used to capture structural heterogeneous network information.

To learn the representation of each target node v_t, HAMP first extracts edges linking other neighboring source nodes to v_t as canonical triplets $\langle v_s, r, v_t \rangle$'s from the heterogeneous network.

Suppose the multimodal and positional node attribute values are represented in the X^q_{ft} and X^q_{pos} matrices, respectively. For both simplicity and without loss of generality, we shall drop node type q from the following description. The *multimodal positional fusion* module first expands the positional attribute vectors with the PosVect to match the dimensions of other multimodal information before the positional and multimodal attribute vectors are fused together with an attention mechanism. For each triplet (or edge) $e_i = \langle v_q, r, v_t \rangle$'s and target node v_t pair, a *scaled dot-product attention message-passing* mechanism is then used to learn the triplet-specific embedding of v_t, denoted by $H_{e_i,t}$. Once we obtain all the triplet embeddings for the target node v_t, for $\{\langle v_s, r, v_t \rangle \in E\}$, the *representation aggregation* step averages across all these embeddings and passes the resultant embedding through a dense layer to obtain v_t's final representation denoted by H_t. We shall elaborate on these steps, as shown in Figure 3, in the subsequent sections.

3.1 Multimodal Positional Fusion

We first project each multimodal attribute vector (with different dimensions) to a common dimension with a dense layer - $X'_{ft} = \text{Proj}(X_{ft})$. We use a positional vectorizer, PosVect, to expand positional attribute vectors, where present, to match the dimensions of the projected multimodal attribute vector, i.e., $X'_{pos} = \text{PosVect}(X_{pos})$. That is, X'_{ft} and X'_{pos} share the same dimension size. The positional vectorizer, inspired by [14, 39], generates higher dimensional representations from the low dimensional positional attributes and extracts important longitudinal information (i.e., relationships between the task and the positional attributes) via different pre-defined functional forms (that can be linear, non-linear or periodic). The set of functional forms chosen should allow for different types of patterns, e.g., linear, non-linear, and/or periodic, representing different relationships between the task and the positional attributes to be captured. Unlike the sinusoidal positional encodings used in transformers that only deal with the sequential positions of word tokens [45], the positional vectorizer is of a more general form to enable it to be applied to different types of positional attributes. For our experiments with HAMP and AHAMP on the RICO and ENRICO dataset, we empirically chose Linear, Sinusoidal, Sigmoid, and Softplus functions to capture linear, non-linear, and periodic patterns for the positional vectorizer for all positional attributes.

Next, we use an attention mechanism to fuse these intermediate representations X'_{ft} and X'_{pos} into X'. The use of an attention mechanism for multimodal positional fusion allows the model to self-discover the relative importance of each information type, and weight contributions accordingly for the task at hand. It also allows us to interpret the relative importance of the different inputs. We now explain the key steps involved in the attention mechanism for multimodal positional fusion in detail. We first apply a non-linear transformation to each of these intermediate representations to obtain the scalars K_m for attribute m where $m \in \{ft, pos\}$.

$$K_m = W^{(1)} \tanh\left(W^{(0)} X'_m + b\right), \tag{2}$$

where $W^{(0)}$ and $W^{(1)}$ are learnable weight matrices and b is the bias vector. These three parameters are shared across multimodal and positional attributes. We then normalize K_m with a softmax function to obtain the attention weights for the respective multimodal and positional attributes:

$$\beta_m = \frac{\exp(K_m)}{\sum_m \exp(K_m)}. \tag{3}$$

Finally, we use these weights to fuse the multimodal and relevant positional representations (i.e., spatial, sequential, and/or hierarchical level representations depending on the node-type) of each

of the node-types and apply a dense layer to obtain the fused representation X'':

$$X'' = \text{Dense}\left(\sum_m \beta_m X'_m\right), \tag{4}$$

where the Dense function is a simple fully-connected linear layer. A bias vector and a non-linear activation layer could also be added where necessary.

The steps outlined above, i.e., the attention fusion of the multimodal and positional attributes, and application of the dense layer to the fused representations, are repeated to generate the query, key and value representations, resulting in X''_Q, X''_K, X''_V for each node. Hence, along with the query, key and value representations, we also introduce the corresponding $W_K^{(0)}$, $W_Q^{(0)}$, $W_V^{(0)}$, $W_K^{(1)}$, $W_Q^{(1)}$, $W_V^{(1)}$, b_K, b_Q, and b_V. Parameters for the dense layer used for projection of the multimodal attribute vectors (Proj), the positional vectorizer (PosVect), attention mechanism and the final dense layer (Dense) as described above are shared between nodes of each node-type.

3.2 Scaled Dot-product Attention Message Passing

In this step, inspired by Hu et al., [22] and Yao et al., [54], we adapt the scaled dot-product attention module commonly used in transformers [45] for the GNN message-passing framework. The scaled dot-product attention mechanism used in transformers for natural language processing usually computes an attention score between every pair of word tokens in a sentence, whereas the scaled dot-product attention mechanism applied to networks is more efficient as it utilizes network information to only compute attention scores between nodes that are neighbors. These attention scores are used to weight the messages propagated from the source to target nodes for aggregation. Using scaled dot-product attention is also more effective than the usual message-passing framework employed in most GNNs as it allows the model to perform the message composition, propagation, and update steps in the GNN message-passing framework based on the self-discovered relative importance of each neighboring source node.

For each canonical triplet, we obtain the node embedding by first computing the attention score AttScore between a target node v_t and each neighboring source node $v_s \in N(v_t)$ as

$$\text{AttScore}_{\langle v_s, r, v_t\rangle} = \text{softmax}_{v_s \in N(v_t)} \text{scale}\left(X''_{K,v_s} W_{att} X''_{Q,v_t}\right), \tag{5}$$

where $N(v_t)$ denotes the neighboring nodes of v_t. W_{att} is a learnable weight matrix, and the scale operation divides the resultant values by the square root of the dimension of the hidden representation per [45].

Next, we use the attention score AttScore to compute the weighted average of features from all source nodes and use it to update the triplet-specific representation of the target node v_t.

$$H_{\langle v_s, r, v_t\rangle, t} = \sum_{v_s \in N(v_t)} \text{AttScore}_{\langle v_s, r, v_t\rangle} \cdot X''_{V,v_s} W_V, \tag{6}$$

where W_V is a learnable weight matrix. Multiple heads can also be incorporated within each scaled dot-product attention module per [45].

3.3 Representation Aggregation

At this point, we have the embeddings of the target node v_t for each of the canonical triplets or edges connected to neighboring nodes $N(v_t)$. We assume each of these canonical triplet-specific embeddings is equally important, and hence perform the final aggregation step by averaging these

embeddings to obtain a single representation $H_t'^{(0)}$. That is,

$$H_t'^{(0)} = \frac{1}{|N(v_t)|} \sum_{v_s \in N(v_t)} H_{\langle v_s, r, v_t \rangle, t}. \tag{7}$$

Besides the above approach, we can consider other aggregation approaches (e.g., using an attention mechanism to weight the different canonical triplet-specific embeddings) but we would include this as part of future work. Finally, a dense layer and a residual connection are applied to obtain the representation of the node $H_t^{(0)} = \text{Dense}(H_t'^{(0)}) + X_t'$, which can then serve as input to another layer comprising the dense layers, scaled dot-product attention message-passing and representation aggregation modules.

3.4 Multiple Message Passing and Aggregation Layers

As shown in Figure 3, the node representations from a prior layer (i.e., *k-1* layer) are passed into dense layers to generate query, key, and value representations as inputs to the scaled dot-product attention messaging passing module. The scaled dot-product attention messaging passing module then generates a representation for each target node for each canonical triplet, i.e., $H^{(k)}_{\langle v_s, r, v_t \rangle, t}$. We then aggregate these target node representations across the canonical triplets $\langle v_s, r, v_t \rangle \in E$ (as described earlier) to obtain $H_t'^{(k)}$. A residual connection is then applied to obtain $H_t^{(k)} = \text{Dense}(H_t'^{(k)}) + H_t^{(k-1)}$.

$H_t^{(k)}$ can then be passed to a task-specific module. If K layers are used, then every node embedding contains information about its K-hop neighborhood. The use of residual connections serves to address potential over-smoothing, which can arise when we have multiple rounds of GNN message-passing over multiple layers [53]. Over-smoothing causes representations for all nodes in a network to become very similar to one another and can lead to poorer performance. The use of residual connections allows information from earlier GNN message-passing layers to flow to the final layer, helping to alleviate the over-smoothing issue.

The HAMP model is optimized with gradient descent with an appropriate loss function $\mathcal{L}(y, \hat{y})$ for different tasks, say cross-entropy for classification task or mean square error loss for regression task, where y is the ground-truth label, and $\hat{y} = f_\theta(G)$, where θ are the parameters of HAMP model.

3.5 AHAMP: Extension for Graph Discovery and Interpretability

In this section, we extend HAMP for adaptive graph discovery and interpretability. Unlike GNNExplainer [55] which proposes the post-hoc interpretability method for homogeneous graphs, we propose a graph discovery and interpretability method for the heterogeneous network, and integrate it within HAMP. We name this model AHAMP as it adaptively discovers important edges for different tasks. The adaptive discovery of important edges has two advantages: (i) it can potentially improve model performance in a prediction task by down-weighting the less important edges for the task; and (ii) it enables interpretation of the contributions of network structural information to the prediction task.

We first introduce a learnable mask E_L (that is shared across all K layers) for edges E, and denote the masking element corresponding to each canonical triplet $\langle v_s, r, v_t \rangle \in E$ as $e_{l, \langle v_s, r, v_t \rangle} \in E_L$. We apply a sigmoid σ function to map each of the masking elements to $[0, 1]$: $\tilde{e}_{l, \langle v_s, r, v_t \rangle} = \sigma(e_{l, \langle v_s, r, v_t \rangle}) \in \tilde{E}_L$. Next, we extend Equation (6) as

$$H^{(k)}_{\langle v_s, r, v_t \rangle, t} = \sum_{v_s \in N(v_t)} \tilde{e}_{l, \langle v_s, r, v_t \rangle} \odot \left(\text{AttScore}^{(k)}_{\langle v_s, r, v_t \rangle} \cdot X''^{(k)}_{V, v_s} W^{(k)}_V \right), \tag{8}$$

where $\odot$ denotes element-wise multiplication. Following GNNExplainer [55], we introduce the following objective function:

$$\begin{aligned}\mathcal{L}' = \mathcal{L}(y, \hat{y} = \mathrm{f}_{\theta,\mathrm{E_L}}(G_L)) \\ - \alpha \sum_{\langle v_s, r, v_t\rangle \in E} \tilde{e}_{l,\langle v_s,r,v_t\rangle} \log(\tilde{e}_{l,\langle v_s,r,v_t\rangle}) + (1 - \tilde{e}_{l,\langle v_s,r,v_t\rangle}) \log(1 - \tilde{e}_{l,\langle v_s,r,v_t\rangle}) \\ + \beta \sum_{\langle v_s, r, v_t\rangle \in E} \tilde{e}_{l,\langle v_s,r,v_t\rangle}\end{aligned}, \tag{9}$$

where the first term is the same loss function used in HAMP for the respective tasks, except that the prediction $\hat{y}$ is based on the masked graph G_L instead of G; the second term is the element-wise binary entropy regularization term [43]; and the third term is the sparsity regularization term [56]. α and β are hyper-parameters. $\tilde{e}_{l,\langle v_s,r,v_t\rangle}$ is distinct from $\mathrm{AttScore}^{(k)}_{\langle v_s,r,v_t\rangle}$ as : (i) $\tilde{e}_{l,\langle v_s,r,v_t\rangle}$ is a learnable parameter shared across all K layers and applies to the whole model regardless of the number of layers in AHAMP, whereas $\mathrm{AttScore}^{(k)}_{\langle v_s,r,v_t\rangle}$ is a layer-specific score; and (ii) the element-wise binary entropy and sparsity regularization terms applied to $\tilde{e}_{l,\langle v_s,r,v_t\rangle}$ help AHAMP discover the most important edges that are relevant to the task. Our experiment results in Section 4.3 also show that the introduction of this graph discovery and interpretability method in AHAMP further improves model performance.

4 EXPERIMENTS

We now conduct several experiments to evaluate the network embeddings learned by HAMP and AHAMP against state-of-the-art baselines. In the following, we describe the experimental datasets and the predictive tasks for comparing the models.

4.1 Datasets

The data used in these experiments are extracted from RICO and ENRICO as mentioned in Section 1. Among the 9,384 Android applications in the RICO repository, we were able to scrape from the Google Play Store the metadata of 6,583 of these applications and their UI screens in Feb. 2020. These applications were released between Jan. 2010 and Apr. 2017. The repository includes images of UI screens and their associated UI view classes, as well as the interaction traces of the UI screens. To validate the result findings against UI datasets with different characteristics, we extract two datasets from the RICO repository, namely: **RICO-N**, comprising the most recently released 1,000 applications (Oct. 2015 to Apr. 2017); and **RICO-O**, comprising the earliest released 1,000 applications (Jan. 2010 to Aug. 2011). To assess HAMP's performance on predicting UI topics, we also utilize the list of UI screens and topic annotations provided in **ENRICO**, and extract other information corresponding to these UI screens (i.e., their mobile applications, UI classes and elements, multimodal, and positional attributes) from the RICO repository. The differences between these datasets are significant as shown in Table 2. RICO-O has around twice the number of nodes and edges as RICO-N, while ENRICO is the smallest dataset. The length of the longest UI screen sequence and the maximum depth of node hierarchy across the three datasets also differ.

For each mobile application, we parse the UI screens and UI view class hierarchies to extract the spatial, sequential, hierarchical, and network information—UI elements linked to their parent UI elements, each in turn linked to parent UI view classes, that are then linked to a series of UI screens, which constitute mobile applications. The result of this step is a heterogeneous network with the associated multimodal, spatial, sequential, and hierarchical level information shown in Figure 1.

Table 3 shows the multimodal and positional attributes of different node types in our datasets. The multimodal attributes X_{ft} for each node-type are derived by encoding the descriptions of

Table 2. Dataset Overview

Datasets	RICO-N	RICO-O	ENRICO
Num. of Application Nodes	1,000	1,000	869
Num. of UI Screen Nodes	5,879	9,108	1,460
Num. of UI View Class Nodes	1,563	2,920	1,506
Num. of UI Element Nodes	109,387	203,522	28,821
Num. of App - UI Screens Edges	5,879	9,108	1,460
Num. of UI Screens - UI View Classes Edges	38,961	68,305	10,113
Num. of UI View Classes - UI Elements Edges	109,387	203,522	28,821
Length of longest sequence	36	46	38
Max. depth of hierarchy	9	10	9
Num. of UI element component-types	26	26	–
Num. of UI screen genres	36	33	–
Num. of UI screen topics	–	–	20
Range of mobile app. ratings	1.15 to 4.92	1.72 to 4.91	–

RICO-N and RICO-O datasets are directly extracted from RICO [9] dataset. We utilize the RICO-N and RICO-O datasets for the UI screen genre and UI element component-type classification tasks as well as the application rating regression task. ENRICO [29] is a smaller dataset randomly sampled from across the RICO dataset (and hence overlaps with RICO-N and RICO-O datasets), with UI screen topics that had been manually annotated (that are not available in the original RICO dataset). We utilize the ENRICO dataset for the UI screen topic classification task.

Table 3. Multimodal Feature Dimensions and Availability of Spatial, Sequential, and Hierarchical Level Information for Different Node-types

	Feature Dim.	Spatial	Sequential	Hierarchical
Mobile Application Nodes - Textual - Glove vectors of app. descriptions	50	No	No	Yes
UI Screen Nodes - Visual - Latent vectors of UI screen images extracted with auto-encoder	64	No	Yes	Yes
UI View Class Nodes - Textual - CharNGram vectors of the names of UI view classes	100	No	No	Yes
UI Element Nodes - Visual - Latent vectors of UI element images extracted with pre-trained ResNet18	512	Yes	No	Yes

the mobile application nodes, images of the UI screen nodes, class names of the UI view class nodes, and images of the UI element nodes. Textual information of the mobile application descriptions is encoded with pre-trained Glove embeddings. Visual information of the UI screens is encoded by training an autoencoder. Textual information of the names of the UI view classes is first pre-processed by breaking them up by their periods, special characters, and camel casing. For example, *com.android. internal.policy.PhoneWindow$DecorView* is tokenized as *android, internal, policy, phone, window, decor, view*. Thereafter, we generate the features by using a pre-trained Char-NGram embedding. Images of the individual UI elements are first extracted using the bounds provided in the extracted UI code. Each image is then passed through a pre-trained ResNet18 model to generate its representations. Other methods can also be used to encode the attributes.

To represent the spatial position X_{sp} of UI element nodes in UI screens, we use the coordinates of the UI elements on the screen (x0,y0,x1,y1). The sequential positions X_{sq} of the UI screens are extracted from the user screen interaction sequences. The hierarchical levels X_{hi} are assigned based on the depth of the node in the hierarchical network.

4.2 Experiment Setup

We compare the performance of HAMP and AHAMP with state-of-the-art baselines on four predictive tasks: (a) classification of UI screen genres; (b) classification of UI element component types; (c) prediction of mobile application ratings; and (d) classification of UI screen topic.

- **Classification of UI screen genre**—For this task, we predict the genre labels of UI screens for the RICO-N and RICO-O datasets. We extract genre labels from the data scraped from the Google Play Store. To predict the UI screen genre, we pass the aggregated representation of a UI screen node generated by HAMP and AHAMP to a dense neural network layer with output dimensions equal to the number of genre classes, and train HAMP and AHAMP with cross-entropy loss. We use macro and micro F1 as the evaluation metrics as they combine both precision and recall which are important for this task. F1 is defined by the harmonic mean of precision and recall scores. For macro F1, we compute the F1 score for each class and average them. Macro F1 thus treats all classes equally in the averaging operation. Micro F1 score on the other hand is defined based on the precision and recall computed from the predicted genre class labels of all UI screens. As the distribution of genre classes is unequal, macro F1 and micro F1 scores can be quite different.
- **Classification of UI element component-types**—For this task, we predict the component-type of UI elements for the RICO-N and RICO-O datasets. Similar to UI screen genre classification, we create another dense neural network layer to predict the component-type of UI element nodes, and train HAMP and AHAMP with cross-entropy loss. We also use macro and micro F1 scores to evaluate the results of this task. To evaluate performance on this task in a manner that is not dependent on the Android nature of the UI view classes, a limitation pointed out in [34], we randomly initialize the attributes of the UI view class nodes instead of using the textual features of the UI view classes for this task. The other features used - application descriptions, UI screen, and element images—are not Android-specific.
- **Prediction of mobile application ratings**—For this task, we predict mobile application ratings for the RICO-N and RICO-O datasets. The rating of a mobile application is computed based on the average of all its user ratings from the Google Play Store. A dense neural network layer with an output dimension of one is added, and HAMP and AHAMP are trained with the mean square error loss for this task. This task is useful for predicting the success of new applications. We use **root mean square error** (**RMSE**) as the evaluation metric.
- **Classification of UI screen topic**—For this task, we predict the topic labels of UI screens for the ENRICO dataset. Similar to UI screen genre classification, we create another dense neural network layer to predict the topic of UI screen nodes, and train HAMP and AHAMP with cross-entropy loss. We also use macro and micro F1 scores to evaluate the results of this task.

For all four tasks, a different HAMP and AHAMP model are trained separately in a supervised manner, and the dataset is split for training/validation/testing in the ratio 60%/20%/20%, e.g., for UI screen genre classification, it means that the model is trained on a subset of 60% of UI screens with their genre labels, validated on 20% of UI screens with their genre labels, and testing results shown in the article based on the final 20% of UI screens with their genre labels. Labels are not utilized as input features, and UI objects that are not in the training, validation, or testing sets would not be used during training; or for validation or testing evaluations, respectively.

Baselines and Settings. We use multi-class logistic regression and linear regression as baselines for the classification task (for UI screen genre, UI element component type, and UI screen

topic classification) and regression task (for mobile application ratings) respectively. For these baselines, the visual features of the UI screen images and UI element images are used as inputs for the classification tasks, while the textual features of the mobile applications' descriptions are used as inputs for the regression task. We also choose an extensive set of state-of-the-art models as strong baselines:

—GCN [27], which normalizes the aggregated representations by the node degrees;
—SGC [51], which has been shown to achieve improved performance with simpler GCN layers;
—GraphSAGE [18] which allows us to adopt a different aggregation method—pooling;
—GAT [46], where different nodes in the neighborhood are assigned different importances during aggregation based on additive attention;
—hGAO [12], which applies hard attention to improve performance and reduce computational cost;
—HAN [49], which is also based on the additive attention mechanism but can deal with heterogeneous networks; and
—Screen2Vec [30], a recently proposed model that can capture both multimodal and sequential information within mobile UIs.

For each of these baselines (other than Screen2Vec), we similarly add a dense neural network layer with output dimensions equal to the number of classes for classification tasks (or a dimension equal to one in the case of the rating prediction task) and train these models with cross-entropy loss (or mean square error loss for the rating prediction task). For Screen2Vec, we utilize the pre-trained models provided by the authors of the work to generate the embeddings for the corresponding UI screens, and then use the embeddings as features to train a **Support Vector Machine** (**SVM**) for the UI screen genre and topic classification tasks. For the mobile application rating task, we obtain the mobile application embeddings by adding the embeddings of all its UI screens and then use the mobile application embeddings as features to train an SVM for the mobile application rating task. We do not compare against Screen2Vec on the UI element component-type classification task as the UI components in the Screen2Vec article differs from the UI elements in our article.

For HAMP, AHAMP, and all network-embedding baselines with attention modules, two layers ($K = 2$) and two heads (where applicable) are used. Two layers were chosen based on empirical experiments, indicating that two hop-away neighbors are useful for the selected predictive tasks. Two layer GNNs have also been found to achieve good results compared to deeper GNNs [1]. Based on our experiments with the validation dataset, we use 64 dimensions for the hidden representations generated by all models. α and β are set to 1.0 and 0.005, respectively, based on empirical experiments. A separate model is trained for each task in a supervised manner. For all models, an Adam optimizer with a maximum learning rate of 0.001 with a cosine annealing scheduler is used. All models are implemented in Pytorch and trained for 3,000 epochs on a 3.60 GHz AMD Ryzen 7 Windows desktop with NVIDIA RTX 3090 GPU and 64 GB RAM.

4.3 Results

4.3.1 UI Screen Genre Classification Results. Table 4 sets out the results relating to UI screen genre classification. HAMP and AHAMP clearly outperform all baselines by a significant margin. Among the baseline models, HAN, which also models heterogeneous network information, comes closest to HAMP and AHAMP, but the gap between the two is still significant. HAMP and AHAMP in particular perform much better than HAN on the RICO-O dataset. Screen2Vec also performs better than most of the other models, demonstrating the value of capturing multimodal and sequential information. However, there is a significant gap between Screen2Vec and

Table 4. UI Screen Genre Classification Results

	RICO-N		RICO-O	
	Micro F1	**Macro F1**	**Micro F1**	**Macro F1**
Log. Regression	0.127	0.059	0.153	0.043
GCN	0.087	0.048	0.137	0.042
SGC	0.046	0.011	0.113	0.012
GraphSAGE	0.079	0.035	0.136	0.038
GAT	0.079	0.058	0.159	0.063
hGAO	0.087	0.067	0.168	0.060
HAN	0.698	0.648	0.517	0.298
Screen2Vec	0.392	0.311	0.466	0.407
HAMP	0.970	0.877	0.921	0.759
AHAMP	**0.993**	**0.966**	**0.997**	**0.962**

Higher is better for micro F1 and macro F1. Best model(s) in bold; second-best model(s) underlined for this and subsequent tables. For the RICO-N dataset, we predict the genres of 5,879 UI screens. For the RICO-O dataset, we predict the genres of 9,108 UI screens. Our proposed models, HAMP and AHAMP outperform all baselines by a significant margin across both datasets.

Table 5. UI Element Component-type Classification Results

	RICO-N		RICO-O	
	Micro F1	**Macro F1**	**Micro F1**	**Macro F1**
Logistic Regression	0.587	0.166	0.616	0.185
GCN	0.587	0.220	0.627	0.236
SGC	0.519	0.215	0.549	0.248
GraphSAGE	0.649	0.279	0.694	0.274
GAT	0.638	0.331	0.693	0.398
hGAO	0.626	0.249	0.683	0.321
HAN	0.470	0.220	0.511	0.219
HAMP	**0.906**	0.891	0.899	0.801
AHAMP	**0.906**	**0.894**	**0.920**	**0.890**

Higher is better for micro F1 and macro F1. For the RICO-N dataset, we predict the component-types of 109,387 UI elements, while for the RICO-O dataset, we predict the component-types of 203,522 UI elements. While baselines perform relatively better on this task than on the UI screen genre classification task, HAMP and AHAMP still outperform all baselines by a significant margin across both datasets.

HAMP/AHAMP, demonstrating the importance of capturing structural network information. We also observe AHAMP consistently performing better than HAMP across datasets and metrics for this task, demonstrating the importance of discovering the more important edges that are relevant to the UI screen genre classification task. As we observe higher micro F1 than macro F1, the genre class distribution in this task is imbalanced.

4.3.2 UI Element Component-type Classification Results. Table 5 sets out the results of the experiments relating to UI element component-type classification. The baseline models perform better on this task compared with UI screen genre classification, though HAMP and AHAMP still outperform all baselines by a significant margin. For this task, HAN does not perform as well as the previous task. GraphSAGE and GAT's performance is closest to HAMP and AHAMP. The differences between performance on the UI element component-type and UI screen genre

Table 6. Application Rating Regression Results (RMSE)

	RICO-N	RICO-O
Linear Regression	0.540	0.669
GCN	0.761	4.131
SGC	1.969	1.900
GraphSAGE	0.500	0.595
GAT	1.354	1.011
hGAO	1.209	1.237
HAN	0.538	0.613
Screen2Vec	0.752	0.657
HAMP	**0.468**	0.577
AHAMP	0.469	**0.547**

RICO application rating ranges from 1 to 5. Lower is better for RMSE. For both the RICO-N and RICO-O datasets, we are predicting the ratings of 1,000 applications. While performance of baselines on this task is varied, HAMP and AHAMP similarly outperform all baselines across both datasets.

classification tasks could be due to differences in the density of different parts of the network, i.e., the ratio of the number of actual edges between nodes over all possible edges between nodes. HAMP and AHAMP are however able to cope with such differences, possibly due to their abilities to capture structural network, multimodal, spatial, sequential, and hierarchical information in a unified manner. We also observe AHAMP performing better than HAMP on this task for the RICO-O dataset, demonstrating the usefulness of discovering the important edges that are relevant to a task.

4.3.3 Application Rating Regression Results. Table 6 shows the results of the experiments relating to prediction of user ratings of mobile applications. HAMP and AHAMP similarly out-perform all baseline models. The performance of baselines is more varied for this task. As application nodes are at the highest level of the network, we could also view this as a sub-graph regression task. This could explain the better performance of GraphSAGE (which pools node representations in the aggregation step) and HAN (due to its ability to deal with heterogeneous networks) relative to other baselines. GCN's performance on the RICO-O dataset is surprisingly poor (with RMSE = 4.131). One possible explanation for its poorer performance could be due to over-smoothing, a well-known issue that GNNs often face [53]. Over-smoothing means that after several iterations of GNN message-passing, the representations for all nodes in a network become very similar, affecting model performance. To check if this is the cause, we ran this experiment with just one GCN layer (as opposed to two), and the performance improved (with RMSE = 2.641). HAMP and AHAMP are less likely to be affected by this issue due to the intrinsic regularization arising from their capturing of structural network, multimodal, spatial, sequential, and hierarchical information, and also due to the residual connections that were introduced in the model. The difference in performance between AHAMP and HAMP on this task is relatively small. Nonetheless, the introduction of the adaptive graph discovery and interpretability method in AHAMP did not lead to any decline in performance.

4.3.4 UI Screen Topic Classification Results. Table 7 sets out the results relating to UI screen topic classification on the ENRICO dataset. The results are consistent with the UI screen genre classification task. HAMP and AHAMP clearly outperform all baselines by a significant margin. Among the baseline models, HAN, which also models heterogeneous network information, comes

Table 7. UI Screen Topic Classification Results

	ENRICO	
	Micro F1	**Macro F1**
Log. Regression	0.264	0.118
GCN	0.290	0.110
SGC	0.179	0.016
GraphSAGE	0.335	0.231
GAT	0.305	0.196
hGAO	0.390	0.278
HAN	0.452	0.436
Screen2Vec	0.336	0.206
HAMP	**0.996**	0.996
AHAMP	**0.996**	**0.997**

Higher is better for micro F1 and macro F1. For the ENRICO dataset, we are predicting the topics of 1,460 UI screens. The results are consistent with the UI screen genre classification task and we see that HAMP and AHAMP outperform all baselines by a significant margin.

closest to HAMP and AHAMP, but the gap is still significant. Screen2Vec is the next best performing model for this task but its F1 results are substantially lower than HAMP and AHAMP, which again demonstrates the importance of capturing structural network information. As the topic class distribution in this task is also imbalanced, we similarly observe higher micro F1 than macro F1. For this task, the difference in performance between AHAMP and HAMP is small. AHAMP only outperforms HAMP on the macro F1 metric for the RICO-O dataset marginally. In other words, the introduction of the adaptive graph discovery and interpretability method in AHAMP did not lead to any decline in performance.

4.4 Ablation Studies

Table 8 sets out the results of the ablation studies for HAMP and AHAMP. From the results of the experiments in Section 4.3, the differences in performance between HAMP and AHAMP and the baseline models already illustrate the benefits of capturing hierarchical networks with different node and edge-types, and the effects of using the scaled dot-product attention message-passing mechanism for heterogeneous networks (which is not present in the baseline models). We further examine the importance of this feature of the HAMP/AHAMP model by using the same weights for all relationship-types, i.e., utilizing the same W_{att} and W_V across all relationship-types (denoted as **No heterogeneous weights**). We see that not capturing the heterogeneity of relationships between UI objects leads to a significant drop in performance. Performance similarly deteriorates significantly when the attention fusion module is not used (denoted as **No attention-fusion**) and we concatenate the multimodal attributes and spatial, sequential and hierarchical level information instead. Removing the PosVect module (denoted as **No PosVect**) also leads to a material drop in the performance of HAMP/AHAMP, albeit to a smaller degree. From the sensitivities of the HAMP/AHAMP model to **No heterogeneous weights**, **No attention-fusion** and **No PosVect**, we can surmise that the combination of these three proposed model features (i.e., heterogeneous weights, attention-fusion, and PosVect), together with the proposed scaled dot-product attention message-passing mechanism for heterogeneous networks (which is not present in the baseline models), account for most of the differences in performance between HAMP/AHAMP and the baseline models.

Table 8. Ablation Study (RICO-N and ENRICO)—For Component(Comp.)-type Classification, UI View Class Features had Already been Randomized in the Main Experiments

	Genre		Comp.-Type		App.	Topic	
	Micro F1	Macro F1	Micro F1	Macro F1	RMSE	Micro F1	Macro F1
No attention-fusion	0.860	0.697	0.884	0.763	0.480	0.840	0.834
No PosVect	0.916	0.746	0.898	0.840	0.488	0.921	0.874
No heterogeneous weights	0.901	0.717	0.888	0.783	0.481	0.849	0.795
Omit spatial location	0.969	0.871	0.899	0.861	0.468	0.965	0.961
Omit sequential position	0.967	0.864	0.905	0.888	0.473	0.976	0.966
Omit hierarchical levels	0.960	0.855	0.901	0.885	0.469	0.962	0.951
Random UI element features	0.930	0.796	0.846	0.676	0.469	0.927	0.919
Random UI view class features	0.948	0.881	–	–	0.468	0.947	0.917
Random UI screen features	0.965	0.888	0.901	0.870	0.468	0.954	0.945
Random app. features	0.886	0.812	0.896	0.838	0.468	0.957	0.957
Proposed HAMP	0.970	0.877	**0.906**	0.891	**0.468**	**0.996**	0.996
Proposed AHAMP	**0.993**	**0.966**	**0.906**	**0.894**	0.469	**0.996**	**0.997**

Higher is better for micro F1 (Micro) and macro F1 (Macro). Lower is better for RMSE.

The impact of varying the input information, by either omitting the positional information (denoted as **Omit spatial location**, **Omit sequential position**, and **Omit hierarchical levels**), or replacing each of the attribute feature matrices with a randomized matrix (denoted as **Random UI element features**, **Random UI view class features**, **Random UI screen features,** and **Random app. features**) have a less significant effect on HAMP/AHAMP's performance, but some of these effects are still material. Application textual and UI element visual features appear to contribute the most to HAMP/AHAMP's performance. For the application rating regression task, varying input information has a relatively small impact on HAMP/AHAMP's performance. Finally, we see that the introduction of the adaptive graph discovery and interpretability method in AHAMP leads to a material improvement in performance across a number of tasks and metrics.

5 INTERPRETABILITY

In this section, we interpret the contribution of different input information for the different tasks for the AHAMP model by examining the learnt attention weights and the contribution of heterogeneous network structural information for the different datasets and tasks. We analyze the AHAMP model as it is able to provide interpretations of the contribution of the multimodal and positional attributes, as well as the contribution of heterogeneous network structural information.

5.1 Interpreting Multimodal and Positional Attributes

We first interpret the contribution of the multimodal and positional attributes for the AHAMP model. To do so, we visualize the attention weights, β_m as described in Section 3, to interpret what AHAMP has learnt for each of the tasks. The attention weights, β_m, are learnt and used to weight different multimodal and positional attribute information during the attention-based fusion step, and provide an indication of the relative importance of the different input information for each task. The visualizations are shown in Figures 4–10. A darker shade of blue indicates higher attention weights, and hence a greater contribution to the model predictions for the respective tasks. Spatial bounds refer to the coordinates of UI element nodes, as explained in Section 4.1. Across all four tasks and the three datasets, multimodal features play an important role. For all tasks and

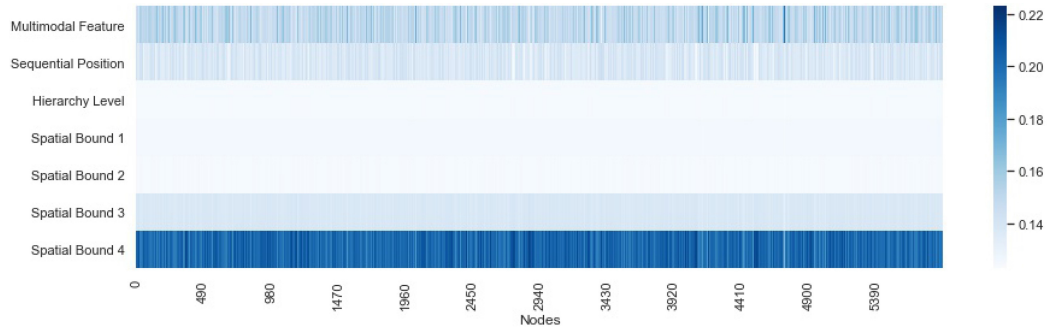

Fig. 4. Learnt Attention Weights for UI Screen Genre Classification (RICO-N). Darker shade of blue indicates higher attention weight β_m for the node. We see that multimodal feature, sequential position, and spatial bounds 3 and 4 are higher.

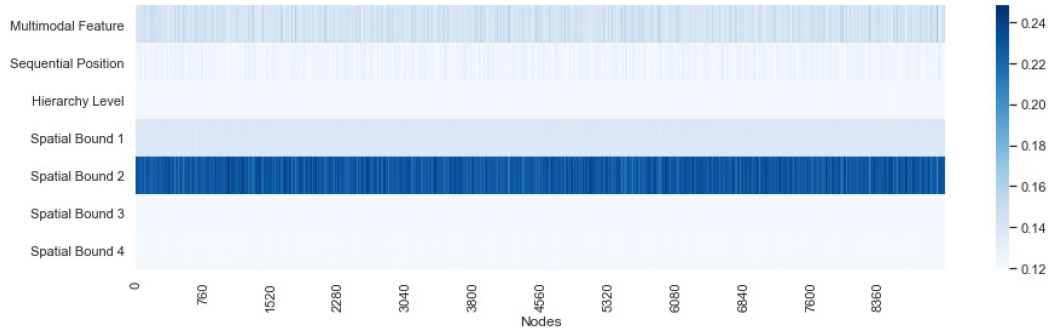

Fig. 5. Learnt Attention Weights for UI Screen Genre Classification (RICO-O). Darker shade of blue indicates higher attention weight β_m for the node. We see that multimodal feature, and spatial bounds 1 and 2 are higher.

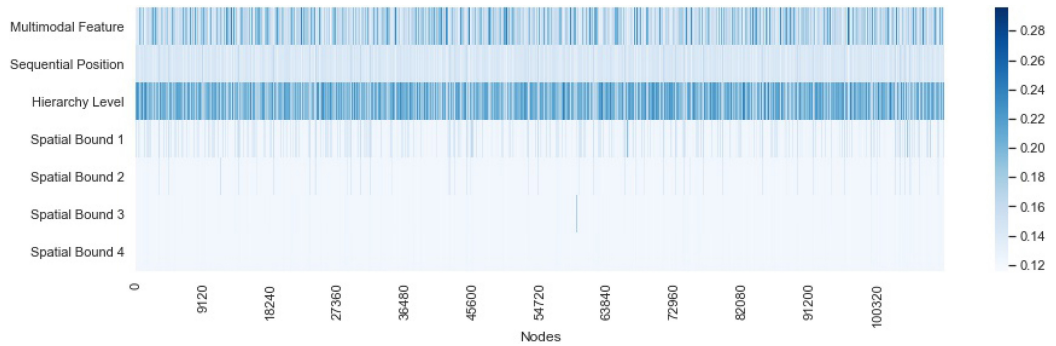

Fig. 6. Learnt Attention Weights for UI Element Component-Type Classification (RICO-N). Darker shade of blue indicates higher attention weight β_m for the node. We see that multimodal feature, sequential position, and hierarchy level are higher.

datasets, we also see different positional information playing important roles. For UI screen genre classification, the attention weights for spatial information, as shown in Figures 4 and 5 appears to be important, which could indicate the importance of element spatial positions in a UI layout for this task. An explanation for this could be that the patterns of spatial positions of different

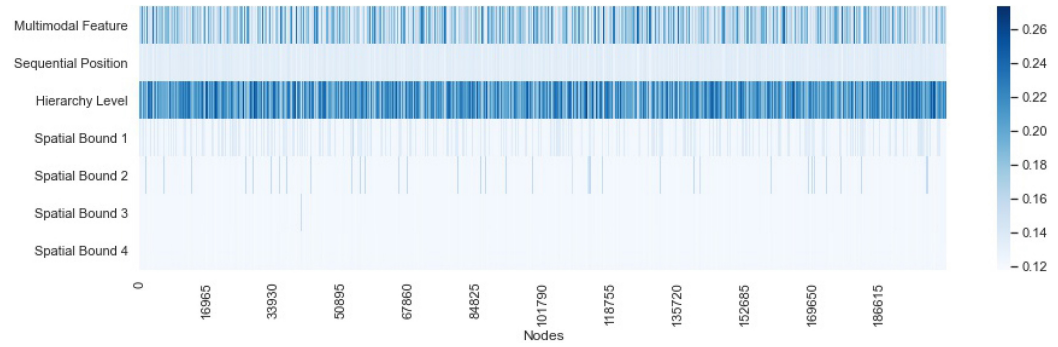

Fig. 7. Learnt Attention Weights for UI Element Component-Type Classification (RICO-O). Darker shade of blue indicates higher attention weight β_m for the node. We see that multimodal feature, sequential position and hierarchy level are higher.

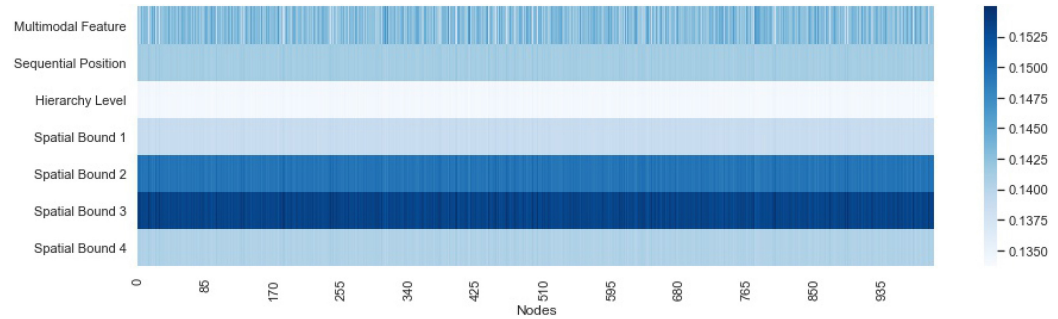

Fig. 8. Learnt Attention Weights for App. Rating Regression (RICO-N). Darker shade of blue indicates higher attention weight β_m for the node. We see that multimodal feature, sequential position and spatial bounds 2 and 3 are higher.

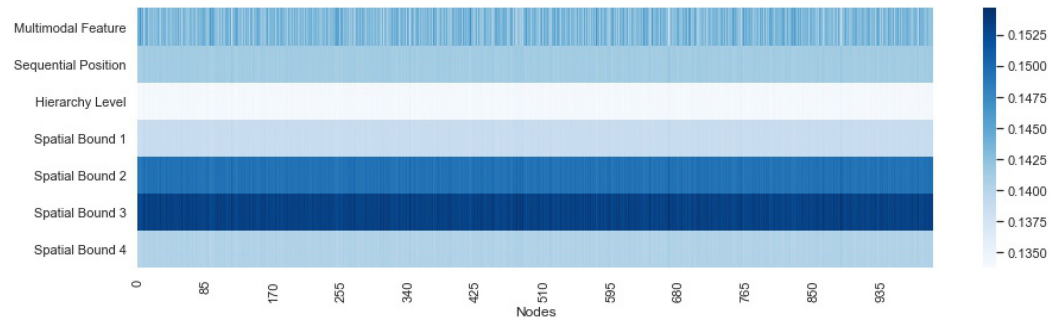

Fig. 9. Learnt Attention Weights for App. Rating Regression (RICO-O). Darker shade of blue indicates higher attention weight β_m for the node. We see that multimodal feature, sequential position and spatial bounds 2 and 3 are higher.

UI elements in a UI screen are indicative of the genre of the UI screen, e.g., the spatial positions of UI elements in UI screens that belong to the *shopping* genre (organized as repeated UI elements in lists or galleries) could be relatively more regular than UI screens that belong to the *game* genre (where the UI elements could be located in a variety of locations on the screen). For UI element

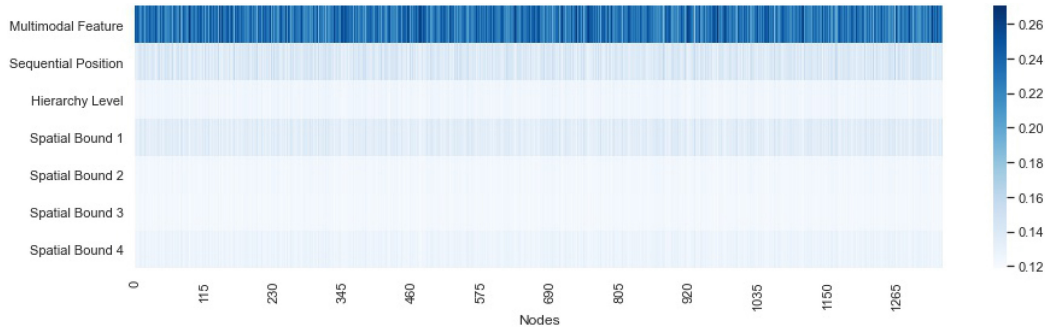

Fig. 10. Learnt Attention Weights for UI Screen Topic Classification (ENRICO). Darker shade of blue indicates higher attention weight β_m for the node. We see that multimodal feature is higher.

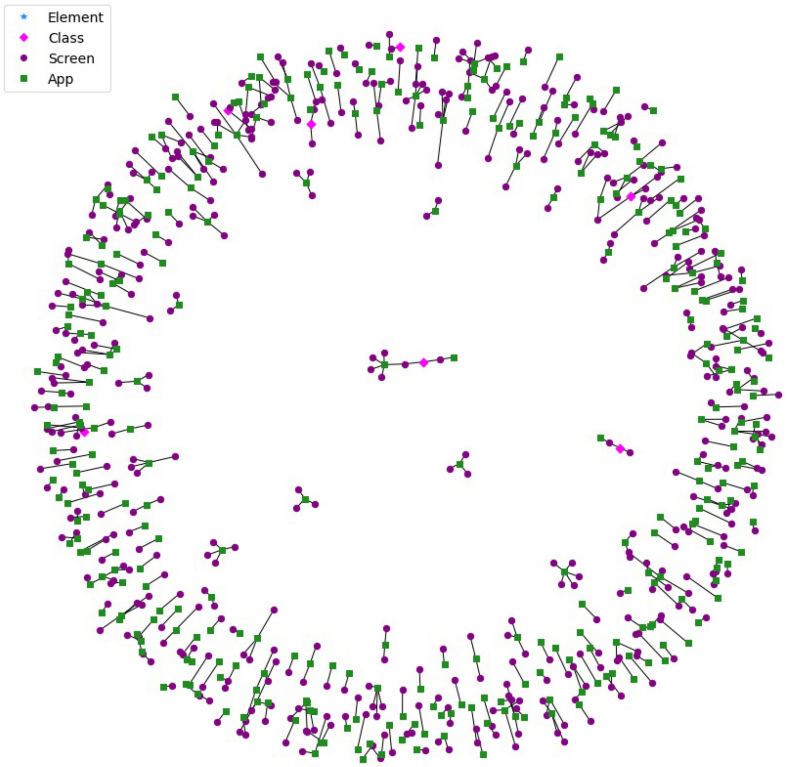

Fig. 11. Important Edges for UI Screen Genre Classification (RICO-N). We observe that the most important relationship edges for this task are between UI screen nodes (in purple) and application nodes (in green). Edges between UI screen nodes and application nodes account for 97.6% (488) of the important relationship edges; edges between UI screen nodes and UI class nodes account for 2.4% (12) of remaining important relationship edges.

component-type classification, the attention weights for hierarchy level information, as shown in Figures 6 and 7 are relatively higher, which makes intuitive sense, since the hierarchical level of elements are likely to be correlated with the role of UI elements in a UI layout. For example, a UI element that is of a *drawer* component-type (e.g., commonly used for menus that slide out

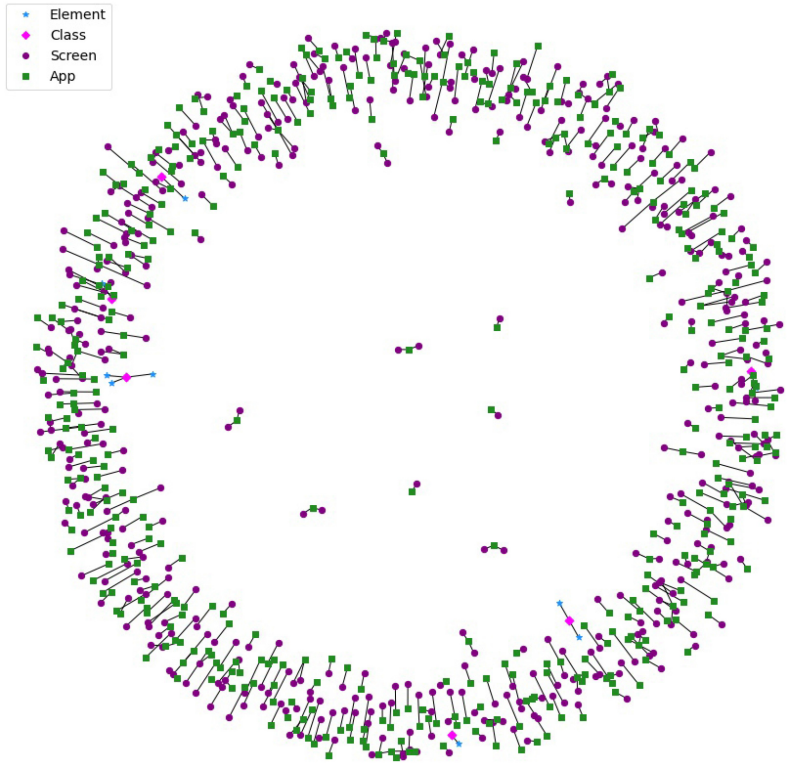

Fig. 12. Important Edges for UI Screen Genre Classification (RICO-O). We observe that the most important relationship edges for this task are between UI screen nodes (in purple) and application nodes (in green). Edges between UI screen nodes and application nodes account for 98.2% (491) of the important relationship edges; edges between UI element nodes and UI class nodes account for 1.8% (9) of remaining important relationship edges.

from the left of the UI screen) would be at a higher level in the hierarchy than UI elements of *text* component-type inside it. For application rating regression, other than spatial positional information, the attention weights for the sequential position, as shown in Figures 8 and 9 also play an important role. One explanation for this is that the sequential position and length of the UI user interaction traces has an important influence on user experience and hence affects average user ratings for the application, which makes intuitive sense. For example, a user utilizing an application with long sequences of UI screens in its user interaction traces is more likely to be frustrated at the time taken to navigate to a desired UI screen, and give a lower rating to the application.

5.2 Interpreting Heterogeneous Network Structural Information

Next we interpret the contribution of heterogeneous network structural information for the different datasets and tasks for the AHAMP model. Following GNNExplainer [55], we use a threshold to select the top 500 edges with the highest weights in $\tilde{E}_L$ and visualize the resultant set of edges with a spring layout to facilitate analysis, with node positions computed based on the Fruchterman-Reingold force-directed algorithm using a scale factor of 0.075, 50 iterations and a threshold of 0.0001.

Fig. 13. Important Edges for UI Element Component-Type Classification (RICO-N). We observe a number of important two-hop relationship edges for this task—between UI element nodes (in blue) and UI screen nodes (in purple) via UI class nodes (in magenta). Edges between UI element and UI element nodes, between UI element nodes and UI class nodes, between UI class nodes and UI screen nodes, and between UI screen nodes and application nodes account for 51.8% (259), 25.6% (128), 17.8% (89), and 4.8% (24) of the important relationship edges, respectively.

5.2.1 Network Interpretation for UI Screen Genre Classification. The edge visualizations for the UI screen genre classification task on the RICO-N and RICO-O datasets are shown in Figures 11 and 12, respectively. We observe that the most important relationship edges for this task are between UI screen nodes (in purple) and application nodes (in green) for both datasets. This makes intuitive sense, since the propagation of the textual description attributes of the application nodes across the application node to UI screen node edges are likely to play a key role in helping classify the UI genre.

5.2.2 Network Interpretation for UI Element Component-type Classification. The visualizations for the UI element component-type classification task on the RICO-N and RICO-O datasets are shown in Figures 13 and 14, respectively. We observe a number of important two-hop relationship edges for this task—between UI element nodes (in blue) and UI screen nodes (in purple) via UI class nodes (in magenta) for both datasets. Intuitively, this could be because the role of an element (i.e., the component-type of the element) is dependent on both the UI class used to instantiate the UI element, and the UI screen that the UI element is a part of.

5.2.3 Network Interpretation for Application Rating Regression. The visualizations for the application rating regression task on the RICO-N and RICO-O datasets are shown in Figures 15 and 16,

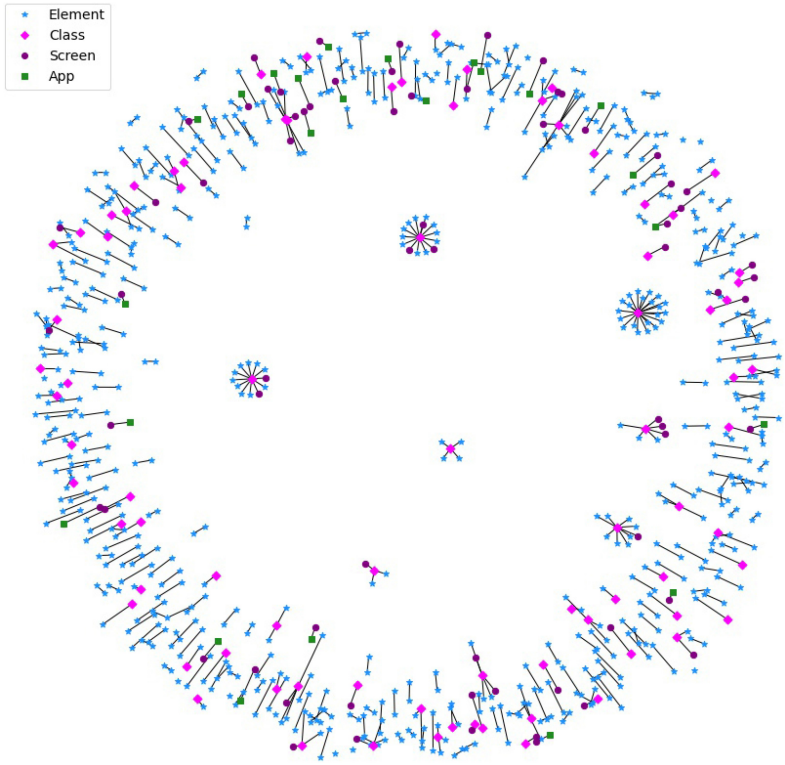

Fig. 14. Important Edges for UI Element Component-Type Classification (RICO-O). We observe a number of important two-hop relationship edges for this task—between UI element nodes (in blue) and UI screen nodes (in purple) via UI class nodes (in magenta). Edges between UI element and UI element nodes, between UI element nodes and UI class nodes, between UI class nodes and UI screen nodes, and between UI screen nodes and application nodes account for 58.8% (294), 27.2% (136), 8.8% (44), and 5.2% (26) of the important relationship edges, respectively.

respectively. We also observe some important two-hop relationship edges for this task—between application nodes (in green) and UI class nodes (in magenta) via UI screen nodes (in purple), particularly for the RICO-N dataset. Intuitively, this could be because the rating of an application may not only depend on the UI screen nodes, but also the choice of UI classes used to instantiate the elements that are used to construct the UI.

5.2.4 Network Interpretation for UI Screen Topic Classification. The visualization for the UI screen topic classification task on the ENRICO dataset is shown in Figure 17. We observe important three-hop relationship edges for this task in a large cluster. Intuitively, this could be because classifying the topic of a UI screen depends on the propagation of multimodal and positional information across multiple node-types.

5.3 Discussion

Based on the results of our experiments, we see that HAMP/AHAMP's ability to capture heterogeneous network structural information, along with the associated multimodal and positional attributes in a unified manner enables it to perform better on a number of tasks relating to the learning of UI semantics and metadata compared with an extensive set of state-of-the-art baselines.

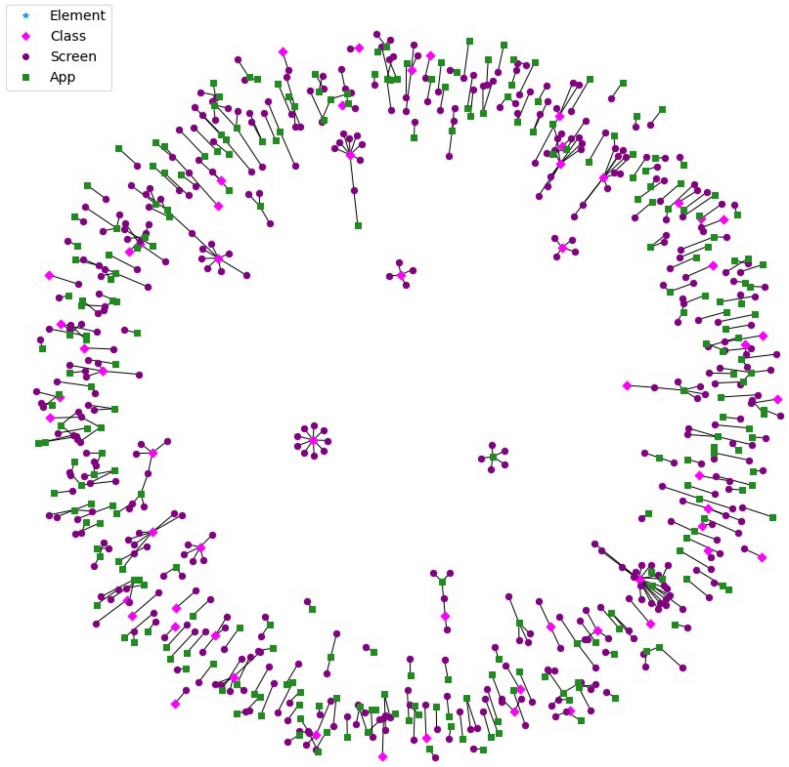

Fig. 15. Important Edges for App. Rating Regression (RICO-N). We observe a few important two-hop relationship edges for this task—between application nodes (in green) and UI class nodes (in magenta) via UI screen nodes (in purple). Edges between application and UI screen nodes account for 69.8% (349) of the important relationship edges, while edges between UI screen nodes and UI class nodes account for the remaining 30.2% (151) of the important relationship edges.

The distribution of genre, topic and component-type classes are imbalanced, and HAMP/AHAMP's relatively good performance on the macro F1 metric on the classification tasks (especially when compared with the baselines) demonstrate their abilities to perform well across different classes despite the imbalanced dataset. The experimental results provide evidence to support our intuition that capturing hierarchical relationships is important for UI-related tasks since part-whole hierarchies not only inform how a user understands the visual layout of UIs but also influence user interactions and experiences. Framing the UI dataset as a heterogeneous network enables similarities between nodes based on structural and regular equivalence to be captured, leading to better performance on tasks. The experiments also show that the key methods proposed in the HAMP/AHAMP's model—adapting the scaled dot-product attention mechanism for a network with different node and edge-types, and combining the proposed positional vectorizer with the attention fusion module to effectively capture both multimodal and positional node attributes - are effective in enabling it to perform better on a range of tasks. The introduction of the adaptive graph discovery and interpretability method in AHAMP further improves performance on a number of tasks and metrics, demonstrating the usefulness of discovering important edges that are relevant to the task. HAMP allows users to interpret the contribution of multimodal and positional information, while AHAMP further allows users to interpret the contribution of heterogeneous network structural information to the different mobile UI-related tasks. This enables users

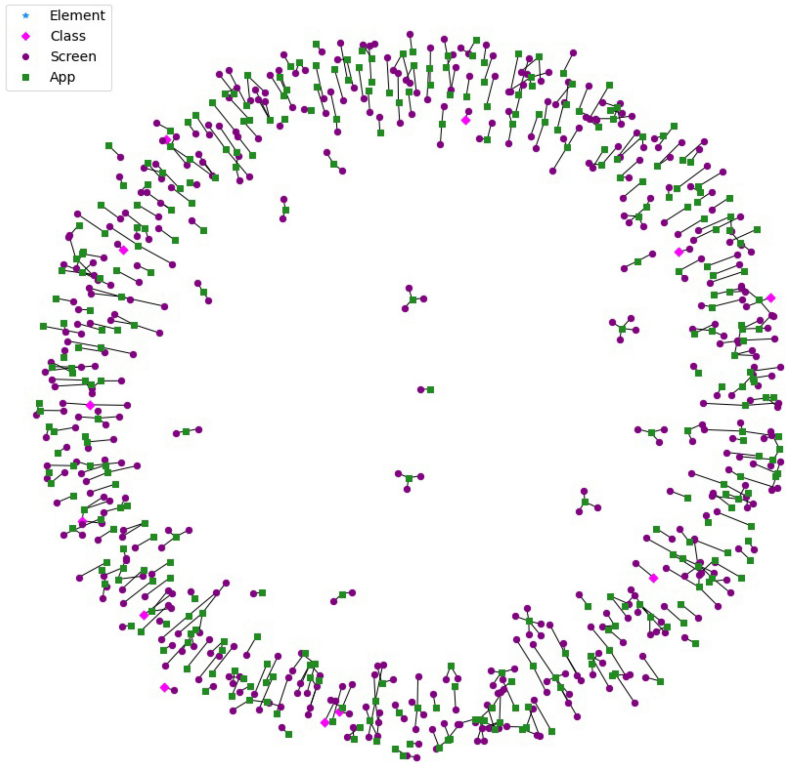

Fig. 16. Important Edges for App. Rating Regression (RICO-O). We observe a few important two-hop relationship edges for this task—between application nodes (in green) and UI class nodes (in magenta) via UI screen nodes (in purple). Edges between application and UI screen nodes account for 97.4% (487) of the important relationship edges, while edges between UI screen nodes and UI class nodes account for the remaining 2.6% (13) of the important relationship edges.

of model predictions to have greater confidence and trust in the predictions, and allows users to understand which UI attributes and parts of the UI network structure to focus on. From the visualization of the learnt attention weights shown in Figures 4–10, we see that positional attributes, i.e., spatial, sequential and hierarchical level information, play a key role in the tasks, highlighting the importance of capturing such information. Such attention weights also enable better interpretability of the predictions. From the visualization of the important discovered edges shown in Figures 11–17, we are able to better understand which parts of the UI network structure to focus on for the different tasks. The model is designed to be generalizable to other UI and design networks with such heterogeneous and hierarchical relationships, and multimodal and positional information. The model can accommodate different types of UI objects and relationships, information from different modalities, as well as different types of positional information. The choice of attention mechanisms that can weight more relevant information based on the domain and task, as well as the adaptive graph discovery mechanism enables the model to be adaptable to different UI applications. For example, a homogeneous network of UI objects with unimodal features would be a special case of the network proposed in our article, and such networks and their features could be captured by the HAMP and AHAMP models. The proposed model and framework is not domain specific and could be extended to other types of UIs, e.g., web, print, tangible, and other predictive tasks associated with such UIs.

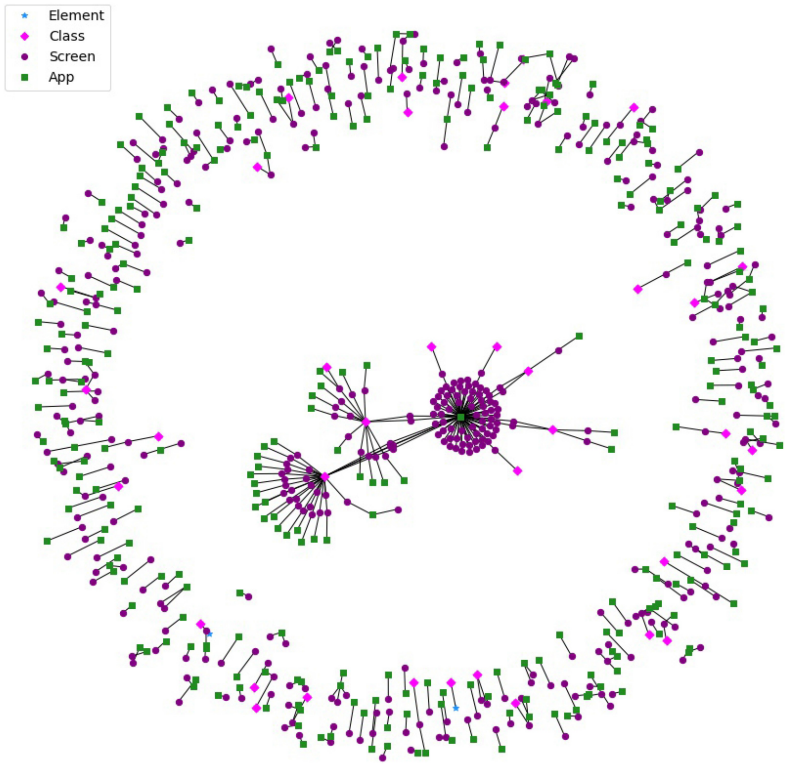

Fig. 17. Important Edges for UI Screen Topic Classification (ENRICO). We observe important three-hop relationship edges for this task in a large cluster. Edges between UI screen nodes and application nodes, between UI screen nodes and UI class nodes, and between UI class nodes and UI element nodes account for 83.2% (416), 16.4% (82), and 0.4% (2) of the important relationship edges, respectively.

6 CONCLUSION AND FUTURE WORK

In this article, we propose HAMP and AHAMP, novel scaled dot-product attention-based GNN message passing models designed for heterogeneous networks with multimodal and positional attributes, that allow users to interpret the contribution of multimodal and positional information, as well as heterogeneous network structural information to the different mobile UI-related tasks. Through experiments involving an extensive set of state-of-the-art baseline models, we demonstrate that HAMP and AHAMP outperform state-of-the-art models on a wide range of tasks for mobile application UIs that have real-world applications. Given HAMP/AHAMP's performance on the three distinct real world datasets, the model is likely to be equally applicable to other UIs even if the characteristics of the information differ. In addition to the tasks shown in Figure 2, future work could explore how HAMP and AHAMP could be used to complement or augment a range of other UI-related systems. HAMP and AHAMP could be incorporated within UI object detection and annotation systems in works such as [8, 20, 50, 58] to capture additional multimodal and structural network information. HAMP and AHAMP could also be used to complement systems that assist UI designers [6, 7, 28, 38]. Data relating to other types of UIs—web, print, tangible—could be collected and similar experiments conducted with such data for tasks in other domains involving design artifacts.

To promote social inclusion, HAMP and AHAMP can be used in real-world applications due to their ability to capture semantics from a wide range of information sources, for example, improving

automatic annotation of UIs for greater accessibility by disabled persons; or assisting designers in assessing their designs to improve the usability of UIs. We should however recognize that a greater dependence on models for such tasks could inadvertently lead to decisions that might disadvantage certain groups of users. For example, automatic annotations for accessibility might benefit certain groups of disabled users to the detriment of others if the data collection process is not carefully designed or is biased; designers might lean towards designs with higher ratings from broad user groups but which could be less accessible to specific groups of users (e.g., color-blind users). While we have designed HAMP and AHAMP with interpretability in mind to help address this, further work could be undertaken to explore how such negative effects could be further managed via better design of data collection processes, and/or greater model interpretability/explainability.

REFERENCES

[1] Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. 2019. N-GCN: Multi-scale graph convolution for semi-supervised node classification. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*.

[2] Gary Ang and Ee-Peng Lim. 2021. Learning network-based multi-modal mobile user interface embeddings. In *Proceedings of the ACM International Conference on Intelligent User Interfaces*.

[3] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. *Computing Research Repository (CoRR)* abs/1806.01261 (2018).

[4] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu, and Magy Seif El-Nasr. 2021. VINS: Visual search for mobile user interface design. *ACM Conference on Human Factors in Computing Systems (CHI'21)*. 423:1–423:14.

[5] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*.

[6] Chunyang Chen, Sidong Feng, Zhenchang Xing, Linda Liu, Shengdong Zhao, and Jinshui Wang. 2019. Gallery D.C.: Design search and knowledge discovery through auto-created GUI component gallery. *Proc. ACM Hum. Comput. Interact.* 3, CSCW (2019), 180:1–180:22.

[7] Chun-Fu (Richard) Chen, Marco Pistoia, Conglei Shi, Paolo Girolami, Joseph W. Ligman, and Yong Wang. 2017. UI X-ray: Interactive mobile UI testing based on computer vision. In *Proceedings of the ACM International Conference on Intelligent User Interfaces*.

[8] Jieshan Chen, Mulong Xie, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, and Guoqiang Li. 2020. Object detection for graphical user interface: Old fashioned or deep learning or a combination?. In *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.

[9] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology*.

[10] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*.

[11] Vijay Prakash Dwivedi and Xavier Bresson. 2021. A generalization of transformer networks to graphs. *Computing Research Repository (CoRR)* abs/2012.09699 (2020).

[12] Hongyang Gao and Shuiwang Ji. 2019. Graph representation learning via hard and channel-wise attention networks. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*.

[13] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the International Conference on Machine Learning*.

[14] Luke B. Godfrey and Michael S. Gashler. 2018. Neural decomposition of time-series data for effective generalization. *IEEE Transactions on Neural Networks and Learning Systems* 29, 7 (2018), 2973–2985.

[15] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. DynGEM: Deep embedding method for dynamic graphs. *Computing Research Repository (CoRR)* abs/1805.11273 (2018).

[16] Asnat Greenstein-Messica, Lior Rokach, and Michael Friedman. 2017. Session-based recommendations using item embedding. In *Proceedings of the ACM International Conference on Intelligent User Interfaces*.

[17] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna R. Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational graph recurrent neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[18] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[19] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval*.

[20] Zecheng He, Srinivas Sunkara, Xiaoxue Zang, Ying Xu, Lijuan Liu, Nevan Wichers, Gabriel Schubiner, Ruby B. Lee, and Jindong Chen. 2021. ActionBert: Leveraging user actions for semantic understanding of user interfaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[21] Geoffrey E. Hinton. 2021. How to represent part-whole hierarchies in a neural network. *Computing Research Repository (CoRR)* abs/2102.12627 (2021).

[22] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of the World Wide Web Conference*.

[23] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. Swire: Sketch-based user interface retrieval. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*.

[24] Wentao Huang, Yuchen Li, Yuan Fang, Ju Fan, and Hongxia Yang. 2020. BiANE: Bipartite attributed network embedding. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval*.

[25] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*.

[26] Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. In *Proceedings of the NIPS Workshop on Bayesian Deep Learning*.

[27] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*.

[28] Chunggi Lee, Sanghoon Kim, Dongyun Han, Hongjun Yang, Young-Woo Park, Bum Chul Kwon, and Sungahn Ko. 2020. GUIComp: A GUI design assistant with real-time, multi-faceted feedback. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*.

[29] Luis A. Leiva, Asutosh Hota, and Antti Oulasvirta. 2020. Enrico: A dataset for topic modeling of mobile UI designs. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services*.

[30] Toby Jia-Jun Li, Lindsay Popowski, Tom M. Mitchell, and Brad A. Myers. 2021. Screen2Vec: Semantic embedding of GUI screens and GUI components. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*.

[31] Yiqiao Li, Jianlong Zhou, Sunny Verma, and Fang Chen. 2022. A survey of explainable graph neural networks: Taxonomy and evaluation metrics. *Computing Research Repository (CoRR)* abs/2207.12599 (2022).

[32] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the World Wide Web Conference*.

[33] J. Liu, C. Xu, C. Yin, W. Wu, and Y. Song. 2022. K-core based temporal graph convolutional network for dynamic graphs. *IEEE Trans. Knowl. Data Eng.* 34, 8 (2022), 3841–3853.

[34] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning design semantics for mobile apps. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. Berlin, Germany.

[35] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized explainer for graph neural network. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[36] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-embedding attributed networks. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

[37] Zaiqiao Meng, Shangsong Liang, Jinyuan Fang, and Teng Xiao. 2019. Semi-supervisedly co-embedding attributed networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[38] Peter O'Donovan, A. Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with interactive layout suggestions. *In Proceedings of the ACM Conference on Human Factors in Computing Systems*.

[39] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *Proceedings of the International Conference on Learning Representations*.

[40] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2020. EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[41] Aneeshan Sain, Ayan Kumar Bhunia, Yongxin Yang, Tao Xiang, and Yi-Zhe Song. 2021. StyleMeUp: Towards style-agnostic sketch-based image retrieval. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[42] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the International Semantic Web Conference*.

[43] Chao Shang, Jie Chen, and Jinbo Bi. 2021. Discrete graph structure learning for forecasting multiple time series. In *Proceedings of the International Conference on Learning Representations*.

[44] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *Computing Research Repository (CoRR)* abs/1706.02263 (2017).

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. *In Proceedings of the International Conference on Learning Representations*.

[47] Minh N. Vu and My T. Thai. 2020. PGM-explainer: Probabilistic graphical model explanations for graph neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[48] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval*.

[49] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous graph attention network. In *Proceedings of the World Wide Web Conference*.

[50] Thomas D. White, Gordon Fraser, and Guy J. Brown. 2019. Improving random GUI testing with image-based widget detection. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*.

[51] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying graph convolutional networks. In *Proceedings of the International Conference on Machine Learning*.

[52] Yingtao Xie, Tao Lin, and Hongyan Xu. 2019. User interface code retrieval: A novel visual-representation-aware approach. *IEEE Access* 7 (2019), 162756–162767.

[53] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the International Conference on Machine Learning*.

[54] Shaowei Yao, Tianming Wang, and Xiaojun Wan. 2020. Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[55] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating explanations for graph neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[56] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[57] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2020. Explainability in graph neural networks: A taxonomic survey. *Computing Research Repository (CoRR)* abs/2012.15445 (2020).

[58] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle I. Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P. Bigham. 2021. Screen recognition: Creating accessibility metadata for mobile applications from pixels. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*.

Received 19 June 2022; revised 1 October 2022; accepted 20 November 2022