3-2022

Learning user interface semantics from heterogeneous networks with multi-modal and positional attributes

Gary ANG
Singapore Management University, gary.ang.2019@phdcs.smu.edu.sg

Ee-peng LIM
Singapore Management University, eplim@smu.edu.sg

Citation

ANG, Gary and LIM, Ee-peng. Learning user interface semantics from heterogeneous networks with multi-modal and positional attributes. (2022). *Proceedings of the 27th Annual Conference on Intelligent User Interfaces, Virtual, 2022 March 22-25*. 433-446.
Available at: https://ink.library.smu.edu.sg/sis_research/6918

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

Gary Ang
Singapore Management University
Singapore, Singapore
gary.ang.2019@phdcs.smu.edu.sg

Ee-Peng Lim
Singapore Management University
Singapore, Singapore
eplim@smu.edu.sg

ABSTRACT

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, 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 paper 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 out-performs other state-of-the-art models on such tasks. We also report our ablation study results on HAMP.

CCS CONCEPTS

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

KEYWORDS

Graph neural networks, transformers, attention mechanism, heterogeneous networks, multimodal, mobile application user interface, supervised learning

ACM Reference Format:
Gary Ang and Ee-Peng Lim. 2022. Learning User Interface Semantics from Heterogeneous Networks with Multimodal and Positional Attributes. In *27th International Conference on Intelligent User Interfaces (IUI '22), March 22–25, 2022, Helsinki, Finland*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3490099.3511143

IUI '22, March 22–25, 2022, Helsinki, Finland

ACM ISBN 978-1-4503-9144-3/22/03…$15.00
https://doi.org/10.1145/3490099.3511143

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 [32] 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 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. Example 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 [20]. 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 learnt for every UI object which can be used in downstream tasks such as UI search, evaluation, annotation, and organization.

In this paper, we use the RICO repository [8, 32] and an enhanced version of RICO, ENRICO [28], 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 1460 UI screens manually annotated with design topics, e.g., a UI screen with a *dialer*, *tutorial*, or *news* topic. **Assigning topics to a UI screen is a non-trivial task, as the same set of UI elements could be used to compose UIs of different topics, e.g. *form*, or *login* topic, and the differences between screens with different topics may require an understanding of the content, e.g. *list*, or *news* topic.** 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 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 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 paper, and examples of predictive application tasks that may be performed using a heterogeneous attributed network model: i) automatic annotation of elements' component type, which could be used to insert metadata for accessibility features in mobile applications; ii) automatic classification of UIs' genres, which could be used for organization, search and retrieval of UIs in a repository; iii) UI rating prediction, which could be used for UI designers to obtain an initial evaluation of their designs; and iv) automatic tagging of UIs with their topics, which could be used for applications such as semantic UI captioning, automatic UI tagging and annotation [28].

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 specifically model spatial and sequential node positional information.

There are works that capture UI multimodal information, such as [15, 29], which generate item embeddings for recommendation tasks from user interaction information and metadata text but do not capture multimodal and positional attributes of a heterogeneous network of UI design objects. [1] uses multimodal GNNs for mobile UI-related tasks but does not incorporate spatial, sequential and hierarchical information, and is designed for bipartite networks, a special type of heterogeneous network. [29] captures multimodal and sequential UI information, but does not capture structural network and positional information.

In this paper, we propose the Heterogeneous Attention-based Multimodal Positional (HAMP) model to address the limitations of such existing models. Specifically, HAMP aims to: (1) capture 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) ensure that low dimensional positional attributes are captured effectively alongside high dimensional multimodal attributes; and (3) self-discover the relative importance of multimodal attributes and positional attributes.

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

- To our knowledge, this is the first work to propose an 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; and
- We show that HAMP consistently out-performs 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

[3, 6, 22, 38, 47] 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. [22] retrieves UI screen images based on UI sketch images by using image embeddings for visual similarity comparisons, but does not capture the structural network information present in the linkages between UI objects. [47] captures structural network information of UI objects to support retrieval applications, but does not capture multimodal attributes. [1] captures structural network and multimodal information but does not incorporate spatial, sequential and hierarchical information, and is designed for a bipartite network, a special kind of heterogeneous network. Screen2Vec [29], a recent work, generates representations

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

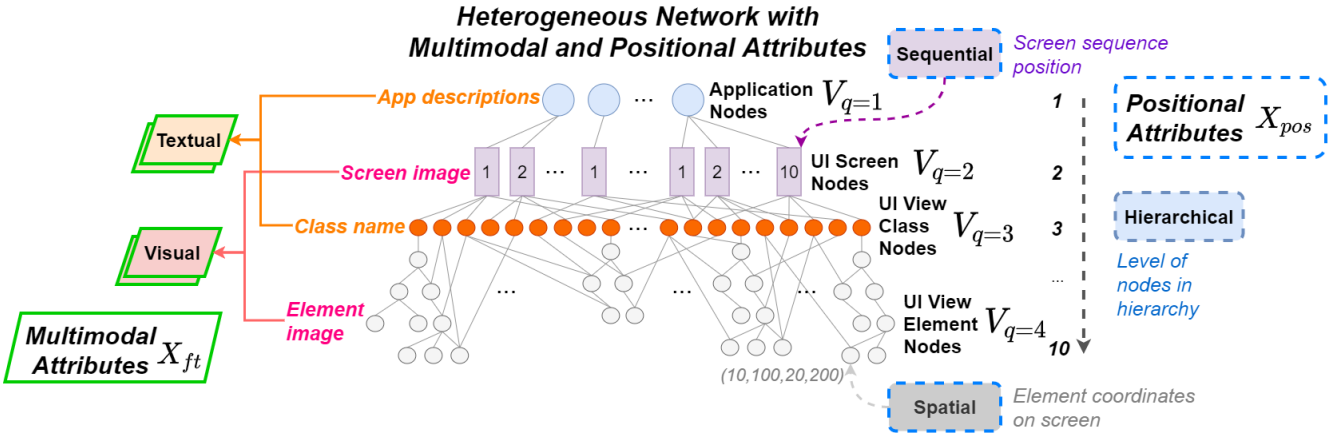

Figure 1: Heterogeneous network to capture hierarchy of UI objects and their multimodal and positional attributes. The heterogeneous network comprises four types of nodes - mobile application nodes, UI screen nodes, UI view class nodes and UI view element nodes, with attributes from different modalities, 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.

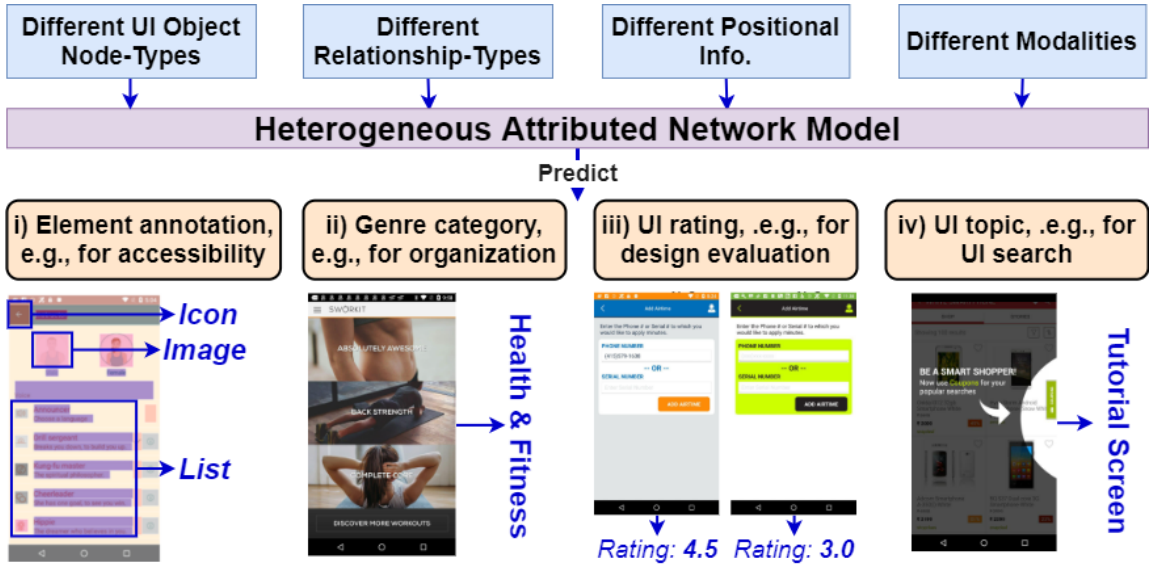

Figure 2: Overview of framework in this paper. A heterogeneous attributed network model captures different UI object node-types with different inter-UI object relationships and multimodal and positional attributes for four different predictive tasks.

of UI screens and components that capture the multi-modal 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.

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) [25] approach applies a variational autoencoder [24] framework to learn the node embeddings of homogeneous networks. The Co-Embedding Attributed Network (CAN) [33] uses two VAE channels to jointly encode and decode the node adjacency matrix and another node feature matrix. Semi-supervised Co-embedding Attributed Network [34] extends CAN to co-embed both attributes and nodes of partially labelled networks. Multinomial VAE [30] is a VAE-based approach that

generates embeddings of heterogeneous network by using a multinomial distribution instead of the Bernoulli distribution used in GVAE.

Graph neural network (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 [2, 12]. Several GNN-based models have been developed. In particular, Graph Convolutional Network (GCN) [26] aggregates features of neighboring nodes and normalizes the aggregated representations by the node degrees. GraphSAGE [17] 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) [42] 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 Network (SGC) [46] further adopts linear layers to process messages as they are passed to neighboring nodes. Hard Graph Attention Operator (hGAO) [11] 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 [39] and Graph Convolutional Matrix Completion [40] use multiple GCNs to encode embeddings of multiple adjacency matrices, one for each edge type, before aggregating them. Neural Graph Collaborative Filtering [43] and LightGCN [18] encode embeddings for different number of hops before aggregating them. [23] captures indirect proximity between the same node types in bipartite networks. Heterogeneous Graph Attention Network (HAN) [44] and General Attributed Multiplex Heterogeneous Network [4] use multiple GNN-based layers to encode networks formed from different metapaths [9] before using an attention mechanism to aggregate the embeddings.

Other than VAE and GNN-based approaches, transformers [41], initially designed for modeling sequential information, have also been generalized for networks/graphs [10]. 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 [21, 49].

Most of the related network embedding works only use information from a single modality, whereas HAMP captures information from multiple modalities. These related works also do not capture spatial, sequential and hierarchical information. With regards to sequential information, there are GNN-related works [14, 16, 31, 37] 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 paper. We also propose the use of a module that can accommodate different functional forms (the positional vectorizer) to extract important lower dimensional spatial, sequential and hierarchical features (representing linear and non-linear patterns) and expand their dimensions to match higher dimensional textual and visual information, which has thus far not featured in GNN-related works. We also use an attention mechanism to self-discover the 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.

3 HETEROGENEOUS ATTENTION-BASED MULTIMODAL POSITIONAL GRAPH NEURAL NETWORK

In our proposed Heterogeneous Attention-based Multimodal Positional (HAMP) GNN model, 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 $V = V_1 \cup \cdots \cup V_Q$. Similarly, E consists of edges of R types, i.e., $E = E_1 \cup \cdots \cup E_R$. In the case of the RICO dataset, each edge type represents a specific part-whole relationship as shown in Figure 1. 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 $r \in \{1, \cdots, R\}$. The set of all canonical triplets is denoted as C.

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. 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.

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, *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

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

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.

Multimodal positional fusion. We first project each multimodal attribute vector (with different dimensions) to a common dimension with a dense layer - $X'_{ft} = PROJ(X_{ft})$. 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. $X'_{pos} = PosVect(X_{pos})$. That is, X'_{ft} and X'_{pos} share the same dimension size. The positional vectorizer, inspired by [13, 36], 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 [41], 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 on the RICO dataset, we empirically chose Linear, Sinusodial, 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}. The use of an attention mechanism here 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 importances of the different inputs. We first apply a non-linear transformation to each of these intermediate representations to obtain the scalars K_m for attribute m where $m \in \{ft, pos\}$.

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

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'':

$$X'' = DENSE(\sum_m \beta_m X'_m) \tag{4}$$

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 $W^{(0)}_K, W^{(0)}_Q, W^{(0)}_V, W^{(1)}_K, W^{(1)}_Q, W^{(1)}_V, b_K$, 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.

Scaled dot-product attention message passing. In this step, inspired by [21, 49], we adapt the scaled dot-product attention module commonly used in transformers [41] 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 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

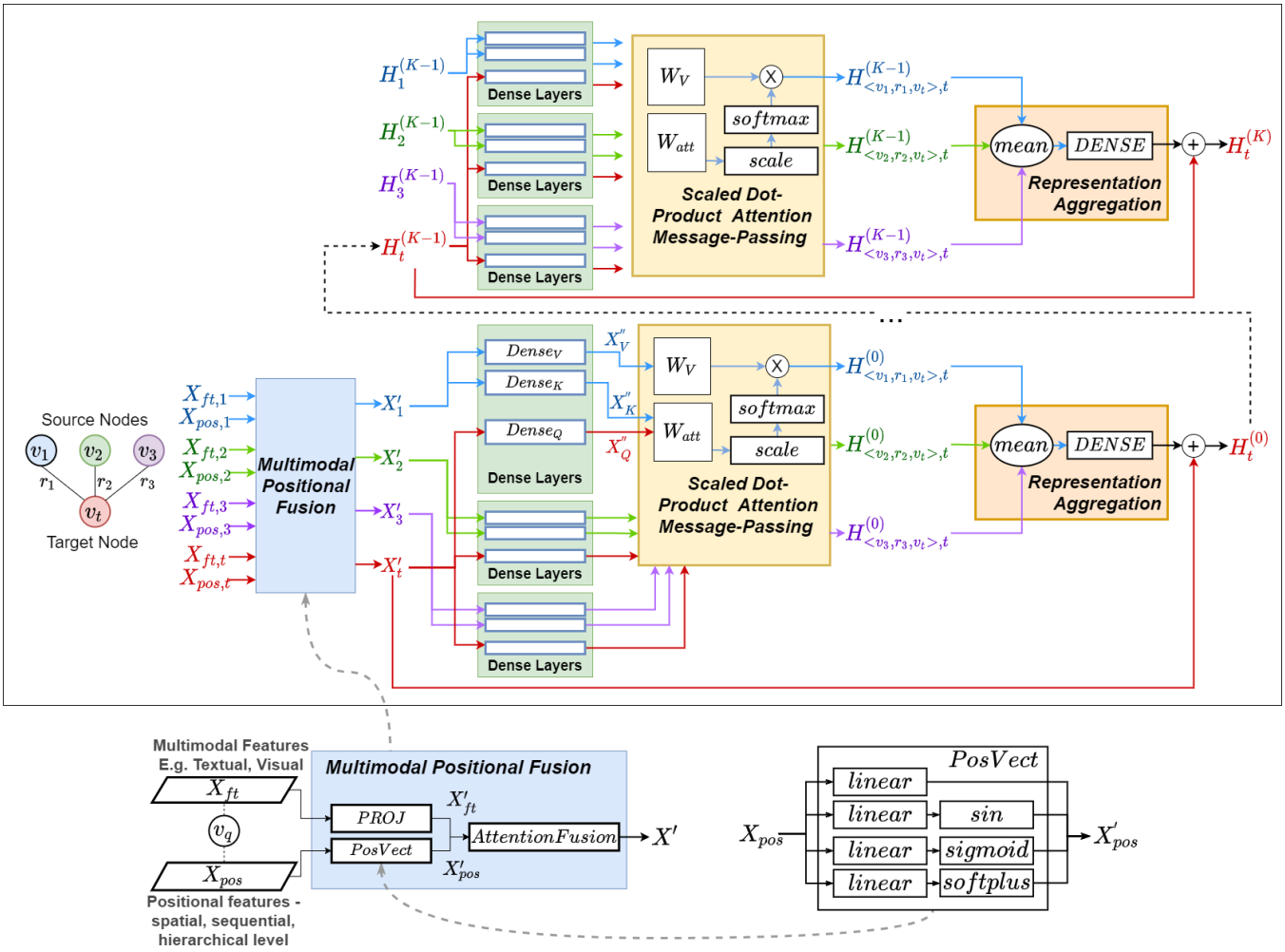

Figure 3: Architecture of HAMP model, which comprises three main components: positional vectorizer (PosVect) that captures linear, non-linear and periodic relationships between positional attributes and different tasks; multimodal positional fusion module that projects multimodal and positional features to a common latent representational space and fuses them with attention mechanisms; and network-encoding layers that capture structural heterogeneous network information.

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 importances 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:

$$AttScore_{\langle v_s,r,v_t\rangle} = softmax_{v_s \in N(v_t)} scale(X''_{K,v_s} W_{att} X''_{Q,v_t}) \tag{5}$$

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 [41].

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.

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

where W_V is a learnable weight matrix. Multiple heads can also be incorporated within each scaled dot-product attention module per [41].

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 $H_t^{\prime(0)}$. That is,

$$H_t^{\prime(0)} = \frac{1}{|N(v_t)|} \sum_{v_s \in N(v_t)} H_{\langle v_s, r, v_t \rangle, t} \tag{7}$$

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 is applied to obtain the representation of the node $H_t^{(0)} = H_t^{\prime(0)} + X_t'$, which can then serve as input to another layer comprising the dense layers, scaled dot-product attention message-passing and representation aggregation modules.

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 $H_t^{\prime(k)}$. A residual connection is then applied to obtain $H_t^{(k)} = H_t^{\prime(k)} + H_t^{(k-1)}$.

$H_t^{(k)}$ 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 [48]. 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.

4 EXPERIMENTS

We now conduct several experiments to evaluate the network embeddings learned by HAMP against state-of-the-art baselines. In the following, we describe the experiment 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 9384 Android applications in the RICO repository, we were able to scrape from the Google Play Store the metadata of 6583 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 1000 applications (Oct 2015 to Apr 2017); and **RICO-O**, comprising the earliest released 1000 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 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 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 CharNGram 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 their 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 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 to a dense neural network layer with output dimensions equal to the number of genre classes, and train HAMP 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

Table 2: Dataset Overview

Datasets	RICO-N	RICO-O	ENRICO
Num. of Application Nodes	1000	1000	869
Num. of UI Screen Nodes	5879	9108	1460
Num. of UI View Class Nodes	1563	2920	1506
Num. of UI Element Nodes	109,387	203,522	28,821
Num. of App - UI Screens Edges	5879	9108	1460
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	-

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

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 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 [32], 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 are computed based on the average of all its user ratings from the Google Play Store. A dense neural network layer with output dimension of one is added, and HAMP is 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 with cross-entropy loss. We also use macro and micro F1 scores to evaluate the results of this task.

For all four tasks, the dataset is splitted 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 paper 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 [26], which normalizes the aggregated representations by the node degrees;
- SGC [46], which has been shown to achieve improved performance with simpler GCN layers;
- GraphSAGE [17] which allows us to adopt a different aggregation method - pooling;
- GAT [42], where different nodes in the neighborhood are assigned different importances during aggregation based on additive attention;
- hGAO [11], which applies hard attention to improve performance and reduce computational cost;
- HAN [44], which is also based on the additive attention mechanism but can deal with heterogeneous networks; and
- Screen2Vec [29], 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 a 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 paper differ from the UI elements in our paper.

For HAMP 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 neighbours are useful for the selected predictive tasks. Based on our experiments with the validation dataset, we use 64 dimensions for the hidden representations generated by all models. 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 3000 epochs on a 3.60GHz AMD Ryzen 7 Windows desktop with NVIDIA RTX 3090 GPU and 64GB RAM.

Table 4: UI Screen Genre Classification Results. Higher is better for micro F1 and macro F1. Best performing results are boldfaced in this and subsequent tables.

	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**

4.3 Results

4.3.1 ***UI Screen Genre Classification Results.*** Table 4 sets out the results relating to UI screen genre classification. HAMP clearly out-performs all baselines by a significant margin. Among the baseline models, HAN, which also models heterogeneous network information, comes closest to HAMP, but the gap between the two is still significant. HAMP in particular performs 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 HAMP, demonstrating the importance of capturing structural network information. 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 still out-performs 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. The differences between performance on the UI element component-type and UI screen genre classification tasks could be due to differences in the density of different parts of the network. HAMP is however able to cope with such differences, possibly due to its ability to capture structural network, multimodal, spatial, sequential and hierarchical information in a unified manner.

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 similarly out-performs 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

Table 5: UI Element Component-Type Classification Results. Higher is better for micro F1 and macro F1.

	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**

Table 6: Application Rating Regression Results (RMSE). RICO application rating ranges from 1 to 5. Lower is better for 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**

Table 7: UI Screen Topic Classification Results. Higher is better for micro F1 and macro F1.

	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**

to deal with heterogeneous networks) relative to other baselines. GCN's performance on the RICO-O dataset is surprisingly poor (RMSE of 4.131). One possible explanation for its poorer performance could be due to over-smoothing, a well known issue that GNNs often face [48]. 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 (RMSE of 2.641). HAMP is less likely to be affected by this issue due to the intrinsic regularization arising from its capturing of structural network, multimodal, spatial, sequential and hierarchical information, and also due to the residual connections that were introduced in the model.

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 clearly out-performs all baselines by a significant margin. Among the baseline models, HAN, which also models heterogeneous network information, comes closest to HAMP, but the gap between the two is still significant. We also see that Screen2Vec performs better than most of the other models but that there is still a significant gap between Screen2Vec and HAMP, 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.

4.4 Ablation Studies

Table 8 sets out the results of the ablation studies. The differences in performance between HAMP 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 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, albeit to a smaller degree. From the sensitivities of the HAMP 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

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. Higher is better for micro F1 and macro F1. Lower is better for RMSE.

	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

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 and the baseline models.

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's performance, but some of these effects are still material. Application textual and UI element visual features appear to contribute the most to HAMP's performance. For the regression tasks, varying input information has a relatively small impact on HAMP's performance.

4.5 Interpretability

We also visualize the attention weights, β_m as described in Section 3, to interpret what HAMP has learnt for each of the tasks and the relative importance of the different input information. The visualizations are shown in Figure 4. A darker shade of blue indicates higher attention weights. The attention weights for the multimodal features (row 1) are generally higher for the UI screen genre and topic classification task, as well as the UI element component-type classification task. Interestingly, they do not feature prominently for the application rating regression task. Instead, the attention weights for the sequential position (row 2) dominates. 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 all three classification tasks, we can see the positional information playing important roles. For UI screen genre classification, spatial information (rows 4 to 6) appears to be important, which could indicate the importance of element spatial positions in a UI layout for this task. For UI screen topic classification, sequential and hierarchical information (rows 2 and 3) appear to play an important role. The importance of sequential information could be due to correlations between the position of a UI screen in user interactions and its topical role. Similarly, the hierarchical structure of the UI screen could also have strong relationships with its topical role. For the UI element component-type classification, hierarchical level information (row 3) appears to play a key role, which makes intuitive sense, since the depth of a UI element in a hierarchy is likely to be correlated to its component-type.

4.6 Discussion

Based on the results of our experiments, we see that HAMP'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. The distribution of genre, topic and component-type classes are imbalanced, and HAMP's relatively good performance on the macro F1 metric on the classification tasks (especially when compared with the baselines) demonstrates its ability 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 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. Further, from the visualization of the learnt attention weights shown in Figure 4, 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. 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

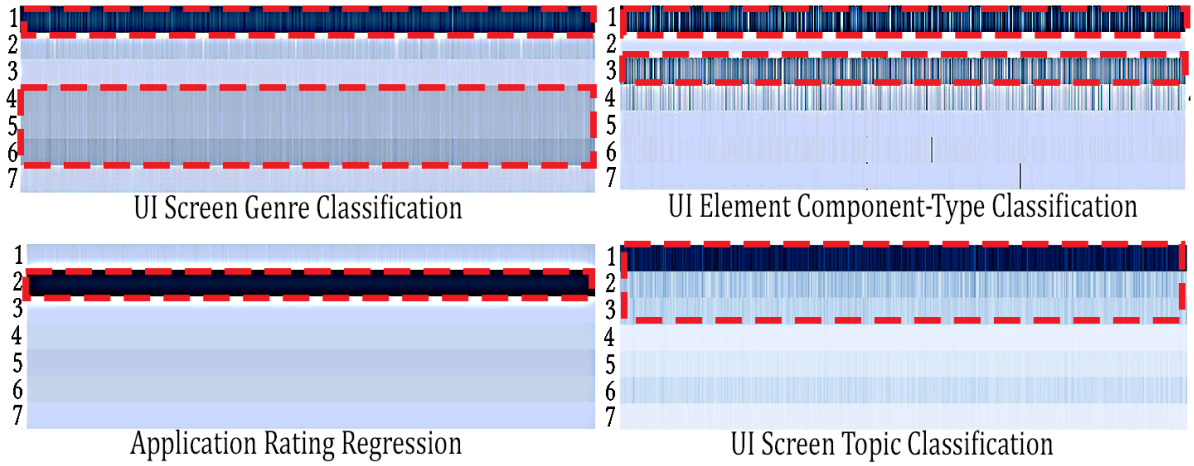

Figure 4: Learnt Attention Weights (RICO-N and ENRICO), i.e. β_m - Row 1 corresponds to the attention weights β_{ft} for the multimodal attributes (e.g., encoded UI screen image); while rows 2 to 7 correspond to the attention weights β_{pos} for the positional attributes. Specifically, row 2 corresponds to attention weights for sequential position; row 3 for hierarchy level; row 4 for spatial coordinate x0; row 5 for spatial coordinate y0; row 6 for spatial coordinate x1; and row 7 for spatial coordinate y1. Each column represents the attention weights of the relevant nodes for the respective tasks (e.g. UI screen nodes for genre classification). Darker shade of blue means higher attention weights for the nodes, e.g. for UI screen genre classification tasks, attention weights for UI screen nodes are highest for row 1 which are the multimodal attributes. Attributes with higher attention weights are highlighted in red boxes.

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 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 paper, and such networks and their features could be captured by the HAMP model. 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.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose HAMP, a novel scaled dot-product attention-based GNN message passing model designed for heterogeneous networks with multimodal and positional attributes. Through experiments involving an extensive set of state-of-the-art baseline models, we demonstrate that HAMP out-performs state-of-the-art models on a wide range of tasks for mobile application UIs that have real-world applications. Given HAMP's performance on the three distinct datasets that were used for experiments, the model is likely to be equally applicable to other UIs even if characteristics of the information differ. In addition to the tasks shown in Figure 2, future work could explore how HAMP could be used to complement or augment a range of other UI-related systems. HAMP could be incorporated within UI object detection and annotation systems in works such as [7, 19, 45, 50] to capture additional multi-modal and structural network information. HAMP could also be used to complement systems that assist UI designers [5, 6, 27, 35]. 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.

In relation to its **societal impact**, we see opportunities for HAMP to benefit a range of real-world applications due to its 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 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.

ACKNOWLEDGMENTS

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.

REFERENCES

[1] Gary Ang and Ee-Peng Lim. 2021. Learning Network-Based Multi-Modal Mobile User Interface Embeddings. In *6th International Conference on Intelligent User Interfaces IUI 2021, College Station, TX, USA, April 14–17, 2021.*

[2] 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. *CoRR* (2018).

[3] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu, and Magy Seif El-Nasr. 2021. VINS: Visual Search for Mobile User Interface Design. *CoRR* (2021).

[4] 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 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage AK USA.

[5] 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, Article 180 (2019).

[6] Chun-Fu (Richard) Chen et al. 2017. UI X-Ray: Interactive Mobile UI Testing Based on Computer Vision. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces.*

[7] 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 *28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering.*

[8] 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 30th Annual Symposium on User Interface Software and Technology.*

[9] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

[10] Vijay Prakash Dwivedi and Xavier Bresson. 2021. A Generalization of Transformer Networks to Graphs. *CoRR* (2021).

[11] Hongyang Gao and Shuiwang Ji. 2019. Graph Representation Learning via Hard and Channel-Wise Attention Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.*

[12] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. *CoRR* (2017).

[13] Luke B. Godfrey and Michael S. Gashler. 2018. Neural Decomposition of Time-Series Data for Effective Generalization. *IEEE Trans. Neural Networks Learn. Syst.* 29, 7 (2018), 2973–2985.

[14] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. DynGEM: Deep Embedding Method for Dynamic Graphs. *CoRR* (2018).

[15] Asnat Greenstein-Messica, Lior Rokach, and Michael Friedman. 2017. Session-Based Recommendations Using Item Embedding. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces.*

[16] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna R. Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational Graph Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems.*

[17] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems.*

[18] 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 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[19] 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 *Thirty-Fifth AAAI Conference on Artificial Intelligence.*

[20] Geoffrey E. Hinton. 2021. How to represent part-whole hierarchies in a neural network. *CoRR* (2021).

[21] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *Proceedings of the 2020 World Wide Web Conference on World Wide Web.*

[22] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. Swire: Sketch-based User Interface Retrieval. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems.*

[23] Wentao Huang, Yuchen Li, Yuan Fang, Ju Fan, and Hongxia Yang. 2020. BiANE: Bipartite Attributed Network Embedding. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[24] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations.*

[25] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. In *NIPS Workshop on Bayesian Deep Learning.*

[26] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations.*

[27] 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 2020 CHI Conference on Human Factors in Computing Systems.*

[28] Luis A. Leiva, Asutosh Hota, and Antti Oulasvirta. 2020. Enrico: A Dataset for Topic Modeling of Mobile UI Designs. In *MobileHCI 2020: 22nd International Conference on Human-Computer Interaction with Mobile Devices and Services.*

[29] Toby Jia-Jun Li et al. 2021. Screen2Vec: Semantic Embedding of GUI Screens and GUI Components. In *CHI '21: CHI Conference on Human Factors in Computing Systems.*

[30] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web.*

[31] Jingxin Liu, Chang Xu, Chang Yin, Weiqiang Wu, and You Song. 2020. K-Core based Temporal Graph Convolutional Network for Dynamic Graphs. *CoRR* (2020).

[32] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning Design Semantics for Mobile Apps. In *The 31st Annual ACM Symposium on User Interface Software and Technology*. Berlin, Germany.

[33] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-Embedding Attributed Networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining.*

[34] Zaiqiao Meng, Shangsong Liang, Jinyuan Fang, and Teng Xiao. 2019. Semi-supervisedly Co-embedding Attributed Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems.*

[35] Peter O'Donovan, A. Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with Interactive Layout Suggestions. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems.*

[36] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *8th International Conference on Learning Representations.*

[37] 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 *The Thirty-Fourth AAAI Conference on Artificial Intelligence.*

[38] Aneeshan Sain, Ayan Kumar Bhunia, Yongxin Yang, Tao Xiang, and Yi-Zhe Song. 2021. StyleMeUp: Towards Style-Agnostic Sketch-Based Image Retrieval. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

[39] 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 *The Semantic Web - 15th International Conference Proceedings.*

[40] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. *CoRR* (2017).

[41] 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 *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems.*

[42] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *6th International Conference on Learning Representations, Conference Track Proceedings.*

[43] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[44] 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 2019 World Wide Web Conference on World Wide Web.*

[45] Thomas D. White, Gordon Fraser, and Guy J. Brown. 2019. Improving random GUI testing with image-based widget detection. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2019.*

[46] 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 36th International Conference on Machine Learning.*

[47] Yingtao Xie, Tao Lin, and Hongyan Xu. 2019. User Interface Code Retrieval: A Novel Visual-Representation-Aware Approach. *IEEE Access* 7 (2019), 162756–162767.

[48] 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 35th International Conference on Machine Learning*.

[49] Shaowei Yao, Tianming Wang, and Xiaojun Wan. 2020. Heterogeneous Graph Transformer for Graph-to-Sequence Learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

[50] 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 *CHI Conference on Human Factors in Computing Systems*.