Graph Transformer for Label Placement

Category: VIS 2024 Full Papers

Paper Type: Applications

Abstract— Placing text labels is a common way to explain key elements in a given scene. Given a graphic input and original label information, how to place labels to meet both geometric and aesthetic requirements is an open challenging problem. Geometry-wise, traditional rule-driven solutions struggle to capture the complex interactions between labels, let alone consider graphical/appearance content. In terms of aesthetics, training/evaluation data ideally require nontrivial effort and expertise in design, thus resulting in a lack of decent datasets for learning-based methods. To address the above challenges, we formulate the task with a graph representation, where nodes correspond to labels and edges to interactions between labels, and treat label placement as a node position prediction problem. With this novel representation, we design a Label Placement Graph Transformer (LPGT) to predict label positions. Specifically, edge-level attention, conditioned on node representations, is introduced to reveal potential relationships between labels. To integrate graphic/image information, we design a feature aligning strategy that extracts deep features for nodes and edges efficiently. Next, to address the dataset issue, we collect commercial illustrations with professionally designed label layouts from household appliance manuals, and annotate them with useful information to create a novel dataset named the Appliance Manual Illustration Labels (AMIL) dataset. In the thorough evaluation on AMIL, our LPGT solution achieves promising label placement performance compared with popular baselines.

Index Terms—Label placement, Graph neural network, Transformer

1 Introduction

Short text annotations, typically referred to as labels, are a popular approach to interpreting crucial elements of a scene, and have been extensively utilized in graphic design [4, 30], information visualization [58, 70], and AR/VR [37, 63]. The placement of labels, *i.e.*, the label layout, greatly affects users' comprehension of the scene. From labeling point features such as cities on a geographical map [6, 41], to placing labels for area features such as tissues in medical illustrations [33, 34], there are many open problems in label placement due to its NP-hard nature [13].

Manual label placement is a time-consuming process [73], which has led to ongoing research into automatic label placement for decades [30, 33, 47, 64]. Recently, a few studies have introduced learning techniques [6, 11], encouraged by the success of deep learning in various domains. Nonetheless, the progress in addressing two critical challenges is still limited. First, the ideal position of a label is strongly related to the positions of all others, as investigated in previous studies. However, these studies largely depend on manually designed rules, which are hard to capture the complicated interactions among labels. Second, ideal label layouts require nontrivial effort and expertise in design. Consequently, datasets used in previous work are small or constructed at the laboratory level. The lack of appropriate benchmarks poses difficulties for learning-based methods, and partially limits deeper exploration of the interactions between labels.

In this work, we address both challenges with novel contributions. First, we propose a novel graph neural network (GNN) solution for label placement, inspired by recent advances in GNNs [66, 74]. In this solution, we represent the label layout by a graph, where nodes correspond to labels and edges to interactions between labels. We then formulate label placement as a node prediction problem, and design an iterative solution to predict node displacement for continuously refining label placement results. Building on the progress in Transformer-based architectures of GNNs [19, 75], we design a *Label Placement Graph Transformer* (LPGT) to predict label positions (Fig. 1). An edge-level attention conditioned on node representations is introduced, and combined with a node-level attention conditioned on edge representations to explore potential label relationships. To capture graphic/image information, we present a feature aligning strategy that effectively extracts deep features for nodes and edges. In particular, the edge features are extracted based on the characteristic that edges traverse different regions of an image.

Second, to alleviate the shortage of high-quality benchmarks, we construct a new dataset, named *Appliance Manual Illustration Labels* (AMIL) dataset, based on illustrations from commercial product manuals. The labels in these illustrations are created by professional designers and thus have excellent readability, unambiguity, compactness, and aesthetics [2]. Additionally, we introduce a metric for evaluating the quality of label layouts. We will release the dataset and our algorithm upon the publication of this paper.

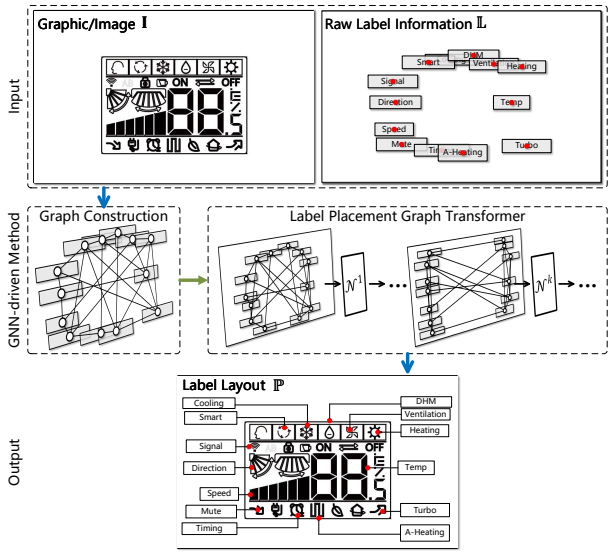

Fig. 1: GNN-driven label placement. For a set of labels to be placed in a graphic, the proposed Label Placement Graph Transformer predicts the label layout given the graphic and raw label information.

In summary, our main contributions include: 1) We formulate label placement as a graph node prediction problem. This novel formulation allows not only natural encoding of interaction among different labels, but also more effective inference techniques such as GNNs (Sec. 3). 2) We develop a novel Label Placement Graph Transformer solution, which we believe is the first GNN solution for label placement. In LPGT, a feature aligning strategy is introduced to efficiently extract appearance features for nodes and edges; and an edge-level attention is proposed to be combined with a node-level attention to encode relationship between labels (Sec. 4). 3) We build, to our best knowledge,

the first high-quality label placement dataset named Appliance Manual Illustration Labels, which contains professionally designed label layouts (Sec. 5). 4) In our carefully designed experiments, LPGT achieves promising performance, both quantitatively and qualitatively, in comparison with other solutions (Sec. 6).

2 Related Work

2.1 Label Placement

According to label positioning, label placement tasks are generally categorized into two types: internal and external. Internal labels are typically placed within the regions of target elements to reduce interference with other elements [34, 47]. However, when elements are not completely mutually exclusive, the label of one element can still obscure other elements, such as labels in hierarchical 3D biological environments [33]. With the significant developments in deep learning across various domains, some recent studies incorporate deep learning architectures for label placement [6]. Overall, internal labeling is constrained by the geometric structures of the target elements.

In contrast, external labels do not overlap with target elements [5]. The connections between them and the elements are represented by leader lines. Two typical groups of labeling approaches have been studied, boundary and flexible labeling. Boundary labeling approaches place labels along a predefined boundary of a target object, such as rectangle [4, 30], circle [63, 76], and contour [14]. Flexible labeling methods place labels in all available spaces [11, 23, 28, 35, 36, 38, 44, 45, 49, 55–57, 62, 64, 65]. In addition, some studies employ both internal and external labeling to explore more layout styles [15].

Despite the advances in label placement, it remains challenging to effectively capture the interactions between labels, as well as a shortage of high-quality benchmarks. In this paper, we overcome these limitations by modeling label placement with GNNs and collecting professionally designed label layouts from commercial product illustrations.

2.2 Graph Neural Networks

GNNs have achieved remarkable improvements in various graph-related tasks [66, 74]. According to the approach of computing relationships between nodes and implementing graph convolution, GNNs can be divided into two groups: Spectral GNNs [1, 16, 32, 39, 67] and Spatial GNNs [3, 9, 22, 25, 43]. The former leverage the eigenvalues and eigenvectors of the graph Laplacian to define graph convolution, while the latter perform convolution operations directly between a node and its neighbors. The impressive success of Transformer in various domains [18, 20, 40, 60] also encourages the introduction of attention mechanisms or Transformer-based architectures in GNNs [8, 19, 50, 53, 54, 61, 72, 75]. Among them, the most related work to ours is [19], which proposes relational attention, a node-level attention that incorporates edge vectors. We introduce this node-level attention into our LPGT with a modification to layer normalization for better optimization, as inspired by [68]. Furthermore, unlike aggregating representations of two associated nodes to update each edge representation in [19], we design an edge-level attention conditioned on node representations to explore label interactions more effectively. Combined with the feature aligning strategy, our LPGT is the first GNN solution for label placement.

3 Problem Formulation

3.1 Problem Definition

In this paper, we adopt external labels for graphics. The key components of an external label include (Fig. 2): 1) Text: annotations that illustrate the target element. 2) Label box: a rectangular box that contains the text. 3) Anchor: a 2D point that indicates the representative position (*e.g.*, the center) on the target element.

Given a set of labels to be placed in a graphic, we define the label placement problem as follows:

Input: A graphic/image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$ and raw label information denoted by $\mathbb{L} = \left\{\left(\mathbf{a}_i \in \mathbb{R}^2, \mathbf{s}_i \in \mathbb{R}^2\right)\right\}_{i=1}^{n_l}$, where $\mathbf{a}_i$ and $\mathbf{s}_i$ describe the anchor position and box size of the i-th label, respectively, and n_l is the number of labels.

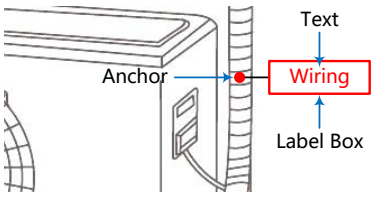

Fig. 2: Components of an external label: text, label box, and anchor.

Table 1: Symbol notations.

Symbol	Meaning
$\mathbb{L},\mathbb{P},\mathbb{D}$	sets of label information, positions, displacements
$\mathbb{G},\mathbb{V},\mathbb{E},\mathbb{N}$	complete graph, sets of node vectors, edge vectors, neighbors
$\mathbb{R},\mathbb{S}$	sets of real numbers, pixel points sampled along an edge
$\mathbf{I},\mathbf{F},\mathbf{W}$	input image, convolutional feature maps, weight matrix
$\mathbf{a},\mathbf{s},\mathbf{p},\delta,\mathbf{o}$	anchor, box size, position, displacement, box vectors of a label
$\mathbf{d}$	distance vector between two labels
$\mathbf{f},\mathbf{z},\mathbf{e}$	deep feature, node representation, edge representation
$\mathbf{q},\mathbf{k},\mathbf{v},\mathbf{r},\mathbf{b}$	QKV vectors of attention, image patch, bias vector
$\mathcal{N},\mathscr{E},\mathscr{D},\mathscr{L}$	GNN module, fusion function, output layer of decoder, loss function
n,d	quantity, dimensionality
z,e,f,g	superscripts or subscripts of node, edge, deep feature, geometric feature

Output: Expected label positions $\mathbb{P} = \left\{\hat{\mathbf{p}}_i \in \mathbb{R}^2\right\}_{i=1}^{n_l}$, where $\hat{\mathbf{p}}_i$ is the final position of i-th label. The labels and corresponding anchors are linked by standard leader lines [5] to form the label layout. A trivial solution is to place labels at the anchor, *i.e.*, $\hat{\mathbf{p}}_i = \mathbf{a}_i$, but coming with poor aesthetics such as label overlap or blocking important information.

3.2 Graph Formulation

We associate each label with a node. A complete graph $\mathbb{G}$ is constructed to capture the relationship between any pair of labels. Specifically, we define it as an attributed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V} = \{\mathbf{z}_i\}_{i=1}^{n_l}$ denotes a set of node vectors; $\mathbf{z}_i = \left[\mathbf{a}_i, \mathbf{s}_i, \mathbf{f}_i^z\right]$, where $\mathbf{f}_i^z$ is the deep features of node i, extracted from the image $\mathbf{I}$ [27], and $[\cdot]$ denotes the column-wise concatenation. Similarly, $\mathbb{E} = \left\{\mathbf{e}_{ij}\right\}$ denotes a set of edge vectors, where $\mathbf{e}_{ij} = \left[\mathbf{a}_i - \mathbf{a}_j, \mathbf{s}_i, \mathbf{s}_j, \mathbf{f}_{ij}^e\right]$, $\mathbf{f}_{ij}^e$ denotes the deep features of edge (i, j), and $n_e = |\mathbb{E}|$ denotes the number of edges. Key symbols are explained in Tab. 1.

Based on the above graph representation, we can transform the label placement problem into a node position prediction problem. While a straightforward strategy is to predict the final positions of labels directly, it is hard to learn such models due to the large variation in the absolute positions. Alternatively, we propose learning the iterative displacements relative to initial positions, a more controllable approach. This is because, in each iteration, we only need to predict small displacement vectors for each label.

With the above idea, we design a Label Placement Graph Transformer network to estimate the displacements $\mathbb{D}$ between the initial and final positions of each label. The displacement of each label consists of multiple steps:

$$\mathbb{D} = \left\{\hat{\mathbb{D}}^k\right\}_{k=1}^{n_s} = \left\{\left\{\hat{\delta}_i^k \in \mathbb{R}^2\right\}_{i=1}^{n_l}\right\}_{k=1}^{n_s} \tag{1}$$

where n_s denotes the number of iterations, $\hat{\delta}_i^k$ denotes the displacement of the i-th label at the k-th step, and $\hat{\mathbb{D}}^k$ denotes the corresponding displacement set. The initial position of each label is set as its anchor, *i.e.*, the central point of its box is aligned with the anchor. The final predicted label layout is achieved by accumulating the displacements:

$$\hat{\mathbf{p}}_i = \mathbf{a}_i + \sum_{k=1}^{n_s} \hat{\delta}_i^k \tag{2}$$

4 Methodology

The proposed LPGT, as shown in Fig. 3, contains two main stages: graph construction and displacement estimation.

- **Graph Construction.** A graph is constructed to represent labels and their relationships. Graphic/image and raw label information

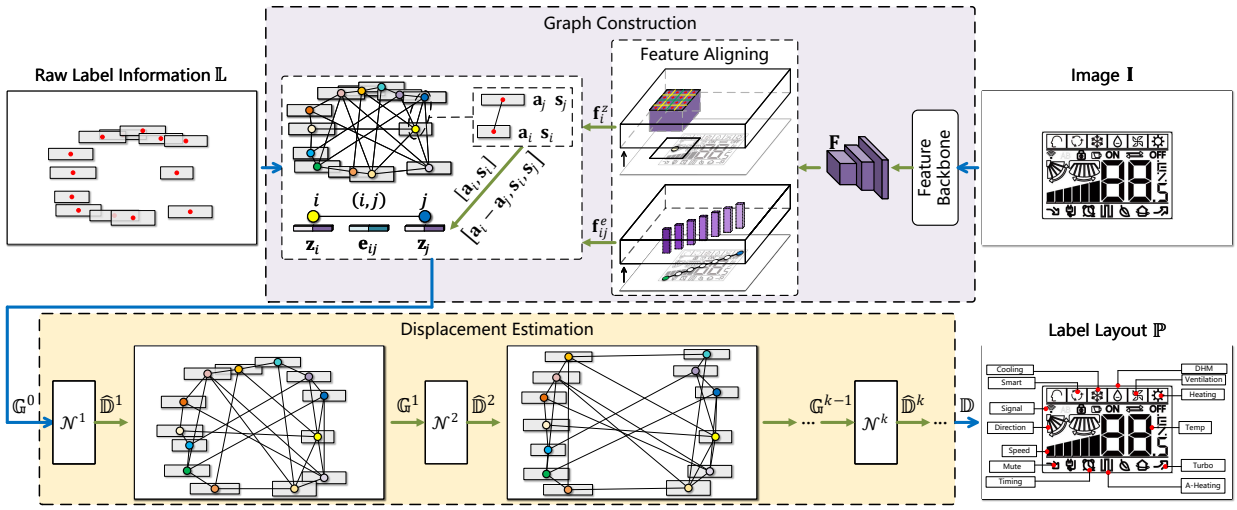

Fig. 3: Architecture of LPGT. First, a complete graph $\mathbb{G}$ is constructed to capture the relationship between labels. Its node and edge features are generated from the label information and image features. Next, given the graph $\mathbb{G}$ as input, LPGT iteratively learns the displacements of the nodes by a sequence of GNN modules. The graph is updated by each module and taken as input for the next module. For clarity, some edges are omitted in the graph $\mathbb{G}$.

are utilized to generate the features of nodes and edges. A feature aligning strategy is designed to extract deep features for nodes and edges. Specially, edge features are constructed by integrating visual information between labels, based on the characteristic that edges traverse diverse areas of an image.

- **Displacement Estimation.** LPGT takes the graph as input, and iteratively predicts displacements for nodes by sequential GNN modules with the same architecture but different weights. Each GNN module estimates displacements for all nodes, and the graph is updated according to these displacements. In the GNN module, a feature fusion mechanism is utilized to jointly incorporate label information and image features into node and edge representations. Besides, node-level attention conditioned on edge representations is used to learn relationships between labels. Edge-level attention conditioned on node representations is specifically designed to explore interactions between label relationships.

4.1 Graph Construction

Given the raw label information $\mathbb{L}$, a complete graph $\mathbb{G}$ is constructed to encode the interaction between labels. To generate informative and distinctive node and edge features, we take full advantage of label information and image features. The labels are expected to move from the anchors to the final positions. Therefore, for each node i, we concatenate the 2D label anchor point $\mathbf{a}_i = (x_i^a, y_i^a)$ and the box size $\mathbf{s}_i = (h_i, w_i)$ as the geometric part $[\mathbf{a}_i, \mathbf{s}_i]$ of its features. To capture the image features for node i, we first extract convolutional feature maps $\mathbf{F} \in \mathbb{R}^{h_f \times w_f \times d_f}$ from the entire image $\mathbf{I}$ by a convolutional neural network (CNN), where h_f and w_f denote the spatial size of deep features, and d_f denotes the dimensions. Deep features contain high-level semantic information in the image, including its shapes, textures, and other visual attributes [27]. It is worth noting that direct application of standard CNNs may be ineffective for line drawings in manual illustrations. To alleviate this issue, we apply Gaussian blur to the input image before feeding them into the feature backbone.

Intuitively, the deep features $\mathbf{f}_i^z \in \mathbb{R}^{d_f}$ of node i can be naturally recovered by a node-to-pixel correspondence on the feature maps $\mathbf{F}$. This node-to-pixel mapping requires that the features $\mathbf{f}_i^z$, which are themselves local feature vectors, are well aligned to preserve the explicit visual information per node. A straightforward nearest neighbor strategy is to extract local feature vectors on the feature maps by coordinate quantization. Node i is mapped onto the feature maps $\mathbf{F}$ at $(x_i^a/s, y_i^a/s)$ based on its anchor coordinates, where s is the scaling factor between the image $\mathbf{I}$ and the feature maps $\mathbf{F}$. Then, the floating point coordinates of each node are quantized to the discrete granularity, *i.e.*,

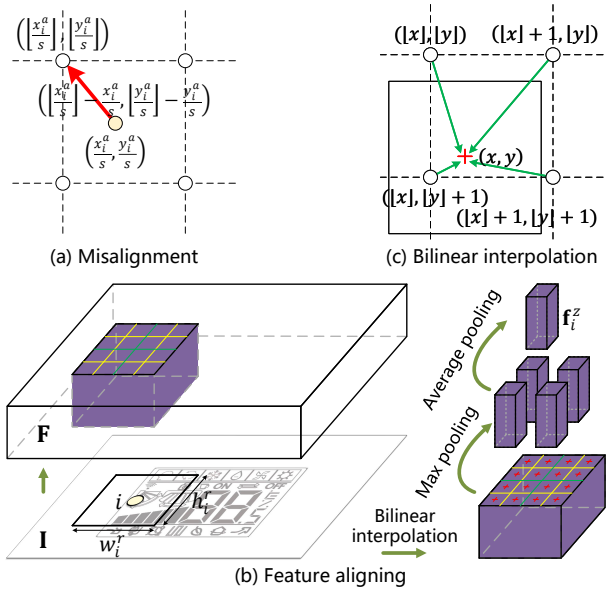

Fig. 4: (a) The straightforward strategy for deep feature extraction causes the misalignment $(\lfloor x_i^a/s \rfloor - x_i^a/s, \lfloor y_i^a/s \rfloor - y_i^a/s)$ between a node and its selected feature vector. (b) Our feature aligning strategy mitigates the misalignment by the node-to-patch correspondence and the bilinear interpolation shown in (c).

$(\lfloor x_i^a/s \rfloor, \lfloor y_i^a/s \rfloor)$, where $\lfloor \cdot \rfloor$ is the flooring operation. The feature vector located at this position is selected as $\mathbf{f}_i^z$. However, the quantization introduces misalignment between a node and its selected feature vector (*e.g.*, the offset in Fig. 4(a)), and fails to ensure the well-alignment requirement. In addition, the selected feature may not embed effective visual information due to blank areas in the input line drawings.

Therefore, we utilize a feature aligning strategy to avoid the harsh quantization (Fig. 4(b)), inspired by the RoI align strategy [26]. For each node i, we construct a patch $\mathbf{r}_i = (h_i^r, w_i^r)$ centered on it, where h_i^r and w_i^r denote the patch size. Then, the patch is mapped onto the feature maps $\mathbf{F}$, and centered on $(x_i^a/s, y_i^a/s)$ with the size scaled to $(h_i^r/s, w_i^r/s)$. Next, the patch is equally divided into four bins, and each bin is further evenly split into four regions. The local feature vector located at the center of each region is bi-linearly interpolated from the neighboring feature vectors, as described in Fig. 4(c). Finally,

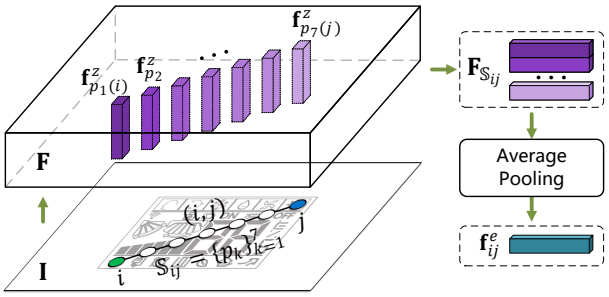

Fig. 5: Feature extraction for edges. A set of pixel points $\mathbb{S}_{ij}$ is uniformly sampled along an edge (i, j). Then, their image features are extracted through the feature aligning process, forming a feature matrix $\mathbf{F}_{\mathbb{S}_{ij}}$. Finally, an average pooling is applied to $\mathbf{F}_{\mathbb{S}_{ij}}$ to obtain the deep edge features $\mathbf{f}^e_{ij}$.

we apply a max pooling over the obtained four feature vectors of each bin to obtain the deep features with the shape $\mathbb{R}^{2\times2\times d_f}$ of the patch $\mathbf{r}_i$. The deep node features $\mathbf{f}^z_i \in \mathbb{R}^{d_f}$ are generated by applying an average pooling over the patch features, followed by L2 normalization. Through this node-to-patch correspondence, the feature aligning strategy extracts the accurate and rich visual information of the target elements for labels. The deep features $\mathbf{f}^z_i$ and the aforementioned geometric features $[\mathbf{a}_i, \mathbf{s}_i]$ constitute the features of node i, *i.e.*, $\mathbf{z}_i = \left[\mathbf{a}_i, \mathbf{s}_i, \mathbf{f}^z_i\right]$.

To build the features of each edge (i, j), the distance between its two nodes i and j is computed, and concatenated with the sizes of two corresponding boxes as the geometric part $\left[\mathbf{a}_i - \mathbf{a}_j, \mathbf{s}_i, \mathbf{s}_j\right]$ of its features. The distance reveals the spatial relation between the two labels.

For the deep features on an edge, we extract rich visual information from its bridge across the two anchor points by sampling and average pooling. Specifically, for an edge (i, j) in Fig. 5, we first use the Bresenham's line algorithm [7] to compute the pixel points along the edge (i, j). Then, seven points are uniformly sampled from them to form a point set $\mathbb{S}_{ij} = \{p_k\}^7_{k=1}$, including the two end nodes i and j, *i.e.*, p_1 and p_7. Next, the image features of these samples are extracted by the feature aligning process, and then stacked to form a feature matrix $\mathbf{F}_{\mathbb{S}_{ij}} \in \mathbb{R}^{|\mathbb{S}_{ij}|\times d_f}$. Finally, we apply an average pooling over $\mathbf{F}_{\mathbb{S}_{ij}}$ to obtain the deep edge features $\mathbf{f}^e_{ij}$, followed by L2 normalization. The features of edge (i, j) consist of the above geometric features $\left[\mathbf{a}_i - \mathbf{a}_j, \mathbf{s}_i, \mathbf{s}_j\right]$ and the deep features $\mathbf{f}^e_{ij}$, *i.e.*, $\mathbf{e}_{ij} = \left[\mathbf{a}_i - \mathbf{a}_j, \mathbf{s}_i, \mathbf{s}_j, \mathbf{f}^e_{ij}\right]$.

4.2 Label Placement Graph Transformer

Using the graph representing the labels, LPGT iteratively learns the displacements of nodes to construct the label layout. It consists of sequential GNN modules $\left\{\mathcal{N}^k\right\}^{n_s}_{k=1}$ with the same architecture but different weights, adapted to the multi-step estimation of the node displacements. For the k-th GNN module (as shown in Fig. 6), it takes the graph $\mathbb{G}^{k-1}$ as input and predicts the k-th step displacement for each node, $\hat{\mathbb{D}}^k = \mathcal{N}^k\left(\mathbb{G}^{k-1}\right)$. Then, the position of each node is updated to $\tilde{\mathbf{p}}^k_i \in \mathbb{R}^2$ according to Eq. (2), *i.e.*, adding the accumulated displacements $\sum^k_{m=1} \hat{\delta}^m_i$ to the initial position $\mathbf{a}_i$. Consequently, the graph is updated to $\mathbb{G}^k$ for the next step. Note that the initial graph $\mathbb{G}^0$ is constructed based on the anchor point of each node and the approach described in Sec. 4.1. The graph update is also achieved in this way by simply replacing anchor points with new node positions.

Each GNN module comprises three main components: encoder, core, and decoder. The details are introduced using the first GNN module $\mathcal{N}^1$ that takes the initial graph $\mathbb{G}^0$ as input.

4.2.1 Encoder

To jointly integrate label information and image features into the representation learning of nodes and edges, we design a feature fusion mechanism in the encoder. The geometric and visual features of nodes are fused by a fusion function $\mathscr{E}^z$, which is implemented by linear projections of the two feature parts based on the respective weight matrices $\mathbf{W}^z_g \in \mathbb{R}^{4\times d_z}$ and $\mathbf{W}^z_f \in \mathbb{R}^{d_f\times d_z}$. The geometric and visual edge features are fused in the same way by a function $\mathscr{E}^e$ with the corresponding weight matrices $\mathbf{W}^e_g \in \mathbb{R}^{6\times d_e}$ and $\mathbf{W}^e_f \in \mathbb{R}^{d_f\times d_e}$:

$$\begin{aligned}\mathbf{z}_i &= \mathscr{E}^z\left(\mathbf{a}_i, \mathbf{s}_i, \mathbf{f}^z_i\right) := \left[\mathbf{a}_i, \mathbf{s}_i\right]\mathbf{W}^z_g + \mathbf{f}^z_i\mathbf{W}^z_f \\ \mathbf{e}_{ij} &= \mathscr{E}^e\left(\mathbf{a}_i - \mathbf{a}_j, \mathbf{s}_i, \mathbf{s}_j, \mathbf{f}^e_{ij}\right) := \left[\mathbf{a}_i - \mathbf{a}_j, \mathbf{s}_i, \mathbf{s}_j\right]\mathbf{W}^e_g + \mathbf{f}^e_{ij}\mathbf{W}^e_f\end{aligned} \tag{3}$$

where d_z and d_e denote the hidden dimensions of the node and edge representations, respectively.

4.2.2 Transformer Layers

The core block consists of several Transformer layers, and each Transformer layer contains a node Transformer (sub)layer and an edge Transformer (sub)layer.

Node Transformer Layer The node Transformer layer achieves the aggregation and updating of node representations through a multi-head self-attention (MHSA) block and a feed-forward network (FFN). The MHSA block is implemented based on the node-level attention in [19]. The node-level attention projects QKV vectors from each node representation, simultaneously conditioned on the edge representation between two nodes. Specifically, the node and edge representations are projected using separate weight matrices, and the QKV vectors of the nodes and edges are then added. Each node i attends to its first-order neighbors $j \in \mathbb{N}_i$, where $\mathbb{N}_i$ denotes the neighbor node set:

$$\begin{aligned}\mathbf{q}^z_{ij} &= \mathbf{z}_i\mathbf{W}^z_{q_1} + \mathbf{e}_{ij}\mathbf{W}^e_{q_1} \\ \mathbf{k}^z_{ij} &= \mathbf{z}_j\mathbf{W}^z_{k_1} + \mathbf{e}_{ij}\mathbf{W}^e_{k_1} \\ \mathbf{v}^z_{ij} &= \mathbf{z}_j\mathbf{W}^z_{v_1} + \mathbf{e}_{ij}\mathbf{W}^e_{v_1}\end{aligned} \tag{4}$$

where $\mathbf{W}^z_{q_1}, \mathbf{W}^z_{k_1}, \mathbf{W}^z_{v_1} \in \mathbb{R}^{d_z\times d_{qkv}}$ and $\mathbf{W}^e_{q_1}, \mathbf{W}^e_{k_1}, \mathbf{W}^e_{v_1} \in \mathbb{R}^{d_e\times d_{qkv}}$ are the weight matrices used for projecting QKV vectors of the node and edge representations respectively in the node-level attention. d_{qkv} denotes the dimensions of the QKV vectors.

The multiple self-attention are calculated and concatenated, then projected to obtain the MHSA. For each node i, the representations of its neighbors are aggregated in the MHSA block. Its representation is then updated by the FFN block, consisting of two linear layers with a ReLU non-linearity.

$$\begin{aligned}\mathrm{SA}^z_{\mathrm{h}}\left(\mathbf{z}_i\right) &= \sum_{j\in\mathbb{N}_i} \mathrm{softmax}_j\left(\frac{\mathbf{q}^{z\mathrm{h}}_{ij}\mathbf{k}^{z\mathrm{h}\top}_{ij}}{\sqrt{d_{qkv}}}\right)\mathbf{v}^{z\mathrm{h}}_{ij} \\ \mathrm{MHSA}^z\left(\mathbf{z}_i\right) &= \left[\mathrm{SA}^z_{\mathrm{h}}\left(\mathbf{z}_i\right)\right]^{n^z_{\mathrm{H}}}_{\mathrm{h}=1}\mathbf{W}^z_o + \mathbf{b}^z_o \\ \mathbf{z}'_i &= \mathrm{MHSA}^z\left(\mathrm{LN}\left(\mathbf{z}_i\right)\right) + \mathbf{z}_i \\ \mathbf{z}''_i &= \mathrm{ReLU}\left(\mathrm{LN}\left(\mathbf{z}'_i\right)\mathbf{W}_1\right)\mathbf{W}_2 + \mathbf{z}'_i\end{aligned} \tag{5}$$

where $\mathbf{W}^z_o \in \mathbb{R}^{n^z_{\mathrm{H}}d_{qkv}\times d^z_o}$ and $\mathbf{b}^z_o \in \mathbb{R}^{d^z_o}$ are the weight matrix and the bias vector for computing the node MHSA, d^z_o denotes the corresponding output dimensions, n^z_{H} is the number of heads, and h is the head index. $\mathbf{W}_1 \in \mathbb{R}^{d_z\times d^z_h}$ and $\mathbf{W}_2 \in \mathbb{R}^{d^z_h\times d_z}$ are the weight matrices of the node FFN block, and d^z_h denotes the corresponding hidden dimensions. In addition, we apply layer normalization (LN) before the MHSA and FFN blocks instead of after them as adopted in [19]. This modification has been widely adopted by current Transformer implementations due to more effective optimization [68]. Residual connections are applied after each block. The node-level attention captures the dependencies between all labels due to the complete graph $\mathbb{G}$ we model.

Edge Transformer Layer We further propose an edge-level attention conditioned on node representations to explore the interactions between label relationships. The edge-level attention projects QKV vectors from each edge representation and associated node representations, then adds the QKV vectors of edges and nodes:

$$\begin{aligned}\mathbf{q}^e_{ij} &= \mathbf{e}_{ij}\mathbf{W}^e_{q_2} + \mathbf{z}''_i\mathbf{W}^z_{q_2} + \mathbf{z}''_j\mathbf{W}^z_{q_2} \\ \mathbf{k}^e_{kl} &= \mathbf{e}_{kl}\mathbf{W}^e_{k_2} + \mathbf{z}''_k\mathbf{W}^z_{k_2} + \mathbf{z}''_l\mathbf{W}^z_{k_2} \\ \mathbf{v}^e_{kl} &= \mathbf{e}_{kl}\mathbf{W}^e_{v_2} + \mathbf{z}''_k\mathbf{W}^z_{v_2} + \mathbf{z}''_l\mathbf{W}^z_{v_2}\end{aligned} \tag{6}$$

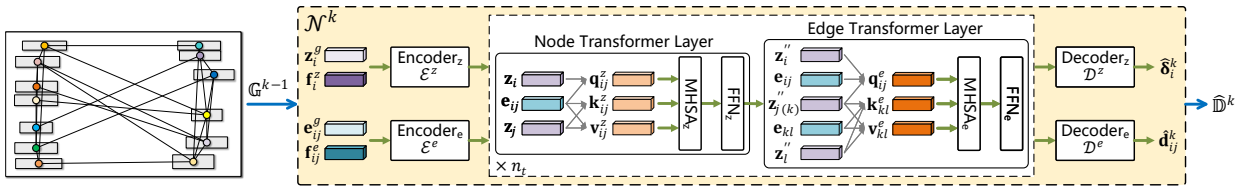

Fig. 6: The k-th GNN module takes the graph $\mathbb{G}^{k-1}$ as input, and predicts the k-th step displacements $\hat{\mathbb{D}}^k$ of nodes. The geometric and visual features are fused in the encoders $\mathscr{E}^z$ and $\mathscr{E}^e$. The node and edge representations are then updated by the n_t node and edge Transformer layers. Finally, the decoders $\mathscr{D}^z$ and $\mathscr{D}^e$ predict the displacements of nodes and the distances between nodes, respectively.

where $\mathbf{W}^z_{q_2}, \mathbf{W}^z_{k_2}, \mathbf{W}^z_{v_2} \in \mathbb{R}^{d_z \times d_{qkv}}$ and $\mathbf{W}^e_{q_2}, \mathbf{W}^e_{k_2}, \mathbf{W}^e_{v_2} \in \mathbb{R}^{d_e \times d_{qkv}}$ denote the weight matrices used for projecting QKV vectors of the node and edge representations respectively in the edge-level attention.

The node-level attention maintains Transformer's $O\left(n_l^2\right)$ complexity. However, if each edge attends to all edges, the computational complexity increases to $O\left(n_l^4\right)$ due to the n_l^2 edges in $\mathbb{G}$. Therefore, we introduce neighbors for each edge and restrict its aggregation within its neighborhood. The neighbors of an edge are defined as edges that share at least one associated node with it. Since each edge only attends to its neighbors, the computational complexity drops to $O\left(n_l^2 \bar{n}_e\right)$, where $\bar{n}_e$ is the average number of neighbors for all edges. Taking advantage of the reduced complexity, the MHSA and FFN blocks update the representation of each edge (i,j) based on the weighted aggregation of the representations from its neighbors $(k,l) \in \mathbb{N}_{ij}$, where $\mathbb{N}_{ij}$ denotes the neighbor edge set:

$$
\begin{aligned}
\mathrm{SA}^e_{\mathrm{h}}\left(\mathbf{e}_{ij}\right) &= \sum_{(k,l)\in\mathbb{N}_{ij}} \mathrm{softmax}_{kl}\left(\frac{\mathbf{q}^{e\mathrm{h}}_{ij}\mathbf{k}^{e\mathrm{h}\top}_{kl}}{\sqrt{d_{qkv}}}\right)\mathbf{v}^{e\mathrm{h}}_{kl} \\
\mathrm{MHSA}^e(\mathbf{e}_{ij}) &= \left[\mathrm{SA}^e_{\mathrm{h}}\left(\mathbf{e}_{ij}\right)\right]_{\mathrm{h}=1}^{n^e_{\mathrm{H}}}\mathbf{W}^o_e + \mathbf{b}^o_e \\
\mathbf{e}'_{ij} &= \mathrm{MHSA}^e\left(\mathrm{LN}\left(\mathbf{e}_{ij}\right)\right) + \mathbf{e}_{ij} \\
\mathbf{e}''_{ij} &= \mathrm{ReLU}\left(\mathrm{LN}\left(\mathbf{e}'_{ij}\right)\mathbf{W}_3\right)\mathbf{W}_4 + \mathbf{e}'_{ij}
\end{aligned}
\tag{7}
$$

where $\mathbf{W}^e_o \in \mathbb{R}^{n^e_{\mathrm{H}} d_{qkv} \times d^e_o}$ and $\mathbf{b}^e_o \in \mathbb{R}^{d^e_o}$ denote the weight matrix and the bias vector for computing the edge MHSA, d^e_o is the corresponding output dimensions, and n^e_{H} is the number of heads. $\mathbf{W}_3 \in \mathbb{R}^{d_e \times d^e_h}$ and $\mathbf{W}_4 \in \mathbb{R}^{d^e_h \times d_e}$ are the weight matrices of the edge FFN block, and d^e_h denotes the corresponding hidden dimensions. Edge-level attention reveals interactions between label relationships.

4.2.3 Decoder

The decoder contains two output layers, $\mathscr{D}^z$ and $\mathscr{D}^e$. The node output layer predicts the displacement of each node, $\hat{\boldsymbol{\delta}}^1_i = \mathscr{D}^z(\mathbf{z}_i)$. The edge output layer estimates the distance vector between the two nodes, $\hat{\mathbf{d}}^1_{ij} \in \mathbb{R}^2 = \mathscr{D}^e(\mathbf{e}_{ij})$. To construct the input graph $\mathbb{G}^1$ of the second GNN module $\mathscr{N}^2$, the current graph $\mathbb{G}^0$ is updated based on the above prediction results. Specifically, the position of each node is updated by adding the displacement to the initial position, $\tilde{\mathbf{p}}^1_i = \mathbf{a}_i + \hat{\boldsymbol{\delta}}^1_i$. The geometric features of each node and edge are updated as $\left[\tilde{\mathbf{p}}^1_i, \mathbf{s}_i\right]$ and $\left[\hat{\mathbf{d}}^1_{ij}, \mathbf{s}_i, \mathbf{s}_j\right]$, respectively. The visual features are constructed over the new node positions through the feature aligning strategy.

For each label, the multi-step displacements are predicted by n_s GNN modules, based on the above three components and the graph update. The final position is then determined by accumulating all displacements from the initial position.

4.3 Training Objective

To optimize all GNN modules $\left\{\mathscr{N}^k\right\}_{k=1}^{n_s}$ of LPGT, we use a multi-phase training strategy based on the iterative solution of the label displacement estimation. The strategy consists of n_s phases. In the k-th phase, the first k modules of LPGT are trained using the following losses: displacement loss $\mathscr{L}_\delta$, distance loss $\mathscr{L}_d$, and overlap loss $\mathscr{L}_o$.

The displacement loss $\mathscr{L}_\delta$ is designed to guide labels to the expected positions. The ground-truth displacements $\mathbb{D}_{gt} = \left\{\boldsymbol{\delta}_i \in \mathbb{R}^2\right\}_{i=1}^{n_l}$ are thus introduced, which are computed based on the anchor $\mathbf{a}_i$ and the ground-truth label position $\mathbf{p}_i \in \mathbb{R}^2$ of each label, *i.e.*, $\boldsymbol{\delta}_i = \mathbf{p}_i - \mathbf{a}_i$. We adopt the Mean Squared Error (MSE) to measure the differences between the accumulated predicted displacements by the first k GNN modules and the ground-truth:

$$
\mathscr{L}^k_\delta = \frac{1}{n_l}\sum_{i=1}^{n_l}\left\|\sum_{m=1}^{k}\hat{\boldsymbol{\delta}}^m_i - \boldsymbol{\delta}_i\right\|^2 \tag{8}
$$

For each label, distance vectors to other labels indicate their spatial relations with respect to it. The distance loss $\mathscr{L}_d$ is proposed to optimize the spatial relations between any two labels. We introduce the ground-truth distance vectors, which are computed based on the ground-truth label positions, $\mathbf{d}_{ij} = \mathbf{p}_i - \mathbf{p}_j$. We utilize this supervision for the predicted distance vectors $\left\{\hat{\mathbf{d}}^m_{ij}\right\}_{m=1}^{k}$ of all the GNN modules trained at the k-th phase:

$$
\mathscr{L}^k_d = \frac{1}{k}\sum_{m=1}^{k}\frac{1}{n_l}\sum_{i=1}^{n_l}\left\|\hat{\mathbf{d}}^m_{ij} - \mathbf{d}_{ij}\right\|^2 \tag{9}
$$

To reduce the overlap between labels, we design the overlap loss $\mathscr{L}_o$. The degree of overlap between any two labels is quantified by the size of the overlap area of their label boxes. Therefore, we introduce Intersection over Union (IoU) to measure the label overlap. It is a metric widely used in the field of computer vision, such as object detection task [51]. It evaluates the degree of overlap between the predicted and ground-truth bounding boxes of an object. The IoU is calculated through dividing the intersection area between the two bounding boxes by the union area of them. It ranges from 0 to 1, where 0 indicates no overlap and 1 denotes perfect overlap. We focus on minimizing the overlap areas of label boxes. In the k-th training phase, each label box is represented by its central point and size, $\mathbf{o}^k_i = \left[\tilde{\mathbf{p}}^k_i, \mathbf{s}_i\right]$. We compute the IoU over all pairs of label boxes to form the overlap loss $\mathscr{L}_o$:

$$
\mathscr{L}^k_o = \sum_{i\neq j}^{n_l}\frac{\left|\mathbf{o}^k_i \cap \mathbf{o}^k_j\right|}{\left|\mathbf{o}^k_i \cup \mathbf{o}^k_j\right|} \tag{10}
$$

where $\left|\mathbf{o}^k_i \cap \mathbf{o}^k_j\right|$ and $\left|\mathbf{o}^k_i \cup \mathbf{o}^k_j\right|$ denote the intersection area and the union area of the two label boxes, respectively.

Finally, we combine the three losses to jointly guide the training of our LPGT:

$$
\mathscr{L}^k = \mathscr{L}^k_\delta + \lambda_d\mathscr{L}^k_d + \lambda_o\mathscr{L}^k_o \tag{11}
$$

where $\lambda_d, \lambda_o \geq 0$ control the relative importance of the distance loss $\mathscr{L}_d$ and the overlap loss $\mathscr{L}_o$, respectively. The loss function $\mathscr{L}$ is adjusted to accommodate the current training phase.

4.4 Implementation Details

The iterative displacement estimation employs four steps, $n_s = 4$. To construct the features for nodes and edges, we adopt ResNet-101 [27] pre-trained on ImageNet [17] as the feature backbone. The CNN feature maps of layers `conv4-23` and `conv5-3` are concatenated to form the features $\mathbf{F}$. The spatial sizes of the input images are set to $H = W = 256$.

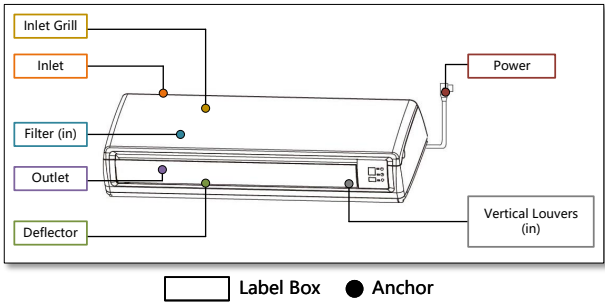

Fig. 7: An illustration with its annotations, including appliance category (*i.e.*, 'air conditioner'), label texts, label boxes, anchors, and leader lines.

Therefore, the dimension of the deep features of nodes and edges is $d_f = 3072$. Many illustrations in the AMIL dataset are line drawings with limited texture information. To extract more effective features, we apply a Gaussian blur to the images before feeding them into the backbone. The patch sizes of the feature aligning strategy are set to $h^r = w^r = 5$. We empirically set the number of Transformer layers to $n_t = 3$, based on the size of the input graphs. The hidden dimensions of the node and edge representations are both set to $d_z = d_e = 192$. The dimensions of the QKV vectors in the node and edge MHSA blocks are both set to $d_{qkv} = 16$. The number of heads and the output dimensions of the two MHSA blocks are both $n_{\mathrm{H}}^z = n_{\mathrm{H}}^e = 12$ and $d_o^z = d_o^e = 192$. The hidden dimensions of the node and edge FFN blocks are set to $d_h^z = 64$ and $d_h^e = 32$, respectively. We use a grid search to set the parameters of losses, $\lambda_d = 0.2$ and $\lambda_o = 0.008$.

5 AMIL Dataset

Several datasets have been used in previous studies to evaluate the effectiveness of label placement methods [6, 15, 55]. However, due to limited data sources and quality, most of these datasets are either relatively small in scale or comprise data samples collected in controlled laboratory environments. The lack of more appropriate benchmarks poses difficulties for learning-based methods, and also partially limits a deeper look into the interactions between labels.

To address this issue, we build a new dataset named the Appliance Manual Illustration Labels dataset. High-quality label layouts typically exhibit four key characteristics: readability, unambiguity, compactness, and aesthetics [2]. For commercial purposes, illustrations from household appliance manuals meet these high standards. To help end-users better understand the operations and functions of household appliances, manufacturers often include illustrations in product manuals. The labels in these illustrations play a crucial role in explaining the functionality of each component, *e.g.*, the on/off button, thus mitigating the risk of user injury due to incorrect operation. Given these requirements, the illustrations are typically created by professional designers, rendering them a valuable resource for our study. Thus motivated, we manually extract useful label placement information from each illustration as depicted in Fig. 7, including appliance category, label texts, label boxes, anchors, and leader lines.

Table 2 presents the statistics of AMIL. It contains 869 label layouts across 11 appliance categories, including 'air conditioner', 'dishwasher', 'disinfection cabinet', 'fridge', 'gas stove', 'oven', 'panel', 'remote', 'washer', 'water heater', and 'water purifier'. These layouts are split into 695 / 174 samples for training and testing, respectively. On average, each sample contains 9.0 labels, ranging from 1 to 27. In addition, the dataset offers a relatively balanced distribution between samples with fewer than 10 labels (529 samples) and those with 10 or more (340 samples), with a ratio of 1.56 : 1. Typically, each label is associated with an anchor, but multiple instances of an element each have their own anchor. In the AMIL dataset, 605 samples feature labels each with a single anchor, while the remaining 264 samples include at least one label with multiple anchors, resulting in a ratio of 2.29 : 1.

We also introduce a metric, Probability of Correct Keypoints (PCK) [71], to evaluate the quality of label layouts. PCK is commonly used to assess the accuracy of semantic correspondence [12, 52].

Table 2: Data statistics of the AMIL dataset.

Category	#Samples	#Labels	$\#\text{Labels}_{\text{avg}}$	$\#\text{Labels}_{\text{max}}$	$\#\text{Labels}_{\text{min}}$
air conditioner	185	1,035	5.6	18	1
dishwasher	21	150	7.1	12	3
disinf. cabinet	4	40	10.0	15	5
fridge	267	2,734	10.2	24	2
gas stove	35	276	7.9	11	4
oven	2	17	8.5	9	8
panel	129	1,175	9.1	20	1
remote	98	1,454	14.8	27	5
washer	9	62	6.9	8	5
water heater	61	266	4.4	12	2
water purifier	58	600	10.3	25	2
Total	869	7,809	9.0	27	1

We use it to measure the accuracy of predicted label positions. The successful placement of a label is determined by the difference between its predicted position $\hat{\mathbf{p}}_i$ and ground-truth position $\mathbf{p}_i$. Specifically, PCK is defined as the ratio of successfully placed labels to all labels:

$$\mathrm{PCK} = \frac{1}{n_l} \sum_{i=1}^{n_l} \mathbb{1}\left(\|\hat{\mathbf{p}}_i - \mathbf{p}_i\| \leq \tau \cdot \max\left(h_l, w_l\right)\right) \tag{12}$$

The indicator function $\mathbb{1}$ returns 1 if the distance between the two label positions $(\hat{\mathbf{p}}_i, \mathbf{p}_i)$ is smaller than $\tau \cdot \max(h_l, w_l)$, *i.e.*, label i is successfully placed. In (12), h_l and w_l are the height and width of the label layout, and $\tau = 0.05$ is the threshold tolerance factor.

6 Experiments

We evaluate the performance of LPGT by comparing it with 11 state-of-the-art GNN models and 4 label placement methods on the proposed AMIL dataset. We also conduct a user study and ablation studies for more evaluation and analysis. For all experiments, optimization is achieved via AdamW optimizer [31] with hyperparameters $\varepsilon =$ 1e-8, $(\beta_1, \beta_2) = (0.9, 0.999)$, and weight decay 0.01. The initial learning rate is set to 2e-4, and halved at regular intervals. The batch size is set to 4. All experiments are run on a single NVIDIA GeForce RTX 4090 GPU and an Intel Core i9-13900k CPU. LPGT and other GNN models are implemented using PyTorch [46] and PyTorch Geometric library [21].

6.1 Evaluation Results

LPGT is compared with 11 GNNs, including GCN [32], MPNN [22], GraphSage [25], GAT [61], GIN [69], GCN2 [10], UniMP [53], EGC [54], GATv2 [8], GPS [50], and ASC [24]. For a fair comparison, we keep the graph construction stage and replace the proposed GNN with these GNNs in the displacement estimation stage. The required settings are listed below:

- Some GNNs (GCN, MPNN, GIN and GCN2) have difficulty placing labels due to the over-smoothing problem and fail to distinguish between different nodes. To solve this problem, we incorporate the initial node features into the input representations of each GNN layer, and add residual connections after each layer.
- Several GNNs are implemented based on attention mechanism or Transformer architecture, including (1) GAT, EGC, and GATv2 (attention-based) and (2) UniMP and GPS (Transformer-based). For these GNNs, we set the number of heads in multi-head attention to 12, which is consistent with our LPGT.
- GCN layers are used for the message passing layers in both GPS and ASC.
- Most GNNs, except for MPNN, GAT, UniMP, and GATv2, focus only on node representations. For these models, we implement encoders with only the node fusion function and decoders with only the node output layer. While MPNN, GAT, UniMP, and GATv2 incorporate edge features into the learning of node representations, there are no independent outputs for edges. Thus, for these four models, we adopt the encoder of our LPGT but continue to use decoders that contain only the node output layer.

Table 3: Comparison of label layout quality PCK@0.05 (%) and efficiency on the AMIL dataset. "Time" indicates the average generation time of a label layout. "#Param" indicates the number of parameters in a GNN model, *i.e.*, without the feature backbone. Time statistics are over all test samples. Numbers in bold indicate the best performance.

Model	ac	d.wash	d.cabi	fridge	g.stove	oven	panel	remote	washer	w.heat	w.puri	Avg.	Time(ms)	#Param
GCN [32]	34.68	64.34	36.67	58.45	57.80	50.00	51.53	42.07	10.00	77.64	19.76	48.75	**7.45**	2.8 M
MPNN [22]	31.09	59.98	20.00	60.84	53.71	37.50	49.47	39.00	14.29	65.97	13.77	46.41	11.87	33.8 M
GraphSAGE [25]	37.03	55.63	33.33	61.24	57.71	25.00	50.42	39.62	21.43	76.81	18.57	49.25	7.53	3.3 M
GAT [61]	25.10	53.54	23.33	54.84	58.68	0.00	42.32	40.23	0.00	71.67	17.26	42.74	13.04	5.6 M
GIN [69]	25.91	56.45	0.00	33.59	35.56	50.00	25.59	12.20	0.00	20.14	15.45	26.20	7.48	3.3 M
GCN2 [10]	37.20	58.69	23.33	62.06	59.16	12.50	52.00	43.38	10.00	73.19	17.40	49.69	7.91	2.8 M
UniMP [53]	37.96	60.17	46.67	63.34	60.59	50.00	56.96	45.58	17.14	74.17	14.65	51.78	12.62	7.0 M
EGC [54]	35.40	55.44	43.33	64.97	65.55	62.50	48.97	41.85	10.00	80.83	15.55	50.68	7.69	**2.6** M
GATv2 [8]	27.14	60.50	40.00	54.61	48.43	25.00	50.87	44.10	0.00	64.31	9.65	43.91	12.07	6.1 M
GPS [50]	20.43	38.07	13.33	41.19	43.99	37.50	37.47	31.45	17.14	30.42	14.70	32.23	10.95	6.4 M
ASC [24]	36.00	64.53	36.67	62.56	60.73	37.50	50.96	42.69	20.00	77.92	15.65	50.18	8.61	3.3 M
LPGT	**42.05**	**76.81**	**46.67**	**72.67**	**73.51**	**87.50**	**57.30**	**50.25**	**24.29**	**85.14**	**34.57**	**59.27**	18.46	11.4 M

For all GNNs, we compute distances between the estimated node positions to replace the predicted distances of the edge output layer, *i.e.*, $\hat{\mathbf{d}}_{ij}^k = \tilde{\mathbf{p}}_i^k - \tilde{\mathbf{p}}_j^k$.

Moreover, we compare LPGT with four label placement methods, including Particle-Based Labeling (PBL) [41], Clutter-Aware Labeling (CAL) [42], SmartOverlays [28], and Semantic-Aware Labeling (SAL) [29]. Since there are no official codes available, we reproduce these methods based on their papers and adjust the parameter settings.

Quantitative Results The quantitative results, presented in Tab. 3, show that LPGT surpasses other GNN models with a placement accuracy of 59.27%, leading in all categories. Eight models—GCN, MPNN, GraphSAGE, GAT, GIN, GCN2, GATv2, and GPS—achieve below 50% PCK, indicating that, on average, they incorrectly place more than half of the labels. While the other three models—UniMP, EGC, and ASC—generate better label layouts than the aforementioned eight, they fall significantly short of LPGT, with gaps in PCK values of 7.49%, 8.59%, and 9.09%, respectively. Despite attempts to optimize parameters for the four label placement methods PBL, CAL, SmartOverlays, and SAL, their performances remain unsatisfactory, with PCK values of 3.61%, 1.98%, 7.14%, and 3.52%, respectively. This underperformance likely stems from their design for maps and street views, not well-suited to the illustrations in the AMIL dataset.

Efficiency Evaluation Further experiments assess the efficiency of LPGT by profiling the time to generate label layouts and the number of model parameters (Tab. 3). We report the parameter count excluding the feature backbone, as it is consistent across all models. To determine layout generation time, we calculate the average time for all test samples. MPNN, GAT, UniMP, GATv2, and LPGT show lower efficiency than other GNNs, attributed to their use of edge features. LPGT, with its relatively high parameter count, aligns with our expectations due to the complex node-to-node and edge-to-edge attention mechanisms it incorporates. Despite LPGT's slower label layout generation compared to other GNNs, its efficiency remains suitable for graphic design.

Qualitative Results The predicted label layouts of several illustrations are visualized in Fig. 8. Compared to GCN, GIN, GATv2, and SmartOverlays, LPGT produces more appropriate layouts with little sensitivity to label quantity. Most of the label positions predicted by the four comparison methods are inappropriate, even overlapping with objects such as the remote sample (the seventh row). GCN outperforms the other three methods, but there is still a gap between it and LPGT in terms of label position accuracy.

6.2 User Study

Along with the above qualitative results, we conduct a user study to explore user preferences for different methods, comparing LPGT with GCN, GIN, GATv2, and SmartOverlays. We select a sample from each category in the test set of the AMIL dataset, applying all five methods to generate label layouts. On average, each sample contains 6.5 labels, ranging from 3 to 15. This provides a diverse basis for evaluation across label quantity and types.

Utilizing a psychophysical paired comparison technique [59] and following the two-alternative forced choice (2AFC) paradigm, we sequentially present each participant with label layout pairs generated by different methods for the same sample, asking them to select their preferred layout. Each sample has 10 label layout pairs. To reduce learning effects and fatigue, the sequence and side placement of layouts are randomized. The study involves 22 participants, consisting of 11 males and 11 females, with an average age of 20.18 years (ranging from 19 to 23). They were recruited through a university forum and were unaware of any of the evaluated methods. Each participant compared 110 label layout pairs, resulting in 88 scores for each layout. These measures ensure sufficient and reliable statistical data in the user study. The participants completed the task in an average time of 21 minutes and 39 seconds.

Each participant's choices are quantified in a count matrix $\mathbf{C}$, then converted into a quality score (z-score) scale for statistical significance analysis based on Thurstone's Law of Comparative Judgment (Case V) [48, 59]. Each element $\mathbf{C}_{ij}$ indicates the preference frequency for method i over method j. A two-tailed test at a significance level of $\alpha = 0.05$ assesses the null hypothesis of *no clear user preference among the evaluated methods*. The results, illustrated in Fig. 9, demonstrate LPGT's superiority with the highest quality score, significantly surpassing other methods, thus preferred by users. There are significant preference differences between SmartOverlays and other four methods, highlighting the value of developing GNN-driven label placement.

6.3 Ablation Study

We conduct ablation studies to shed light on the impact of each key design within LPGT. Additionally, the findings from these ablations open up the potential pathways for future research, creating possibilities for further optimization of our method.

Key Components First, we implement a model without the **f**eature **a**ligning (FA) strategy. The deep features of nodes and edges are recovered directly from the feature maps as mentioned in Sec. 4.1. The results in Tab. 4 show that the performance improves by 4.18% from **(b)** to **(a)**. Then, we define two other models: (1) one without the **e**dge **f**eature **e**xtraction (EFE) strategy; (2) the other without the **e**dge-level **a**ttention (EA). For the former, the deep features of each edge are computed as the differences of the features of its associated nodes: $\mathbf{f}_{ij}^e = \mathbf{f}_i^v - \mathbf{f}_j^v$. For the latter, we aggregate the local information of the edge to replace the edge-level attention. The representations of neighbor edges of the edge are summed, then concatenated with the associated node representations. Next, these representations are aggregated by two linear layers with a ReLU non-linearity:

$$\mathbf{e}'_{ij} = \mathrm{ReLU}\left(\left[\sum_{(k,l)\in\mathbb{N}_{ij}} \mathbf{e}_{kl}, \mathbf{z}''_i, \mathbf{z}''_j\right]\mathbf{W}_5\right)\mathbf{W}_6 + \mathbf{e}_{ij} \tag{13}$$

where $\mathbf{W}_5 \in \mathbb{R}^{(d_e+2d_z)\times d_e}$ and $\mathbf{W}_6 \in \mathbb{R}^{d_e\times d_e}$ are the weight matrices used for the aggregation of the edge representations. The results in Tab. 4 show that the performance decreases by 1.13% (from **(a)**

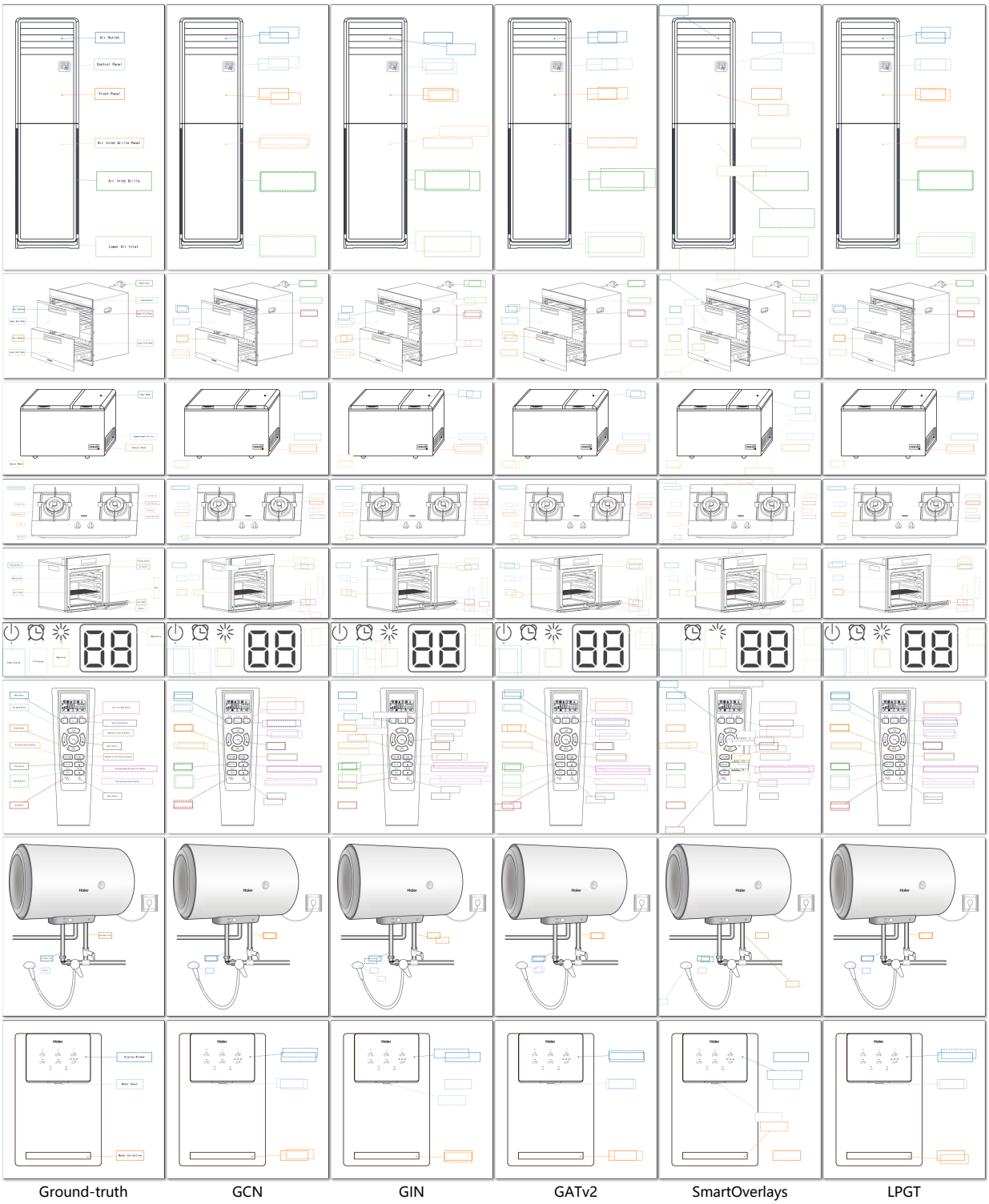

Fig. 8: Qualitative results on the AMIL dataset. Solid and dashed label boxes denote the ground-truth and predicted positions of labels, respectively. Different colors are used to distinguish labels. Label texts are omitted in the predicted label layouts for clarity.

to **(c)**) and 2.25% (from **(a)** to **(d)**), respectively, when the two components are removed. The above results demonstrate the effectiveness of the three key components in exploring the label interactions.

Loss Functions To evaluate the effectiveness of each loss function, we first train a model using only the displacement loss $\mathscr{L}_\delta$. Then, two other models are trained: (1) one without the distance loss $\mathscr{L}_d$; (2) the other without the overlap loss $\mathscr{L}_o$. The PCK 56.94% of **(e)** in Tab. 4 demonstrates that the displacement loss is effective to guide the learning of LPGT. Besides, the PCK improvements from **(f)** to **(a)** (1.74%) and from **(g)** to **(a)** (1.07%) suggest that the distance loss and the overlap loss guide the labels to pursue appropriate spatial relations and avoid overlapping each other.

Visual Features We consider the geometric features as essential for learning label positions. To achieve more ideal label layouts, we introduce the visual features based on graphic appearance. The aforementioned experimental findings demonstrate that the combination of

Table 4: Ablation studies of LPGT on the AMIL dataset. Time statistics are over the same setting with Tab. 3. "−" indicates that the component is removed. "#Param" indicates the number of parameters in the whole model. Numbers in bold indicate the best performance.

Model	ac	d.wash	d.cabi.	fridge	g.stove	oven	panel	remote	washer	w.heat	w.puri	Avg.	Time(ms)	#Param
(a) LPGT	42.05	76.81	46.67	72.67	73.51	**87.50**	57.30	**50.25**	24.29	85.14	**34.57**	**59.27**	18.46	53.9 M
(b) − FA	40.29	79.00	40.00	70.05	68.83	62.50	48.88	43.00	14.29	84.86	27.73	55.09	17.53	53.9 M
(c) − EFE	45.12	70.97	**46.67**	72.04	70.62	**87.50**	56.22	41.62	10.00	87.22	32.21	58.14	15.26	53.9 M
(d) − EA	44.45	76.81	43.33	71.99	72.66	**87.50**	53.42	44.47	10.00	79.17	24.25	57.02	15.47	52.6 M
(e) $-\mathscr{L}_d - \mathscr{L}_o$	39.19	71.54	46.67	73.10	67.90	75.00	52.99	48.79	7.14	86.94	25.56	56.94	18.46	53.9 M
(f) $-\mathscr{L}_d$	44.00	78.49	40.00	71.70	68.70	62.50	54.77	43.36	17.14	**88.61**	27.20	57.53	18.46	53.9 M
(g) $-\mathscr{L}_o$	41.94	71.86	43.33	**73.24**	69.81	62.50	54.32	47.72	7.14	**88.61**	33.19	58.20	18.46	53.9 M
(h) − VF	38.16	46.32	43.33	65.78	**76.63**	25.00	55.08	39.73	35.71	77.78	9.43	51.89	**11.20**	**6.7** M
(i) − MPT	34.60	72.84	43.33	68.23	64.33	62.50	54.43	49.24	7.14	80.56	27.27	54.22	18.46	53.9 M
(j) − BF	42.76	78.11	43.33	69.06	70.81	75.00	**58.45**	43.25	**52.86**	83.89	27.14	57.27	18.46	11.4 M
(k) − GB	**47.06**	**79.79**	43.33	71.25	71.01	62.50	54.21	44.82	24.29	80.69	27.46	57.82	18.46	53.9 M

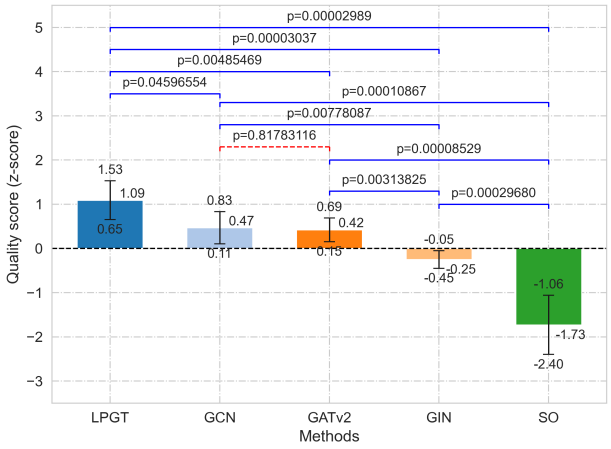

Fig. 9: Results of the user study: quality scores with 95% confidence intervals. Solid blue brackets highlight statistically significant disparities between method pairs, annotated with p-values, while red dashed brackets indicate method pairs lacking statistically significant differences.

both features is effective for label placement. To further analyze the role of each feature, we conduct experiments by eliminating the **v**isual **f**eatures (VF). As shown in Tab. 4, despite a decrease in the PCK value from **(a)** to **(h)**, the model still achieves favorable performance compared to other GNNs listed in Tab. 3. These results indicate that the geometric features support the generation of satisfactory label layouts, and the visual features are effective for further optimization.

Training Strategies In the proposed **m**ulti-**p**hase **t**raining (MPT) strategy, the first k GNN modules of LPGT are trained at the k-th phase. To study its effect on the optimization of LPGT, we present another strategy for comparison. In the k-th phase, only the k-th GNN module is trained. The obvious performance drop from **(a)** to **(i)** in Tab. 4 demonstrates the effectiveness of MPT. In addition, we implement two other models: (1) one without **b**ackbone **f**ine-tuning (BF); (2) the other without **G**aussian **b**lur (GB). The PCK improvements from **(j)** to **(a)** (2.00%) and from **(k)** to **(a)** (1.45%) in Tab. 4 suggest that both approaches have positive effects on extracting more effective features.

6.4 Summary

The quantitative comparison demonstrates that LPGT is effective and robust for placing varying numbers of labels across different object categories. In terms of aesthetics, LPGT achieves appropriate label layouts with less overlapping and clutter, as shown in the qualitative results. The user study suggests that participants prefer LPGT-generated layouts over those from other methods, aligning with both the quantitative and qualitative results. In addition, the user preference differences between GNN models and traditional label placement methods highlight the power of GNNs in label placement. Based on the findings that variations in the key designs of LPGT influence overall performance, the ablation studies indicate the effectiveness of these designs.

7 Discussion

It is worth noting that although our learning-based solution is evaluated on graphics, it may be applied to other labeling tasks with appropriate extensions. For interactive graph labeling, the changeable label layout can be represented by a dynamic graph instead of a static one. Additionally, the learning of node and edge representations should incorporate temporal information by weighting updates based on the recency and frequency of user interactions. Similarly, our method should be generalizable to other cases beyond our focus (*i.e.*, external labels). For internal labels, the iterations of the displacement estimation need to be reduced since the labels are generally placed within the target regions. For mixed labeling, the intrinsic differences between internal and external labels result in diverse node types and more complex relationships between labels, which requires a heterogeneous graph.

Based on the rich annotation and the quality metric, the AMIL dataset could be utilized to train and evaluate learning-based models, thus contributing to significant advances in label placement. However, it still has some limitations. First, the labels are not highly dense and crowded, thus it is not suited to evaluate the performance of methods that focus on placing dense labels. Second, the sample size of several categories is small, *e.g.*, 'oven'. This requires a pre-training procedure for learning-based approaches that depend on large-scale data. In addition, some limitations of our method need to be explained despite its promising performance in label placement. First, to reduce the computational complexity of edge-level attention, we restrict each edge to attend to its neighbor edges. In future work, we will explore more efficient approaches to expand each edge's receptive field to cover the entire graph. Second, our method focuses on label position prediction, and directly uses standard leader lines to connect labels to anchors. It would be worthwhile to develop learning-based leader line generation approaches to improve the level of automation in label placement.

8 Conclusion

In this paper, we model label layouts with graph representation and formulate label placement as a graph node prediction problem. Based on the formulation, we develop a Label Placement Graph Transformer network, which iteratively predicts and refines label positions. Specifically, in LPGT, an edge-level attention mechanism conditioned on node representations is designed and combined with a node-level attention to explore the interactions between labels. Furthermore, we propose a feature aligning strategy to effectively integrate graphic/image information into node and edge features. The edge features are specially constructed to enhance the descriptions of the label relationships, based on the spatial traversing property of edges. In addition, a benchmark dataset containing high-quality label layouts from product illustrations is created. Experimental results demonstrate the effectiveness of the proposed method.

References

[1] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proc. of ICML*, pp. 21–29. PMLR, New York, 2019. doi: 10.48550/arXiv.1905.00067 2

[2] K. Ali, K. Hartmann, and T. Strothotte. Label layout for interactive 3d illustrations. *Journal of World Society for Computer Graphics*, 13(1):1–8, 2005. 1, 6

[3] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. doi: 10.48550/arXiv.1806.01261 2

[4] M. A. Bekos, M. Kaufmann, M. Nöllenburg, and A. Symvonis. Boundary labeling with octilinear leaders. *Algorithmica*, 57(3):436–461, 2010. doi: 10.1007/S00453-009-9283-6 1, 2

[5] M. A. Bekos, B. Niedermann, and M. Nöllenburg. External labeling techniques: A taxonomy and survey. *Comput. Graph. Forum*, 38(3):833–860, 2019. doi: 10.1111/CGF.13729 2

[6] P. Bobák, L. Čmolík, and M. Čadík. Reinforced Labels: Multi-agent deep reinforcement learning for point-feature label placement. *IEEE Trans. Vis. Comput. Graph.*, 2023. doi: 10.1109/TVCG.2023.3313729 1, 2, 6

[7] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965. doi: 10.1147/SJ.41.0025 4

[8] S. Brody, U. Alon, and E. Yahav. How attentive are graph attention networks? In *Proc. of ICLR*, 2022. doi: 10.48550/arXiv.2105.14491 2, 6, 7

[9] I. Chami, Z. Ying, C. Ré, and J. Leskovec. Hyperbolic graph convolutional neural networks. In *Proc. of NeurIPS*, pp. 4869–4880. MIT Press, Cambridge, 2019. doi: 10.1145/3580305.3599562 2

[10] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. Simple and deep graph convolutional networks. In *Proc. of ICML*, pp. 1725–1735. PMLR, New York, 2020. doi: 10.48550/arXiv.2007.02133 6, 7

[11] Z. Chen, D. Chiappalupi, T. Lin, Y. Yang, J. Beyer, and H. Pfister. RL-LABEL: A deep reinforcement learning approach intended for ar label placement in dynamic scenarios. *IEEE Trans. Vis. Comput. Graph.*, 30(1):1347–1357, 2024. doi: 10.1109/TVCG.2023.3326568 1, 2

[12] S. Cho, S. Hong, and S. Kim. CATs++: Boosting cost aggregation with convolutions and transformers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(6):7174–7194, 2023. doi: 10.1109/TPAMI.2022.3218727 6

[13] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Trans. Graph.*, 14(3):203–232, 1995. doi: 10.1145/212332.212334 1

[14] L. Čmolík and J. Bittner. Real-time external labeling of ghosted views. *IEEE Trans. Vis. Comput. Graph.*, 25(7):2458–2470, 2018. doi: 10.1109/TVCG.2018.2833479 2

[15] L. Čmolík, V. Pavlovec, H.-Y. Wu, and M. Nöllenburg. Mixed Labeling: Integrating internal and external labels. *IEEE Trans. Vis. Comput. Graph.*, 28(4):1848–1861, 2020. doi: 10.1109/TVCG.2020.3027368 2, 6

[16] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. of NeurIPS*, pp. 3837–3845. MIT Press, Cambridge, 2016. doi: 10.48550/arXiv.1606.09375 2

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. of CVPR*, pp. 248–255. IEEE, New York, 2009. doi: 10.1109/CVPR.2009.5206848 5

[18] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*, pp. 4171–4186. ACL, Stroudsburg, 2019. doi: 10.18653/V1/N19-1423 2

[19] C. Diao and R. Loynd. Relational Attention: Generalizing transformers for graph-structured tasks. In *Proc. of ICLR*, 2023. doi: 10.48550/arXiv.2210.05062 1, 2, 4

[20] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of ICLR*, 2021. doi: 10.48550/arXiv.2010.11929 2

[21] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *Proc. of ICLR*, 2019. doi: 10.48550/arXiv.1903.02428 6

[22] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proc. of ICML*, pp. 1263–1272. PMLR, New York, 2017. doi: 10.48550/arXiv.1704.01212 2, 6, 7

[23] R. Grasset, T. Langlotz, D. Kalkofen, M. Tatzgern, and D. Schmalstieg. Image-driven view management for augmented reality browsers. In *Proc. of ISMAR*, pp. 177–186. IEEE, New York, 2012. doi: 10.1109/ISMAR.2012.6402555 2

[24] A. Gravina, D. Bacciu, and C. Gallicchio. Anti-Symmetric DGN: a stable architecture for Deep Graph Networks. In *Proc. of ICLR*, 2023. doi: 10.48550/arXiv.2210.09789 6, 7

[25] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proc. of NeurIPS*, pp. 1024–1034. MIT Press, Cambridge, 2017. doi: 10.48550/arXiv.1706.02216 2, 6, 7

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proc. of ICCV*, pp. 2961–2969. IEEE, New York, 2017. doi: 10.1109/ICCV.2017.322 3

[27] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, pp. 770–778. IEEE, New York, 2016. doi: 10.1109/CVPR.2016.90 2, 3, 5

[28] S. Hegde, J. Maurya, A. Kalkar, and R. Hebbalaguppe. SmartOverlays: A visual saliency driven label placement for intelligent human-computer interfaces. In *Proc. of WACV*, pp. 1121–1130. IEEE, New York, 2020. doi: 10.1109/WACV45572.2020.9093587 2, 7

[29] J. Jia, S. Elezovikj, H. Fan, S. Yang, J. Liu, W. Guo, C. C. Tan, and H. Ling. Semantic-aware label placement for augmented reality in street view. *The Vis. Comput.*, 37:1805–1819, 2021. doi: 10.1007/S00371-020-01939-W 7

[30] P. Kindermann, B. Niedermann, I. Rutter, M. Schaefer, A. Schulz, and A. Wolff. Multi-sided boundary labeling. *Algorithmica*, 76:225–258, 2016. doi: 10.1007/S00453-015-0028-4 1, 2

[31] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of ICLR*, 2015. doi: 10.48550/arXiv.1412.6980 6

[32] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proc. of ICLR*, 2017. doi: 10.48550/arXiv.1609.02907 2, 6, 7

[33] D. Kouřil, L. Čmolík, B. Kozlíková, H.-Y. Wu, G. Johnson, D. S. Goodsell, A. Olson, M. E. Gröller, and I. Viola. Labels on Levels: Labeling of multi-scale multi-instance and crowded 3d biological environments. *IEEE Trans. Vis. Comput. Graph.*, 25(1):977–986, 2018. doi: 10.1109/TVCG.2018.2864491 1, 2

[34] D. Kouřil, T. Isenberg, B. Kozlíková, M. Meyer, M. E. Gröller, and I. Viola. HyperLabels: Browsing of dense and hierarchical molecular 3D models. *IEEE Trans. Vis. Comput. Graph.*, 27(8):3493–3504, 2020. doi: 10.1109/TVCG.2020.2975583 1, 2

[35] E. Kruijff, J. Orlosky, N. Kishishita, C. Trepkowski, and K. Kiyokawa. The influence of label design on search performance and noticeability in wide field of view augmented reality displays. *IEEE Trans. Vis. Comput. Graph.*, 25(9):2821–2837, 2018. doi: 10.1109/TVCG.2018.2854737 2

[36] B. Lee, M. Sedlmair, and D. Schmalstieg. Design patterns for situated visualization in augmented reality. *IEEE Trans. Vis. Comput. Graph.*, 30(1):1324–1335, 2024. doi: 10.1109/TVCG.2023.3327398 2

[37] T. Lin, Z. Chen, Y. Yang, D. Chiappalupi, J. Beyer, and H. Pfister. The quest for omnioculars: Embedded visualization for augmenting basketball game viewing experiences. *IEEE Trans. Vis. Comput. Graph.*, 29(1):962–971, 2023. doi: 10.1109/TVCG.2022.3209353 1

[38] T. Lin, Y. Yang, J. Beyer, and H. Pfister. Labeling out-of-view objects in immersive analytics to support situated visual searching. *IEEE Trans. Vis. Comput. Graph.*, 29(3):1831–1844, 2023. doi: 10.1109/TVCG.2021.3133511 2

[39] M. Liu, H. Gao, and S. Ji. Towards deeper graph neural networks. In *Proc. of SIGKDD*, pp. 338–348. ACM, New York, 2020. doi: 10.1145/3394486.3403076 2

[40] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin Transformer: Hierarchical vision transformer using shifted windows. In *Proc. of ICCV*, pp. 10012–10022. IEEE, New York, 2021. doi: 10.1109/ICCV48922.2021.00986 2

[41] M. Luboschik, H. Schumann, and H. Cords. Particle-based labeling: Fast point-feature labeling without obscuring other visual features. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1237–1244, 2008. doi: 10.1109/TVCG.2008.152 1, 7

[42] Y. Meng, H. Zhang, M. Liu, and S. Liu. Clutter-aware label layout. In *Proc. of PacificVis*, pp. 207–214. IEEE, New York, 2015. doi: 10.1109/PACIFICVIS.2015.7156379 7

[43] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman Go Neural: Higher-order graph neural networks. In *Proc. of AAAI*, pp. 4602–4609. AAAI, Menlo Park, 2019. doi: 10.1609/AAAI.V33I01.33014602 2

[44] H. Nguyen, S. Ketchell, U. Engelke, B. Thomas, and P. de Souza. HoloBee: Augmented reality based bee drift analysis. In *Proc. of ISMAR*, pp. 87–92. IEEE, New York, 2017. doi: 10.1109/ISMAR-ADJUNCT.2017.38 2

[45] J. Orlosky, K. Kiyokawa, T. Toyama, and D. Sonntag. Halo Content: Context-aware viewspace management for non-invasive augmented reality. In *Proc. of IUI*, pp. 369–373. ACM, New York, 2015. doi: 10.1145/2678025.2701375 2

[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Proc. of NeurIPS*, pp. 8024–8035. MIT Press, Cambridge, 2019. doi: 10.48550/arXiv.1912.01703 6

[47] V. Pavlovec and L. Čmolík. Rapid Labels: Point-feature labeling on gpu. *IEEE Trans. Vis. Comput. Graph.*, 28(1):604–613, 2021. doi: 10.1109/TVCG.2021.3114854 1, 2

[48] M. Pérez-Ortiz and R. K. Mantiuk. A practical guide and software for analysing pairwise comparison experiments. *CoRR*, abs/1712.03686, 2017. doi: 10.48550/arXiv.1712.03686 7

[49] J. Qian, Q. Sun, C. Wigington, H. L. Han, T. Sun, J. Healey, J. Tompkin, and J. Huang. Dually Noted: Layout-aware annotations with smartphone augmented reality. In *Proc. of CHI*, pp. 1–15. ACM, New York, 2022. doi: 10.1145/3491102.3502026 2

[50] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. In *Proc. of NeurIPS*, pp. 14501–14515. MIT Press, Cambridge, 2022. doi: 10.48550/arXiv.2205.12454 2, 6, 7

[51] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, 2016. doi: 10.1109/TPAMI.2016.2577031 5

[52] I. Rocco, R. Arandjelovic, and J. Sivic. Convolutional neural network architecture for geometric matching. In *Proc. of CVPR*, pp. 6148–6157. IEEE, New York, 2017. doi: 10.1109/CVPR.2017.12 6

[53] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun. Masked Label Prediction: Unified message passing model for semi-supervised classification. In *Proc. of IJCAI*, pp. 1548–1554. IJCAI, 2021. doi: 10.24963/IJCAI.2021/214 2, 6, 7

[54] S. A. Tailor, F. L. Opolka, P. Lio, and N. D. Lane. Adaptive filters and aggregator fusion for efficient graph convolutions. *CoRR*, abs/2104.01481, 2021. doi: 10.48550/arXiv.2104.01481 2, 6, 7

[55] M. Tatzgern, D. Kalkofen, R. Grasset, and D. Schmalstieg. Hedgehog Labeling: View management techniques for external labels in 3D space. In *Proc. of VR*, pp. 27–32. IEEE, New York, 2014. doi: 10.1109/VR.2014.6802046 2, 6

[56] M. Tatzgern, D. Kalkofen, and D. Schmalstieg. Dynamic compact visualizations for augmented reality. In *Proc. of VR*, pp. 3–6. IEEE, New York, 2013. doi: 10.1109/VR.2013.6549347 2

[57] M. Tatzgern, V. Orso, D. Kalkofen, G. Jacucci, L. Gamberini, and D. Schmalstieg. Adaptive information density for augmented reality displays. In *Proc. of VR*, pp. 83–92. IEEE, New York, 2016. doi: 10.1109/VR.2016.7504691 2

[58] W. Tong, M. Xia, K. K. Wong, D. A. Bowman, T.-C. Pong, H. Qu, and Y. Yang. Towards an understanding of distributed asymmetric collaborative visualization on problem-solving. In *Proc. of VR*, pp. 387–397. IEEE, New York, 2023. doi: 10.1109/VR55154.2023.00054 1

[59] K. Tsukida, M. R. Gupta, et al. How to analyze paired comparison data. *UWEE Technical Report 206*, 2011. 7

[60] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. of NeurIPS*, pp. 5998–6008. MIT Press, Cambridge, 2017. doi: 10.48550/arXiv.1706.03762 2

[61] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *Proc. of ICLR*, 2018. doi: 10.48550/arXiv.1710.10903 2, 6, 7

[62] W.-C. Wang, C.-Y. Chiou, C.-R. Huang, P.-C. Chung, and W.-Y. Huang. Spatiotemporal coherence-based annotation placement for surveillance videos. *IEEE Trans. Circuit Syst. Video Technol.*, 28(3):787–801, 2016. doi: 10.1109/TCSVT.2016.2629340 2

[63] X. Wei, X. Shi, and L. Wang. Label guidance based object locating in virtual reality. In *Proc. of ISMAR*, pp. 441–449. IEEE, New York, 2022. doi: 10.1109/ISMAR55827.2022.00060 1, 2

[64] Z. Wen, W. Zeng, L. Weng, Y. Liu, M. Xu, and W. Chen. Effects of view layout on situated analytics for multiple-view representations in immersive visualization. *IEEE Trans. Vis. Comput. Graph.*, 29(1):440–450, 2022. doi: 10.1109/TVCG.2022.3209475 1, 2

[65] J. Woodward and J. Ruiz. Analytic review of using augmented reality for situational awareness. *IEEE Trans. Vis. Comput. Graph.*, 29(4):2166–2183, 2022. doi: 10.1109/TVCG.2022.3141585 2

[66] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(1):4–24, 2020. doi: 10.1109/TNNLS.2020.2978386 1, 2

[67] Z. Xinyi and L. Chen. Capsule graph neural network. In *Proc. of ICLR*, 2019. Available at https://openreview.net/forum?id=Byl8BnRcYm. 2

[68] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu. On layer normalization in the transformer architecture. In *Proc. of ICML*, pp. 10524–10533. PMLR, New York, 2020. doi: 10.48550/arXiv.2002.04745 2, 4

[69] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *Proc. of ICLR*, 2019. doi: 10.48550/arXiv.1810.00826 6, 7

[70] M. Yamaguchi, S. Mori, P. Mohr, M. Tatzgern, A. Stanescu, H. Saito, and D. Kalkofen. Video-annotated augmented reality assembly tutorials. In *Proc. of UIST*, pp. 1010–1022. ACM, New York, 2020. doi: 10.1145/3379337.3415819 1

[71] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2878–2890, 2012. doi: 10.1109/TPAMI.2012.261 6

[72] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. Do transformers really perform badly for graph representation? In *Proc. of NeurIPS*, pp. 28877–28888. MIT Press, Cambridge, 2021. doi: 10.48550/arXiv.2106.05234 2

[73] P. Yoeli. The logic of automated map lettering. *The Cartographic Journal*, 9(2):99–108, 1972. 1

[74] H. Yuan, H. Yu, S. Gui, and S. Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(5):5782–5799, 2023. doi: 10.1109/TPAMI.2022.3204236 1, 2

[75] B. Zhang, S. Luo, L. Wang, and D. He. Rethinking the expressive power of gnns via graph biconnectivity. In *Proc. of ICLR*, 2023. doi: 10.48550/arXiv.2301.09505 1, 2

[76] Z. Zhou, L. Wang, and V. Popescu. A partially-sorted concentric layout for efficient label localization in augmented reality. *IEEE Trans. Vis. Comput. Graph.*, 27(11):4087–4096, 2021. doi: 10.1109/TVCG.2021.3106492 2