# Learning network-based multi-modal mobile user interface embeddings

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 network-based multi-modal mobile user interface embeddings. (2021). *Proceedings of the 26th Annual Conference on Intelligent User Interfaces, College Station, Texas, 2021 April 14-17*. 366-376.
**Available at:** https://ink.library.smu.edu.sg/sis_research/7049

# Learning Network-Based Multi-Modal Mobile User Interface Embeddings

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

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

## ABSTRACT

Rich multi-modal information - text, code, images, categorical and numerical data - co-exist in the user interface (UI) design of mobile applications. UI designs are composed of UI entities supporting different functions which together enable the application. To support effective search and recommendation applications over mobile UIs, we need to be able to learn UI representations that integrate latent semantics. In this paper, we propose a novel unsupervised model - Multi-modal Attention-based Attributed Network Embedding (MAAN) model. MAAN is designed to capture both multi-modal and structural network information. Based on the encoder-decoder framework, MAAN aims to learn UI representations that allow UI design reconstruction. The generated embedding can be applied to a variety of tasks: predicting UI elements associated with UI screens, inferring missing UI screen and element attributes, predicting UI user ratings, and retrieving UIs. Extensive experiments, including user evaluations, conducted on two datasets from RICO, a rich real-world mobile UI repository, demonstrates that MAAN out-performs other state-of-the-art models.

## CCS CONCEPTS

• **Computing methodologies → Neural networks**; • **Human-centered computing → User interface management systems**; • **Information systems → Multimedia and multimodal retrieval**.

## KEYWORDS

Network embedding, mobile application user interface, unsupervised retrieval, multi-modal

**ACM Reference Format:**
Gary Ang and Ee-Peng Lim. 2021. Learning Network-Based Multi-Modal Mobile User Interface Embeddings. In *26th International Conference on Intelligent User Interfaces (IUI '21), April 14–17, 2021, College Station, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3397481.3450693

## 1 INTRODUCTION

The increasing maturity of mobile application design/development practices means that a mobile UI designer/developer can easily refer to existing UI repositories as part of the development process. Such repositories include multi-modal information - network structures consisting of UI screens and UI elements of mobile applications with links between them, visual information from UI screen images, codes associated with UI elements, textual description of the associated applications, categorical application genres and application ratings. The RICO dataset[2, 19] is one such repository. It includes data from more than 9,000 Android applications. Such repositories are extremely valuable when we search relevant UI objects for design reference or for replacing some existing UI objects as part of the UI development process. However, manually sifting through UI objects in large repositories is not feasible. A naive retrieval system based on keywords or genre would not be adequate as such features are sparse and do not capture rich semantics in UI interface designs. Latent representations, also known as embeddings, can be developed to better capture underlying similarities. However, most approaches for generating embeddings only capture information from a single modality. Existing multimedia approaches to represent UI screen image also do not capture the non-Euclidean nature of network structures linking UI interface components together. Hence, an approach that can generate embeddings that capture information from multiple modalities and structural network information is needed.

State-of-the-art attributed network embedding models generate embeddings that capture structural network information, but still *do not capture information from multiple modalities and node types.* Figure 1 provides an overview of the UI dataset. The figure depicts two types of nodes, i.e. UI screen and UI elements. The multi-modal features of a UI screen include its image, description, and genre. Similarly, a UI element's features include its class name, and component type.

Figure 1 also shows our proposed multi-modal network embedding framework which includes a multi-modal attention-based attributed network embedding (MAAN) model to be used to learn the embeddings representations of UI screens and UI elements which can be used in several downstream tasks, e.g. predicting links between UI screens and elements for UI design reconstruction, rating predictions, missing attribute inference and UI screen retrieval. MAAN aims to incorporate several new requirements: 1) **capture correlations between information from different modalities**; 2) **ensure no one modality dominates the generated embedding**; and 3) **effectively balance between the increased number of training objectives**. Capturing correlations, or affinities, between information from multiple modalities (e.g. between visual styles and application genres, or visual structure and code classes) plays a critical role in generating informative embeddings. The dimensions of modalities could differ significantly

*IUI '21, April 14–17, 2021, College Station, TX, USA*
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8017-1/21/04...$15.00
https://doi.org/10.1145/3397481.3450693

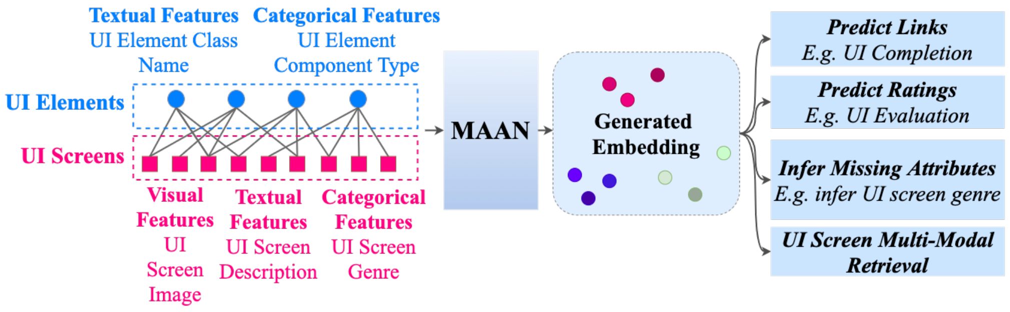

**Figure 1: UI dataset in a network embedding framework**

(e.g. number of genres vs. number of words) and simply encoding them together could cause the generated embedding to focus only on one or two modes. The increased number of modalities also means additional training objectives, and a need to balance between these objectives that could diverge. There have been other works that also take a network-based approach or used latent representations/embeddings for tasks relating to user interactions[3, 7, 31, 32], but *the use of network embedding models for such mobile UI development related tasks has thus far not been explored.*

This paper proposes as part of MAAN a **novel graph variational autoencoder (GVAE)-based model to generate multimodal embeddings of UI screen and element nodes.** Specifically, our GVAE **jointly embeds information from multiple modalities**; uses an **attention mechanism to discover correlations between different modalities and weight their contribution**; and **improves existing loss functions used for graph variational autoencoders** to learn more meaningful embeddings. Our key contributions are as follows:

- To our knowledge, this is the first work to propose a multimodal network embedding model that combines attention mechanisms with the variational autoencoding framework to generate embeddings for link prediction, attribute inference, regression and retrieval tasks relating to mobile UIs.
- We propose a two-stage encoding process to ensure no one modality dominates the generated embedding.
- We use the attention mechanism to capture relationships between information from different modalities, which is different from past works[12] that use attribute-to-attribute matrices.
- The use of the Maximum Mean Discrepancy (MMD)[35] term to more effectively balance between multiple training objectives has not featured in existing GVAEs.
- We show that MAAN consistently out-performs other state-of-the-art models on two datasets across an extensive set of tasks. User evaluations on retrieval results show that MAAN out-performs the other models by as much as eight times.

## 2 RELATED WORK

Key related works in the areas of UI retrieval and evaluation, and network embedding models are outlined in this section.

### 2.1 UI retrieval and evaluation

Swire[11] is a system that retrieves UI screen images based on UI sketches. Both UI sketches and UI screenshots are represented as image embeddings for visual similarity comparison. Swire however only captures information from a single modality. [34] converts UIs into sequences to support retrieval applications. This approach only captures structural network information of UI objects. [17] developed a design assistant that uses measures of visual complexity[24] to help assess visual complexity of UI screens. This work focuses on the ranking of UI screens instead of their embedding representations.

### 2.2 Network embedding models

As UI screens and elements are inter-connected in a large network, we thus review network embedding models.

*2.2.1 Homogeneous Networks.* DeepWalk [23] and Node2Vec [8] extend the idea of word embeddings to networks by treating paths as sentences and nodes as words. They however do not utilize the attributes of nodes when learning network embeddings. Graph neural networks (GNN) learn node embeddings by propagating the attributes of nodes repeatedly or over multiple neural network layers via a message passing framework[6]. Graph Convolutional Network (GCN)[16] aggregates features of neighboring nodes and normalizes the aggregated representations by the node degrees. GraphSAGE[9] further considers mean, LSTM or pooling aggregation methods. While GCN uses the full neighborhood, GraphSAGE samples a fixed number of neighbors. In the Graph Attention Network (GAT)[28], different nodes in the neighborhood are assigned different importances during aggregation. Messages passed between each layer in most GNNs go through non-linear layers such as rectified linear activation units (ReLU). [33] finds that similar performance can be achieved even without such layers. The Graph Variational Autoencoder(GVAE)[15] applies the variational autoencoder (VAE)[14] framework to learn the node embeddings of homogeneous networks. The Co-Embedding Attributed Network(CAN)[21] uses two VAE channels to jointly encode and decode the adjacency matrix and a single feature matrix to jointly learn the representations of both nodes and attributes. Semi-supervised Co-embedding Attributed Network (SCAN) [22] extends CAN to co-embed both attributes and nodes of partially labelled networks.

*2.2.2 Heterogeneous Networks.* Such models have also been applied to heterogeneous networks (networks with multiple node or edge-types). Relational Graph Convolutional Networks[25] and Graph Convolutional Matrix Completion [26] use multiple GCNs to encode embeddings of multiple adjacency matrices, one for each edge type, before aggregating them. Neural Graph Collaborative Filtering[29] and LightGCN[10] encode embeddings for different number of hops before aggregating them. [13] captures indirect proximity between the same node types in bipartite networks. Multinomial VAE[18] is a VAE-based approach that generates embeddings by using a multinomial distribution instead of the Bernoulli distribution used in GVAE. Heterogeneous Graph Attention Network[30] and General Attributed Multiplex Heterogeneous Network[1] use multiple GNN-based layers to encode networks formed from different metapaths[4] before using an attention mechanism to aggregate the embeddings.

While the related works outlined in this section only use information from a single modality, the proposed model in this paper uses a VAE-based approach to encode information from multiple modalities. A two step encoding process is used so that no one modality dominates the generated embedding. Further, we introduce an attention mechanism to capture the correlations between different modalities and allow the underlying importance of each modality to be self-discovered. Instead of just applying the generated embeddings to common link prediction and attribute inference tasks, we use the generated embeddings to predict continuous labels (application ratings) and for multi-modal retrieval of the UI screens.

## 3 MULTI-MODAL ATTENTION-BASED ATTRIBUTED NETWORK EMBEDDING

To cope with both multi-modal features and the heterogeneity of structural network information in UI designs, we propose a new embedding model known as **Multi-modal Attention-based Attributed Network Embedding (MAAN).** It assumes that UI designs can be represented by a bipartite graph $G = (V, E, X)$ constructed from a UI design dataset (e.g. RICO), where $V$ comprises two disjoint sets of nodes $U$ and $W$. $U$ represents UI screen nodes and $W$ represents UI element nodes. Edges only exist between different node-types, i.e. $E \subseteq U \times W$. $e_{ij}$ represents the edge between $v_{U,i}$ and $v_{W,j}$. $X$ comprises multi-modal features, $P$ of these attributes are for UI screens and $Q$ of them are for UI elements, denoted by $\{X_{U,1}, ..., X_{U,P}\}$, and $\{X_{W,1}, ..., X_{W,Q}\}$ respectively. MAAN will jointly embed the nodes and their multi-modal features as low-dimensional vectors by learning the mapping functions $f : V \mapsto Z \in \mathbb{R}^{|V| \times d}$ for the nodes, and $g : X_{V,r} \mapsto Z_r \in \mathbb{R}^{F_r \times d}$ for features in each modality where $d \ll |V|$.

We first construct a $|V| \times |V|$ heterogeneous adjacency matrix $A$ from the network edges $E$:

$$A = \begin{bmatrix} 0 & M \\ M^T & 0 \end{bmatrix}$$

where $M \in [0, 1]^{|U| \times |W|}$ is the UI screen to UI element adjacency matrix.

We also construct the multi-modal feature matrices $\{X_{U,1}, ..., X_{U,P}\}$, $\{X_{W,1}, ..., X_{W,Q}\}$. Each feature matrix $X_{V,r} \in \mathbb{R}^{|V| \times F_r}$ for modalities $r \in \{p, q\}$ of nodes $V \in \{U, W\}$ contains attribute information for all $|V|$ nodes. Where a feature does not apply to the node, e.g. feature $p$ for a node $v$ in $W$, the feature matrix $X_{U,p}$ will have the row corresponding to node $v$ filled with zeros. MAAN uses one VAE channel to encode/decode the adjacency matrix; and one VAE channel to encode/decode every feature matrix. Hence, it requires $P + Q + 1$ VAE channels. VAE is chosen because of its expressiveness, generative ability and flexibility to be combined with other deep learning models. Figure 2 depicts the different channels in the MAAN architecture, which consists of a multi-modal attention-based GAE and several feature VAEs.

### 3.1 Multi-modal Attention-based GAE

This module generates the node embeddings. Multiple GATs encode node features from different modalities, which are then fused with an attention mechanism. The fused node features are encoded with GCNs into Gaussian embeddings, and an inner product decoder used to reconstruct the adjacency matrix by sampling from the learned Gaussian embeddings. The module first takes as inputs the adjacency matrix $A$ and the node features $\{X_{U,1}, ..., X_{U,P}\}$, $\{X_{W,1}, ..., X_{W,Q}\}$. It uses $P + Q$ GAT modules to separately transform a node's features of different modalities to hidden representations. Encoding each modality with separate GAT modules before combining them avoids any modality dominating the final embedding, and allows the model to accommodate any number of modalities. For each layer in a GAT module, we first take as input the node feature of node $i$ and apply a linear transformation $s^{(l)}_{i,r} = W^{(l)} x^{(l)}_{i,r}$ to each node where $W^{(l)}$ is the learnable weight matrix of the $l^{\text{th}}$ layer, and $x^{(l)}_{i,r}$ is either the original feature or the hidden representation of the node $i$ from an earlier layer. We then compute the pair-wise un-normalized attention score between node $i$ and each of its neighbors, say node $j$. We concatenate the hidden representations of nodes $i$ and $j$ before taking a dot product of it with a learnable weight vector $a^{(l)T}$. A LeakyReLU activation is then applied - $e^{(l)}_{ij} = LeakyReLU(a^{(l)T}(s^{(l)}_{i,r} || s^{(l)}_{j,r}))$. Softmax is then used to compute the attention scores to weight the hidden representations received by node $i$ from its neighboring nodes $N(i)$ -

$$\alpha^{(l)}_{ij} = \frac{exp(e^{(l)}_{ij})}{\sum_{k \in N(i)} exp(e^{(l)}_{ik})}$$

The final step in each GAT layer aggregates the hidden representations received by node $i$ from its neighbors $N(i)$, weighted by the attention scores.

$$x^{(l+1)}_{i,r} = \sigma(\sum_{j \in N(i)} \alpha^{(l)}_{ij} s^{(l)}_{j,r})$$

The steps outlined above are repeated for each layer in the GAT module with node features from each modality, resulting in each node having $P + Q$ separate sets of intermediate hidden representations $X'_{V,1}, .., X'_{V,R} \in \mathbb{R}^{|V| \times d'}$ for modalities $R \in \{P, Q\}$ of nodes $V \in \{U, W\}$, where $d' > d$. Setting the dimensions of intermediate node representations to be higher than the final latent representation of nodes allows for a more gradual compression of the embedding space.

Next, we use an attention mechanism to fuse representations from different modalities into a $|V| \times d''$ representation matrix

**Table 1: Summary of Key Notations**

| Symbol | Description |
|---|---|
| $V$ | Nodes in graph $G$ comprising disjoint sets of nodes $U$ and $W$ with $P$ and $Q$ multi-modal features respectively |
| $F_r$ | Feature dimension for the rth modality where $r \in \{p, q\}$ |
| $d, d', d''$ | Dimensions of final and intermediate representations of nodes and features |
| $X_{V,1}, .., X_{V,r} \in \mathbb{R}^{\lvert V \rvert \times F_r}$ | Multi-modal features $r \in \{p, q\}$ of nodes $V \in \{U, W\}$ |
| $X'_{V,1}, .., X'_{V,r} \in \mathbb{R}^{\lvert V \rvert \times d'}$ | Hidden multi-modal representations of nodes after GAT modules |
| $X''_V \in \mathbb{R}^{\lvert V \rvert \times d''}$ | Hidden node representations after multi-modal attention fusion |
| $Z \in \mathbb{R}^{\lvert V \rvert \times d}$ | Final latent representation/embedding of nodes |
| $H_r \in \mathbb{R}^{F_r \times d'}$ | Hidden representation of feature from the rth modality where $r \in \{p, q\}$ |
| $Z_r \in \mathbb{R}^{F_r \times d}$ | Final latent representation/embedding of feature from the rth modality where $r \in \{p, q\}$ |

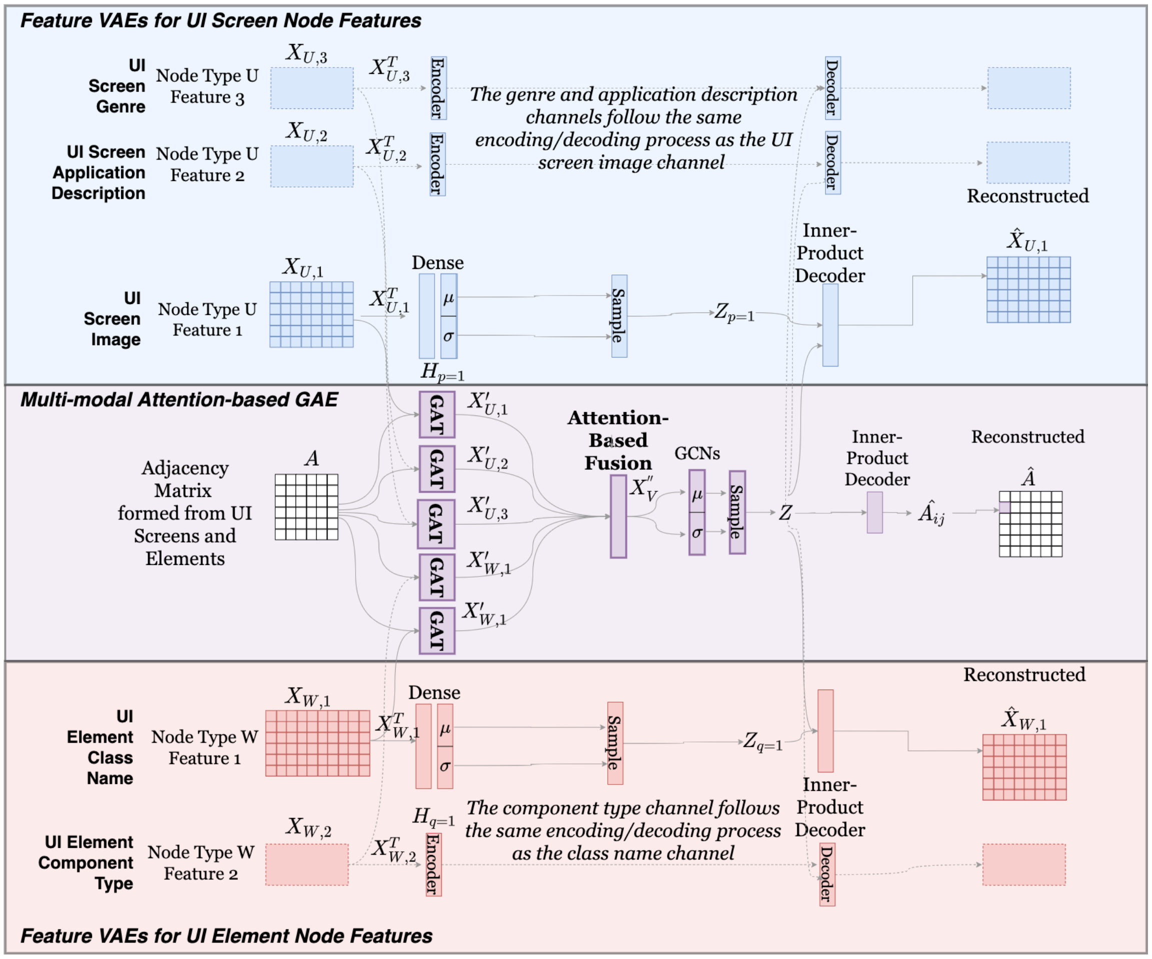

**Figure 2: MAAN Architecture for RICO with P=3 VAE channels for UI screen nodes and Q=2 VAE channels for UI element nodes**

$X''_V$. The use of the attention mechanism here allows the model to self-discover correlations between different modalities, and weight contributions accordingly.

We first apply a non-linear transformation to each of these representations for each node to obtain a scalar $k_{i,r}$ for each modality, which represents the importance of each modality.

$$k_{i,r} = W^{(1)}_{att} tanh(W^{(0)}_{att} x'_{i,r} + b)$$

where $x'_{i,r}$ is the intermediate hidden representation of the node for the $r^{th}$ modality, $W^{(0)}_{att}$ and $W^{(1)}_{att}$ are learnable weight matrices and $b$ is the bias vector. The parameters are shared across all modalities. We then normalize $k_{i,r}$ with a softmax function to obtain the weights:

$$\beta_{i,r} = \frac{exp(k_{i,r})}{\sum_{r=1}^{R} exp(k_{i,r})}$$

where $r \in \{p, q\}$ and $R \in \{P, Q\}$.

Finally, we use these weights to fuse the multiple representations for each node: $x''_i = \sum_{p=1}^{P} \beta_{i,p} x'_{i,p} + \sum_{q=1}^{Q} \beta_{i,q} x'_{i,q}$.

At this point, we denote the resultant aggregated hidden representations for all nodes as $X''_V \in \mathbb{R}^{|V| \times d''}$ where $d'' \leq d'$ as we progressively compress the dimension of the node representations. Two layers of GCNs are then used to encode the aggregated hidden representations $X''_V$ into the Gaussian embeddings. The use of stochastic Gaussian embeddings allows for more expressive embeddings to be generated.

$$X_V''^{(1)} = ReLU(\tilde{A} X''_V W_V^{(0)})$$

$$[\mu_V, \sigma_V^2] = \tilde{A} X_V''^{(1)} W_V^{(1)}$$

where $\tilde{A} = D^{-0.5} A D^{-0.5}$ is the symmetrically normalized adjacency matrix with $D$ as the adjacency matrix's degree matrix, $\mu_V$ and $\sigma_V^2$ are the means and variances of the learned Gaussian embeddings of the nodes, and $W_V^{(0)}$, $W_V^{(1)}$ the learnable weight matrices.

The final embedding of each node can then be computed as $Z_i = \mu_{V,i} + \sigma_{V,i}^2 \odot \epsilon$ where $\epsilon$ is a random variable sampled from a standard normal distribution. To reconstruct the adjacency matrix, we first apply an inner product between the embeddings of node $i$ and node $j$ to obtain:

$$[\mu_{E_{i,j}}, \sigma_{E_{i,j}}^2] = sigmoid(Z_i^T Z_j)$$

As the edges are binary, the edge between nodes $i$ and $j$ can be reconstructed with:

$$p_\theta(A_{ij} | Z_i, Z_j) = Ber(\mu_{E_{i,j}})$$

where $Ber(\mu_{E_{i,j}})$ is the Bernoulli distribution parameterized by $\mu_{E_{i,j}}$.

## 3.2 Feature VAE

In MAAN, we use feature VAEs to encode node features of each modality. The feature VAE adopts stochastic Gaussian embeddings to allow for more expressive embeddings. Two linear layers with a tanh activation are first used to infer the Gaussian embeddings:

$$H_r = tanh(X_{V,r}^T W_r^{(0)} + b^{(0)})$$

$$[\mu_r, \sigma_r^2] = H_r W_r^{(1)} + b^{(1)}$$

where $\mu_r$ and $\sigma_r^2$ are the means and variances of the learned Gaussian embeddings of the features, and $W_r^{(0)}$ and $W_r^{(1)}$ are learnable weight matrices. The embedding of each feature can then be computed as $Z_{r,a} = \mu_{r,a} + \sigma_{r,a}^2 \odot \epsilon$ where $a$ is the index of the feature, e.g. index of *Social* genre for *UI screen genre modality*, and $\epsilon$ is a random variable sampled from a standard normal distribution. We then reconstruct the feature matrix by first applying an inner product between the embedding of the nodes and features

$$[\mu_{E_{r,i,a}}, \sigma_{E_{r,i,a}}^2] = sigmoid(Z_i^T Z_{r,a})$$

The node to feature edges for feature matrices with continuous values can be reconstructed with:

$$p_\theta(X_{r,i,a} | Z_i, Z_{r,a}) = \mathcal{N}(\mu_{E_{r,i,a}}, \sigma_{E_{r,i,a}}^2 I)$$

where $\mathcal{N}(\mu_{E_{r,i,a}}, \sigma_{E_{r,i,a}}^2 I)$ is the Gaussian distribution parameterized by $\mu_{E_{r,i,a}}$ and $\sigma_{E_{r,i,a}}^2$. The node to feature edges for feature matrices with binary values are reconstructed with:

$$p_\theta(X_{r,i,a} | Z_i, Z_{r,a}) = Ber(\mu_{E_{r,i,a}})$$

where $Ber(\mu_{E_{r,i,a}})$ is the Bernoulli distribution parameterized by $\mu_{E_{r,i,a}}$.

## 3.3 Objective Functions

Under the VAE framework, we want $q_\phi(z)$ for the encoders in the MAAN model to match the prior distribution $p_\theta(z)$ for the decoders in the MAAN model and the evidence lower bound objective (ELBO) is - $\mathcal{L}_{ELBO}(x) = \mathbb{E}_{q_{\phi(z|x)}}[log p_\theta(x|z)] - KL(q_\phi(z|x) || p(z))$. The first term measures the reconstruction loss while the second term is the Kullback-Leibler (KL) divergence. During training, the reparameterization trick is used to rewrite the expectation with respect to $q_\phi(z|x)$ such that the Monte Carlo estimate of the expectation is differentiable with respect to $\phi$[14]. This VAE set-up is known to suffer from over-fitting when the KL divergence term is not strong enough or to generate uninformative embeddings when the KL divergence term is too restrictive. We use the MMD term[35] instead of the KL divergence term for the feature VAEs. We do not use $\alpha/\lambda$ hyper-parameters to balance between the KL divergence and MMD terms as proposed in [35] as tuning these hyper-parameters did not improve performance significantly. The use of the MMD term improves training stability and enhances the model's ability to learn meaningful embeddings from multiple modalities. The resulting objective function which we use in our model for the multi-modal attention-based GAE is:

$$\begin{aligned} \mathcal{L}_{ELBO}(x) &= \mathcal{L}_{reconstruction} + \mathcal{L}_{KL} \\ &= \mathbb{E}_{q_{\phi(z|x)}}[log p_\theta(x|z)] - KL(q_\phi(z|x) || p(z)) \end{aligned}$$

The resulting objective function which we use in our model for the feature VAEs is:

$$\begin{aligned} \mathcal{L}_{ELBO}(x) &= \mathcal{L}_{reconstruction} + \mathcal{L}_{MMD} \\ &= \mathbb{E}_{q_{\phi(z|x)}}[log p_\theta(x|z)] + MMD(q_\phi(z) || p(z)) \end{aligned}$$

We use the binary cross-entropy loss for $\mathcal{L}_{reconstruction}$ in the multi-modal attention-based GAE. For the feature VAEs, we use the binary cross-entropy loss when reconstructing a feature matrix of binary values, and mean square error loss when reconstructing a feature matrix of continuous values.

## 4 EXPERIMENTS

In this section, we conduct several experiments to evaluate MAAN against other baselines to determine the importance of representation learning using multi-modal information. The evaluation is conducted using a range of prediction and retrieval tasks. We finally examine the use of attention mechanism to explain the focus of the generated embeddings.

## 4.1 Datasets

The two datasets used in the experiments are extracted from **RICO**, a repository covering the mobile UIs of 9,384 Android applications. We scraped updated metadata of these applications from Google Play Store and filtered out applications and UI screens still available for download in Feb 2020. This leaves us 6,583 applications, released from Jan 10 to Apr 17. The repository includes UI screenshots and their UI elements. To ensure that the experiment findings are valid for UI datasets with different characteristics, we extract two datasets from the RICO repository, namely: (i) RICO-N, comprising the most recently released 1000 applications (Oct 15 to Apr 17); and (ii) RICO-O, comprising the earliest released 1000 applications (Jan 10 to Aug 11). The differences between the two datasets are significant, as shown in Table 2. RICO-O is about twice the size of RICO-N by number of UI screens, UI elements and edges. The UI elements of both datasets have different dimensions for multi-hot class name vectors. The distributions of the genres in the two datasets are also different. The most popular genre in RICO-N is Social, whereas News and Magazines genre is most popular in RICO-O. RICO-O and RICO-N share 33 common genres between their UI screens, and RICO-N has three additional genres. We next extract UI elements occurring within each UI screen from the view hierarchies. A high number of occurrences of a UI element in a UI screen suggests that the UI element is important to the UI screen. On the other hand, we observe that a low number of occurrences are often boilerplate elements used to initialize the UI structure. Hence, we include an edge between a pair of UI screen and UI element only if the UI element occurs more than five times in the UI screen. Instead of this empirical approach, we can consider a heuristic or model-based approach that can discriminate between the informativeness of different UI elements but would include this as part of future work.

Visual information is then encoded with a separate autoencoder. Textual information is encoded with pre-trained Glove embeddings. We pre-process the class names of the UI elements 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 class name features by: 1) using a pre-trained Char-NGram (CHAR) embedding; 2) using bag-of-words (BOW) representation. UI screen ratings are based on the ratings of the applications that they are associated with. Edge-lists are then divided into training, validation and testing datasets in the ratios 85%/5%/10% for the link prediction task. Under the co-embedding framework, we use the same generated embedding to evaluate both link prediction and attribute inference tasks. Hence, nodes that are part of the edges in the validation and testing edge-lists are used as validation/testing datasets for the attribute inference task, and their attribute values set to zero in the training dataset. For the rating prediction task, UI screen ratings are divided into training and testing datasets in the ratio 80%/20%. We do not utilize validation data for the rating prediction task as we simply use a simple series of dense layers for regression with the generated embeddings and do not tune any hyper-parameters.

## 4.2 Experiment Setup

We compare the performance of MAAN with other state-of-the-art models on tasks that could facilitate the mobile UI design/development process. Each task is evaluated with separate evaluation metrics:

- **Link prediction** - This refers to the prediction of UI screens to be associated to each UI element via unobserved links. Each model returns the top ranked (or similar) UI screens for each UI element using the embeddings of the UI screens. To train and evaluate a prediction model with both positive and negative links, we randomly generate 10 pairs of nodes not in $E$ for each positive link $(v_{1i}, v_{2j})$ as negative links. For each model, we use the inner product of embeddings to predict the probability of a link forming between a pair of UI screen and UI element nodes. The model then ranks the candidate links in the test data by the inner product score. Area under the receiver operating characteristic curve (AUC) and average precision (AP) are used as the evaluation metrics.
- **Prediction of UI screen ratings** - We predict UI screen ratings for the RICO dataset by using generated node embeddings as inputs to a regression model implemented with a series of dense layers. This task is useful for predicting the success of new applications. We use root mean square error (RMSE) as the evaluation metric.
- **Attribute inference** - We infer missing binary and continuous valued attributes of UI screens and elements based on affinities between nodes and attributes. For binary valued attributes, we treat this as predicting links between nodes and attributes, and hence also sample 10 negative instances for each positive instance as per the link prediction task. We compare MAAN with CAN as GAT is not designed for attribute inference. The inner-product of generated node embeddings and attribute/feature embeddings is used to infer missing attributes. For binary attribute inference which involves UI screen genre, UI element class name BOW, and UI element component type, we measure the performance by AUC and AP. For UI element class name inference, we also split the class name into name tokens and compute the multi-label weighted F1 score to assess the performance of the model in inferring the class name-level attribute (and not just token-level attributes). The multi-label weighted F1 score returns the average score of all tokens in the class name weighted by support (number of true instances for each token). For continuous attribute inference which involves UI screen image, UI screen description, and UI element class name CHAR, RMSE is used.
- **Retrieval** - This facilitates the retrieval of UI screens for an input UI screen query. The relevant results should be visually similar UI screens considering their multiple modalities. In this task, we use human judgement to determine the the relevant UI screen results. We use Average Precision@5 (AP@5) and Normalized Discounted Cumulative Gain (NDCG) to evaluate the retrieval accuracy. More details on the computation of these metrics are provided in Section 4.6

**Baselines and Settings.** Co-embedding Attributed Network model (CAN) [21] and Graph Attention Network (GAT) [28] are chosen as baselines as they represent VAE and GNN-based state-of-the-art models respectively. We add a linear feature auto-encoder channel to GAT for it to co-embed representations of both attributes and nodes. For MAAN, we use four single-headed layers for each GAT module, and two layers for the GCN module. Feature VAEs use dense layers as encoders. We use the radial basis function as the kernel for MMD loss. Based on experiments with the validation

**Table 2: Dataset Statistics**

| | RICO-N | RICO-O |
|---|---|---|
| UI Screens/Elements | 5879/1563 | 9108/2920 |
| Number of Edges | 10762 | 19418 |
| Number of UI elements for each UI screen (Number of UI screens for each UI element) | | |
| Average | 1.9 (17.6) | 1.9 (16.7) |
| Maximum | 6.0 (1004.0) | 7.0 (1711.0) |
| Minimum | 1.0 (1.0) | 1.0 (1.0) |
| Attribute Types and Dimensions | | |
| **Visual** - Latent vectors of UI screen images *(Continuous)* | 64 | 64 |
| **Textual** - Glove vectors of app. descriptions for UI screen *(Continuous)* | 50 | 50 |
| **Categorical** - Multi-hot vectors of app. genre for UI screen *(Binary)* | 36 | 33 |
| **Textual** - Multi-hot/CharNGram vectors of class name of UI element *(Binary/Continuous)* | 1548/100 | 2161/100 |
| **Categorical** - Multi-hot vectors of UI element component type *(Binary)* | 25 | 25 |

dataset, the dimensions of the hidden node and feature representations $d'$ and $d''$ are fixed at 64, and the dimension of the final node and feature embeddings $d$ is fixed at 32. Training is run for 1000 epochs to generate the embeddings. The regression model for prediction of UI screen ratings is trained for 5000 epochs. For all models, an Adam optimizer with a learning rate of 0.02 with a cosine annealing scheduler is used and dropout is set to 0.1. All models are implemented with Pytorch.

## 4.3 Link Prediction Results

Table 3 shows the results of the experiments relating to link prediction. MAAN effectively utilizes multi-modal attributes to generate embeddings that consistently outperform baseline models on the link prediction task. CAN and GAT have greater difficulty utilizing continuous valued attributes to generate embeddings that perform well on the link prediction task, compared to when binary valued attributes are used. Differences in GAT performance when using UI element class names as attributes across the two datasets are likely due to differences in such attributes across the two datasets, e.g. differences in dimensions. VAE-based MAAN and CAN are less affected by such differences.

## 4.4 UI Screen Rating Prediction Results

Table 3 also sets out the results of regression experiments. Here, one of the MAAN models returns the best results for both datasets. The differences in performance across all models for the UI screen rating prediction task is more narrow compared with other tasks. A potential reason for this could be that there are very little differences in the strength of the relationships between the different attributes and UI screen ratings.

## 4.5 Attribute Inference Results

Table 4 sets out the results of the experiments relating to attribute inference. The embeddings generated by MAAN consistently outperforms embeddings generated by the baseline models across all modalities on this task. The differences in performance between MAAN and CAN using continuous valued attributes and binary valued attributes are even more stark here compared with the link prediction task. In some cases (e.g. screen description), the RMSE for the CAN model is around five times worse than the MAAN model.

## 4.6 Retrieval Evaluation Results

## 4.7 Qualitative Analysis

We used an AMT platform to recruit participants for evaluating the relevance of retrieval results. In this task, we use RICO-N dataset only and compare with five baseline models. The first four baseline models are the CAN and GAT models that utilize UI screen image and element component type attributes as they are likely to return better retrieval results. The fifth baseline model is a state-of-the-art metric learning-based multimedia retrieval model - VSE++ [5]. We choose VSE++ as it is also an unsupervised model, and allows us to compare MAAN with a model that is able to utilize information from two modalities. Figure 3 provides an overview of the user evaluation process. To generate the user evaluation surveys, we use 20 UI screen images as queries to retrieve the top five results (based on nearest neighbors) from MAAN and the five chosen baseline models described above. For each query UI screen, we construct five survey tasks. Each survey task shows the same query UI screen and 6 result UI screens, one result UI screen from MAAN and each of the five baseline models. For quality assurance, we also add either the query UI screen or a non-existent UI screen to each survey task. Each survey is then assigned to three AMT workers. Workers are asked to compare the query with randomly ordered results from MAAN and the baseline models and select UI screens that are most similar. If a worker fails to select the ground truth duplicate query UI screen image or selects the non-existent UI screen, the worker's responses for the entire survey will be discarded. The survey process is repeated until we obtain three good quality responses for each survey. Two metrics are used to measure retrieval performance. For **precision@5**, a result is deemed relevant if two or more workers select it as relevant. As workers can select multiple UI screens for each query UI screen image, a visual check is also conducted to assess the reasonableness of their responses, and we compute the **NDCG** metric with the actual number of workers selecting the result as relevant. Table 5 sets out the retrieval results. MAAN outperforms all baseline models by a significant margin. CAN using the binary valued UI element component type attribute performs better than the other baseline models, in line with what we had observed for other tasks. VSE++, even though it uses two modalities, is the worst performer. This could be due to VSE++ not capturing network structural information. This highlights the importance of capturing network structural information.

**Table 3: Link Prediction and Regression Results. Best and second best performing models per metric marked in boldface and underlined, where applicable, for this and subsequent tables on experimental results. For regression - RICO UI screen rating ranges from 1 to 5; RMSE using mean of training dataset is 0.583 and 0.554 for RICO-N and RICO-O respectively. For all tables: Comp. refers to component, Desc. refers to description. Higher (lower) values indicate better performance for AUC/AP (RMSE).**

| | Link Prediction | | | | Regression | |
|---|---|---|---|---|---|---|
| | RICO-N | | RICO-O | | RICO-N | RICO-O |
| Model | AUC | AP | AUC | AP | RMSE | RMSE |
| GAT(UI Screen Image) | 0.686 | 0.122 | 0.761 | 0.156 | 0.562 | 0.517 |
| GAT(UI Screen Desc.) | 0.680 | 0.122 | 0.729 | 0.141 | 0.552 | 0.522 |
| GAT(UI Screen Genre) | 0.939 | 0.843 | 0.928 | 0.658 | 0.682 | 0.536 |
| GAT(UI Element Class Name CHAR) | 0.379 | 0.068 | 0.861 | 0.716 | 0.555 | 0.520 |
| GAT(UI Element Class Name BOW) | 0.335 | 0.065 | 0.958 | 0.798 | 0.579 | 0.541 |
| GAT(UI Element Comp. Type) | 0.934 | 0.730 | 0.973 | 0.829 | 0.604 | 0.536 |
| CAN(UI Screen Image) | 0.972 | 0.881 | 0.960 | 0.828 | 0.617 | 0.534 |
| CAN(UI Screen Desc.) | 0.545 | 0.206 | 0.903 | 0.807 | 0.654 | 0.522 |
| CAN(UI Screen Genre) | 0.967 | 0.912 | 0.902 | 0.799 | 0.636 | 0.555 |
| CAN(UI Element Class Name CHAR) | 0.502 | 0.093 | 0.486 | 0.086 | 0.682 | 0.531 |
| CAN(UI Element Class Name BOW) | 0.965 | 0.856 | 0.960 | 0.842 | 0.593 | 0.542 |
| CAN(UI Element Comp. Type) | 0.963 | 0.829 | 0.938 | 0.866 | 0.638 | 0.533 |
| MAAN(All Features Class Name CHAR) | **0.981** | **0.913** | **0.994** | **0.957** | **0.550** | **0.511** |
| MAAN(All Features Class Name BOW) | 0.977 | 0.905 | 0.981 | 0.917 | 0.585 | 0.531 |

**Table 4: Attribute Inference. Weighted multi-label F1 score for UI Element Class Name in brackets. Higher (lower) values indicate better performance for AUC/AP/F1 score (RMSE).**

| | AUC/AP for binary valued attributes | | | RMSE for continuous valued attributes | | |
|---|---|---|---|---|---|---|
| Model | Screen Genre | Element Class Name *BOW* | Element Comp. Type | Screen Image | Screen Desc. | Element Class Name *CHAR* |
| | | | RICO-N | | | |
| CAN | 0.501/0.091 | 0.817/0.571 (0.170) | 0.733/0.198 | 1.451 | 8.958 | 5.685 |
| MAAN | **0.776/0.223** | **0.922/0.663 (0.170)** | **0.928/0.541** | **0.994** | **1.675** | **1.107** |
| | | | RICO-O | | | |
| CAN | 0.665/0.148 | 0.929/0.720 (0.196) | 0.500/0.090 | 2.454 | 12.032 | 5.806 |
| MAAN | **0.800/0.231** | **0.959/0.784 (0.196)** | **0.958/0.636** | **1.097** | **2.285** | **1.427** |

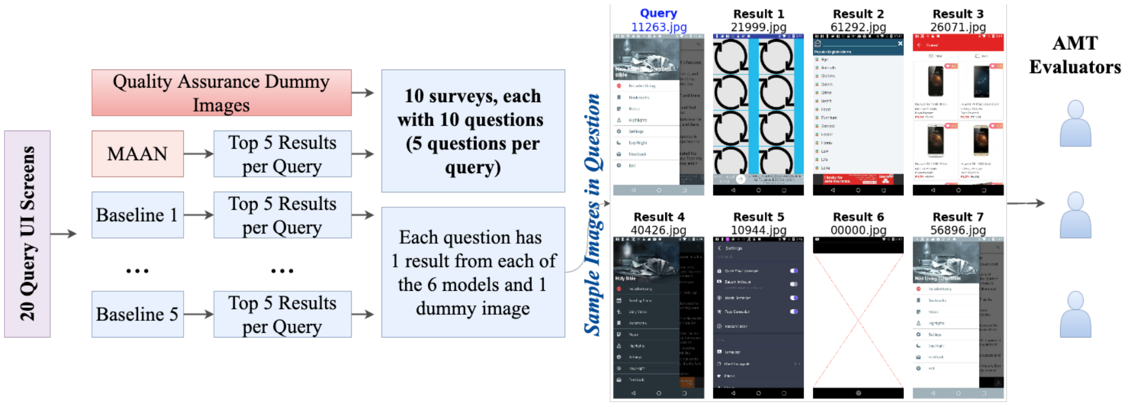

**Figure 3: Retrieval evaluation and sample of retrieval results shown to AMT workers. Results are randomly ordered. Result 6 is a dummy non-existent UI screen. Result 7 is result retrieved by MAAN and is not identical to query as they are menus for different bible versions.**

*4.7.1* ***Ablation Studies***. Tables 6 and 7 set out the results of the ablation studies for different model configurations of MAAN using the RICO-N dataset. The choices made in the proposed MAAN lead to better link prediction and regression performance. For attribute inference, multi-head can improve performance for some attributes. [27] found that multi-head attention allows information from different representation sub-spaces at different positions to be attended to. Here, multi-head allows the model to jointly attend to information from different nodes. Tables 8 and 9 set out the results of ablation studies for different number of modalities. MAAN-2

**Table 5: Retrieval Evaluation - RICO-N. 42.0% of all results selected as relevant by 2 or more AMT workers. Higher values indicate better performance.**

| Model | Precision @5 | NDCG |
|---|---|---|
| GAT(UI Screen Image) | 25.0% | 58.0% |
| GAT(UI Element Comp. Type) | 25.0% | 58.0% |
| CAN(UI Screen Image) | 19.0% | 60.4% |
| CAN(UI Element Comp. Type) | <u>52.0%</u> | <u>82.2%</u> |
| VSE++(UI Screen Image and UI Element Comp. Type) | 11.0% | 55.2% |
| MAAN | **85.0%** | **93.7%** |

**Table 6: Model Ablation - Link Prediction/Regression. Higher (lower) values indicate better performance for AUC/AP (RMSE).**

| | Link Prediction | | Regression |
|---|---|---|---|
| Modification | AUC | AP | RMSE |
| Replace GAT layers with GCN | 0.966 | 0.896 | 0.594 |
| Use multi-heads (4 heads) for each GAT layer | 0.970 | 0.888 | 0.636 |
| Replace attention fusion with addition | 0.689 | 0.129 | 2.570e5 |
| Proposed MAAN | 0.981 | 0.913 | 0.550 |

**Table 7: Model Ablation - Attribute Inference. Higher (lower) values indicate better performance for AUC/AP (RMSE).**

| | AUC/AP for binary valued attributes | | RMSE for continuous valued attributes | | |
|---|---|---|---|---|---|
| Modification | UI Screen Genre | UI Element Comp. Type | UI Screen Image | UI Screen Desc. | UI Element Class Name *CHAR* |
| Replace GAT layers with GCN | 0.609/0.138 | 0.761/0.266 | 0.819 | 2.186 | 3.835 |
| Use multi-heads (4 heads) for each GAT layer | 0.771/0.201 | 0.955/0.680 | 1.122 | 1.589 | 1.037 |
| Replace attention fusion with addition | 0.519/0.098 | 0.686/0.140 | 2.487 | 1.059 | 1.611 |
| Proposed MAAN | 0.776/0.223 | 0.928/0.541 | 0.994 | 1.675 | 1.107 |

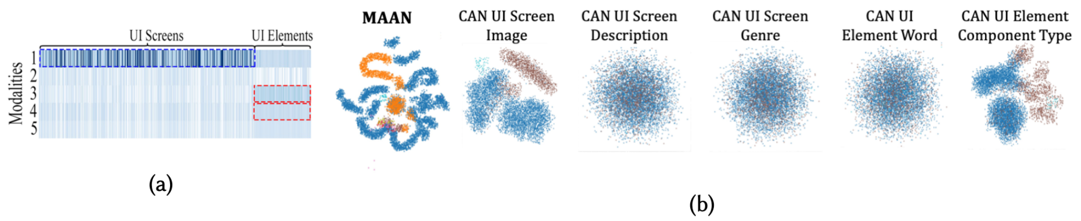

**Figure 4: (a) Attention weight visualization: Modality 1 corresponds to UI screen image; 2 to UI element class name; 3 to UI screen description; 4 to UI screen genre; 5 to UI element component type. Darker shade of blue means higher weights. The areas with higher weights for UI screen and element nodes are boxed in blue and red respectively. (b) Visualization of embedding spaces.**

**Table 8: Modality Ablation - Link Prediction/Regression. Higher (lower) values indicate better performance for AUC/AP (RMSE).**

| | Link Prediction | | Regression |
|---|---|---|---|
| Modification | AUC | AP | RMSE |
| MAAN-2 | 0.977 | 0.907 | 0.564 |
| MAAN-3 | 0.716 | 0.617 | 0.555 |
| MAAN-4 | 0.907 | 0.828 | 0.600 |
| MAAN All | 0.981 | 0.913 | 0.550 |

refers to MAAN with UI screen images and element class-names (CHAR) attributes. We further add the UI screen description for MAAN-3, and add the UI screen genre for MAAN-4. For link prediction and regression, using more modalities improves performance. For attribute inference, using all features (i.e. MAAN All) does not necessarily lead to better performance. Different combinations of features, i.e. feature selection, can further improve the performance of MAAN.

**Attention weights learnt by the MAAN model for the RICO-N dataset** - $\beta_{i,r}$ - are plotted as a heatmap in Figure 4(a). We observe that the generated embeddings for the UI screen nodes (node 0 to 5878) generally have higher attention weights for the screen image modality. The difference between the weights for different modalities for the UI element nodes is less obvious. The slightly higher weights for screen description and genre modalities for UI element

**Table 9: Modality Ablation - Attribute Inference. Higher (lower) values indicate better performance for AUC/AP (RMSE).**

| Modification | UI Screen Image (RMSE) | UI Element Class Name *CHAR* (RMSE) | UI Screen Desc. (RMSE) | UI Screen Genre (AUC/AP) |
|---|---|---|---|---|
| MAAN-2 | 1.353 | 0.712 | - | - |
| MAAN-3 | 0.729 | 0.536 | 0.699 | - |
| MAAN-4 | 0.659 | 1.152 | 1.312 | 0.746/0.193 |
| MAAN All | 0.994 | 1.107 | 1.675 | 0.776/0.223 |

nodes (node 5879 to 7441) could be due to the UI elements being themselves part of the UI screen image. We also use t-Distributed Stochastic Neighbor Embedding[20] to project the generated embeddings to two dimensions to **visualize the embedding spaces**. Figure 4(b) visualizes the embedding spaces generated for the RICO-N dataset. The presence of more clusters seems correlated with better task performance. The embedding generated by the MAAN model has more obvious node clusters than those from CAN. For CAN, the embeddings for modalities where performance is poorer do not have clear node clusters forming.

## 5 CONCLUSION AND FUTURE WORK

Based on the results of our experiments, we see that the MAAN model, due to its use of 1) information from multiple modalities, 2) attention mechanism, and 3) the MMD loss term performs more consistently than the baseline models whose performance varies based on the nature of the information. We also see that the use of node-wise attention in GAT layers together with modality-wise attention fusion leads to better performance. The learned attention weights also help us understand the importance of information from each modality. The experiment results also demonstrate that MAAN is able to generate UI screen and element embeddings that can be used in a range of downstream tasks, e.g. predicting links between UI screens and elements for UI design reconstruction, UI rating predictions, missing UI attribute inference and UI screen retrieval. Given MAAN's performance on the two distinct datasets that were used for experiments, the model is likely to be equally applicable to iOS applications even if characteristics of the information differ. Directions for future work include:

- Multiple edge-types: The current MAAN model utilizes structural network information for a single edge-type. Capturing other edge-types could improve performance.
- Positional information: We did not capture the order of the attributes - e.g. order of UI elements in the view hierarchy. Such positional information could improve task performance.
- End-to-end model: Information from some modalities - e.g. images, descriptions - are encoded separately. Integrating the encoders within MAAN could lead to better performance.

## ACKNOWLEDGMENTS

This research is supported by the National Research Foundation, Singapore under its International Research Centres in Singapore Funding Initiative. 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.

## REFERENCES

[1] 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.

[2] 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 (UIST '17)*.

[3] Michal Derezinski, Khashayar Rohanimanesh, and Aamer Hydrie. 2018. Discovering Surprising Documents with Context-Aware Word Representations. In *23rd International Conference on Intelligent User Interfaces* (Tokyo, Japan) *(IUI '18)*. New York, NY, USA.

[4] 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*. Halifax NS Canada.

[5] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. VSE++: Improving Visual-Semantic Embeddings with Hard Negatives. In *Proceedings of the British Machine Vision Conference*.

[6] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. *CoRR* (2017). arXiv:1704.01212

[7] Asnat Greenstein-Messica, Lior Rokach, and Michael Friedman. 2017. Session-Based Recommendations Using Item Embedding *(IUI '17)*. New York, NY, USA.

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, California, USA.

[9] 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 2017, 4-9 December 2017, Long Beach, CA, USA*.

[10] 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*. Virtual Event China.

[11] 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*.

[12] Feiran Huang, Xiaoming Zhang, Jie Xu, Chaozhuo Li, and Zhoujun Li. 2019. Network embedding by fusing multimodal contents and links. *Knowledge-Based Systems* 171 (2019), 44–55.

[13] 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*. Virtual Event China.

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

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

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

[17] 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*. Honolulu, Hawaii, USA.

[18] 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*. Lyon, France.

[19] 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.

[20] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[21] 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*.

[22] Zaiqiao Meng, Shangsong Liang, Jinyuan Fang, and Teng Xiao. 2019. Semi-supervisedly Co-embedding Attributed Networks. In *Advances in neural information processing systems*.

[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, New York, USA.

[24] Andreas Riegler and Clemens Holzmann. 2018. Measuring Visual User Interface Complexity of Mobile Applications With Metrics. *Interacting with Computers* 30, 3 (2018), 207–223.

[25] 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, ESWC 2018, Heraklion, Crete, Greece.*

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

[27] 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 2017, Long Beach, CA, USA.*

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018).

[29] 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*. Paris France.

[30] 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*. San Francisco, CA, USA.

[31] Evgenia Wasserman Pritsker, Tsvi Kuflik, and Einat Minkov. 2017. Assessing the Contribution of Twitter's Textual Information to Graph-Based Recommendation. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces* (Limassol, Cyprus) *(IUI '17)*. New York, NY, USA.

[32] Kodzo Wegba, Aidong Lu, Yuemeng Li, and Wencheng Wang. 2018. Interactive Storytelling for Movie Recommendation through Latent Semantic Analysis. In *23rd International Conference on Intelligent User Interfaces* (Tokyo, Japan) *(IUI '18)*. New York, NY, USA.

[33] 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, ICML, Long Beach, California, USA.*

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

[35] Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2019. InfoVAE: Balancing Learning and Inference in Variational Autoencoders. *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, Honolulu, Hawaii, USA.*