Mixture of Cluster-guided Experts for Retrieval-Augmented Label Placement

Pingshun Zhang, Enyu Che, Yinan Chen, Bingyao Huang, Haibin Ling and Jingwei Qu

Abstract—Text labels are widely used to convey auxiliary information in visualization and graphic design. The substantial variability in the categories and structures of labeled objects leads to diverse label layouts. Recent single-model learning-based solutions in label placement struggle to capture fine-grained differences between these layouts, which in turn limits their performance. In addition, although human designers often consult previous works to gain design insights, existing label layouts typically serve merely as training data, limiting the extent to which embedded design knowledge can be exploited. To address these challenges, we propose a mixture of cluster-guided experts (MoCE) solution for label placement. In this design, multiple experts jointly refine layout features, with each expert responsible for a specific cluster of layouts. A cluster-based gating function assigns input samples to experts based on representation clustering. We implement this idea through the Label Placement Cluster-guided Experts (LPCE) model, in which a MoCE layer integrates multiple feed-forward networks (FFNs), with each expert composed of a pair of FFNs. Furthermore, we introduce a retrieval augmentation strategy into LPCE, which retrieves and encodes reference layouts for each input sample to enrich its representations. Extensive experiments demonstrate that LPCE achieves superior performance in label placement, both quantitatively and qualitatively, surpassing a range of state-of-the-art baselines. Our algorithm is available at https://github.com/PingshunZhang/LPCE.

Index Terms—Label placement, Mixture of experts, Retrieval augmentation

1 Introduction

Labels are widely used in graphic design [4] and information visualization [38] to provide auxiliary information for a scene. A label is a brief text annotation connected to its target element by a leader line. The placement of labels, *i.e.*, determining the label layout, is crucial for users' perception of the scene. Therefore, extensive research has explored the automatic layout generation. This ranges from specialized explanations, such as annotations in medical illustrations [33], to visualization guidance, such as locating objects in virtual environments [60].

Label placement has typically been formulated as an optimization problem in previous methods, given its NP-hard nature [12]. These methods design cost functions with specific constraints tailored to scenarios and placement rules. Then, algorithms such as greedy or dynamic programming are employed to search for the optimal solution among possible label layouts [5, 34, 43, 61].

The tremendous success of deep learning in various fields has spurred recent research to focus on learning-based methods [7, 11, 46]. Nonetheless, the progress in addressing two critical challenges remains limited. First, the discrimination of label layout characteristics requires refinement. Target objects exhibit a wide variety of categories and structures, and their label layouts are highly diverse. For learning-based methods, training a single model restricts refinement in capturing differences between layouts, thus hindering performance improvement. Instead of using a single model, employing a separate model to generate each layout would further enhance performance. However, using excessively fine-grained models is impractical. Second, the exploration of label layout data remains insufficient. Designers often refer to existing works within creative workflows to absorb useful insights. However, most current methods utilize existing label layouts only as training samples; even approaches like that of Vollick *et al.* [57], which specify layout styles from a single example, still offer limited design insights and struggle to address the diversity of label layouts.

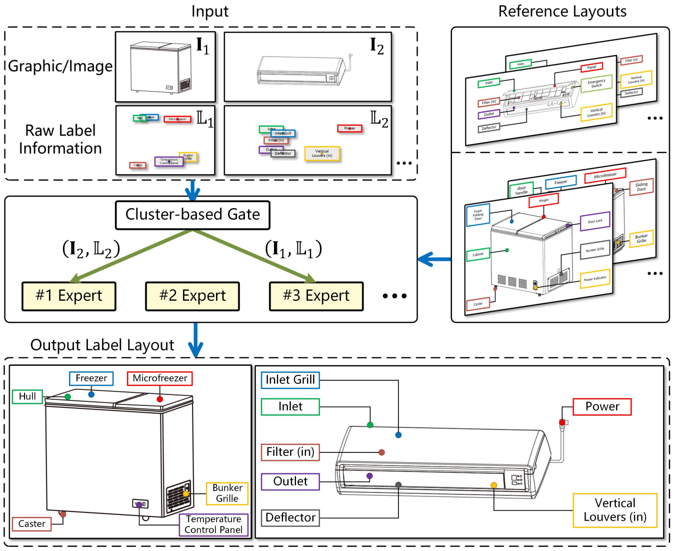


Fig. 1: Conceptual illustration of LPCE: integrating mixture of cluster-guided experts with retrieval augmentation.

In this work, we address both challenges with novel contributions. First, we propose a mixture of cluster-guided experts (MoCE) for label placement (Fig. 1). In this solution, multiple experts are employed to learn features for label layouts, with each expert responsible for a group of layouts. This design enables the capture of more discriminative layout features. A MoCE layer is constructed by integrating multiple feed-forward networks (FFNs) in a graph Transformer block, with each expert consisting of a pair of FFNs. We then design a cluster-based gating function that assigns input samples to the appropriate expert via clustering. Finally, a *Label Placement Cluster-guided Experts* (LPCE) model is formed to predict label layouts.

Second, to better leverage existing layouts, we introduce a retrieval augmentation (RA) strategy into LPCE. For an input sample, reference label layouts are retrieved from a given dataset via nearest neighbor search. These layouts are encoded and then fused with the features

- *Pingshun Zhang, Enyu Che, Yinan Chen, Bingyao Huang and Jingwei Qu are with College of Computer and Information Science, Southwest University. E-mail: z2211973606@email.swu.edu.cn, enyuche@gmail.com, out1147205215@outlook.com, bhuang@swu.edu.cn, qujingwei@swu.edu.cn.*
- *Haibin Ling is with Department of Artificial Intelligence, Westlake University. E-mail: linghaibin@westlake.edu.cn.*
- *Jingwei Qu is the corresponding author.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

of the input sample. The injection of reference layouts improves the quality of label placement.

In summary, our main contributions include:

- We propose a mixture of cluster-guided experts architecture for label placement. Fine-grained characteristics of diverse label layouts are captured by multiple experts, each responsible for a group of layouts. The cluster-based gating function assigns input samples to their matching experts by clustering them.
- We develop a novel Label Placement Cluster-guided Expert model, which we believe to be the first mixture-of-experts (MoE)-based solution for label placement. Multiple FFNs form a MoCE layer, in which a pair of FFNs acts as an expert.
- We present a retrieval augmentation strategy, which injects features from reference layouts into label placement. This approach partially emulates the common practice of human designers drawing inspiration from previous works.
- Thorough experimental results show that LPCE surpasses various state-of-the-art methods in label placement, achieving superior performance both quantitatively and qualitatively.

2 Related Work

2.1 Label Placement

Traditional rule-based methods often treat label placement as an optimization problem, designing cost functions based on specific requirements of different scenarios. Boundary labeling is popular due to its high readability [2,35,37,45,55,62]. It places labels along a predefined boundary around target elements, with shapes such as rectangles [4,31], object contours [14,41], and circles [60,67]. When the space for labels is limited, some studies adopt internal labeling, *i.e.*, placing labels inside the regions of target elements, thus avoiding occlusion of other elements [13,18,33,34,43,50]. Additionally, some studies use a mixed strategy of internal and boundary labeling to explore more layout options [15,19,42]. These works provide valuable experience for label layout generation. With the significant developments of deep learning in various domains, several studies have incorporated learning techniques into label placement [11,23,38]. SmartOverlays (SO) [23] uses saliency maps to guide label placement in videos, reducing occlusion of visually important regions. Semantic-Aware Labeling (SAL) [26] incorporates both saliency and high-level semantic cues via a learned guidance map to optimize label placement in street-view scenes. However, learning-based methods primarily use a single model, which limits their ability to capture the differences between various label layouts. Moreover, layout samples are used solely as training data, restricting the design insights that can be extracted.

A recent study, LPGT [46], formulates label placement as a node prediction problem and iteratively learns label positions using a sequence of graph Transformer modules. We adopt this formalization, but by designing cluster-guided experts, we directly predict label positions, avoiding iterative estimation and achieving superior performance.

2.2 Mixture of Experts

The core idea behind MoE is to divide the model into multiple specialized sub-models, known as experts, where each expert specializes in a specific part of the problem space [25,28]. A gating function dynamically selects a subset of these experts for each input. Based on the design of gating functions, MoE can be broadly classified into two categories: dense MoE and sparse MoE. Dense MoE activates all experts during each iteration [39,63]. Although it improves performance, this approach also results in considerably increased computational overhead. Consequently, with the rapid development of large language models (LLMs), sparse MoE has gained more popularity. It activates only relevant experts for each input. This selective activation renders sparse MoE a powerful and flexible framework in diverse domains [16,27,49,66].

MoE for GNNs   Graph neural networks (GNNs) and graph Transformer models have excelled in a range of tasks involving graph structures [65]. For GNNs with MoE, early efforts combine GNNs from different domains to assemble knowledge, approximating a fixed-weight MoE [1,54,59]. Recent advancements have integrated MoE to enhance graph analysis by using GNN or fully connected layers as experts, realizing adaptive aggregation [58] and domain-specific experts [30]. GMoE [58] allows each node to select experts suited to diverse graph structures and receptive fields. In contrast, we integrate multiple FFNs in the MoCE layer, where each expert consists of a pair of FFNs dedicated to refining node and edge representations.

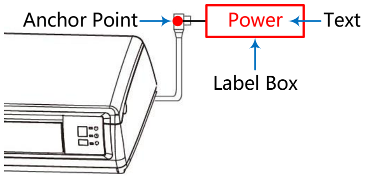


Fig. 2: Constituents of a label: text, label box, and anchor point.

2.3 Retrieval-Augmented Generation

Retrieval augmentation leverages either external memory [53] or the training data itself [24] to enhance generative models without expanding network parameters. In fields such as natural language processing and computer vision, this approach dynamically fetches relevant information using retrieval mechanisms, *e.g.*, k-nearest neighbors from pre-calculated embeddings, to inform the generation process [3,6,21]. Recent advances in retrieval-augmented generation for visualization have also shown promise. For instance, FinFlier [22] and Data Playwright [52] employ external memory-based retrieval to dynamically fetch relevant contextual information, which is then seamlessly integrated into the generation process to enhance the quality and relevance of the generated content. Given the availability of the SWU-AMIL dataset [46] for label placement, we opt to use the training data itself.

3 Preliminaries

In this section, we follow [46] to provide the problem formulation and review general GNN-based label placement.

3.1 Problem Formulation

A label consists of three essential constituents (Fig. 2):

- **text**: the short description of the target element,
- **label box**: a rectangular region enclosing the text, and
- **anchor point**: a representative position on the target element.

The label placement problem aims to position a given set of labels within a graphic. Its input consists of a graphic/image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$ and a label set $\mathbb{L} = \left\{\left(\mathbf{a}_i \in \mathbb{R}^2, \mathbf{b}_i \in \mathbb{R}^2\right)\right\}_{i=1}^{n_l}$, where each label is represented by its anchor point $\mathbf{a}_i = (x_i, y_i)$ and box size $\mathbf{b}_i = (h_i, w_i)$, and n_l is the number of labels. Each label is initially placed at its anchor. The objective is to compute the expected label displacements $\mathbb{D} = \left\{\hat{\mathbf{d}}_i \in \mathbb{R}^2\right\}_{i=1}^{n_l}$, where $\hat{\mathbf{d}}_i$ is the estimated displacement of the i-th label from its anchor to the final position.

Graph Construction   We construct a complete attributed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ to represent the labels and their interactions, where each label is associated with a node. $\mathbb{V} = \left\{\left(\mathbf{z}_i^g \in \mathbb{R}^4, \mathbf{z}_i^v \in \mathbb{R}^{d_v}\right)\right\}_{i=1}^{n_l}$ denotes the set of nodes with attributes. The node attribute consists of the geometric component $\mathbf{z}_i^g = [\mathbf{a}_i, \mathbf{b}_i]$ and the visual component $\mathbf{z}_i^v$. To construct the visual attribute, we first extract convolutional feature maps $\mathbf{F} \in \mathbb{R}^{h\times w\times d_v}$ from the entire image $\mathbf{I}$ using a convolutional neural network (CNN), where h and w denote the spatial size of convolutional features, and d_v denotes the dimension. We then perform bilinear interpolation and max pooling on the image patch features corresponding to the node. Analogously, $\mathbb{E} = \left\{\left(\mathbf{e}_{ij}^g \in \mathbb{R}^6, \mathbf{e}_{ij}^v \in \mathbb{R}^{d_v}\right)\right\}$ is the set of edges with attributes. The geometric attribute $\mathbf{e}_{ij}^g = \left[\mathbf{a}_i - \mathbf{a}_j, \mathbf{b}_i, \mathbf{b}_j\right]$ of each edge (i, j) includes the distance vector between its two associated nodes and the sizes of corresponding boxes. The visual attribute $\mathbf{e}_{ij}^v$ is extracted from the image region between the two nodes by sampling pixel points and applying average pooling to their visual features.

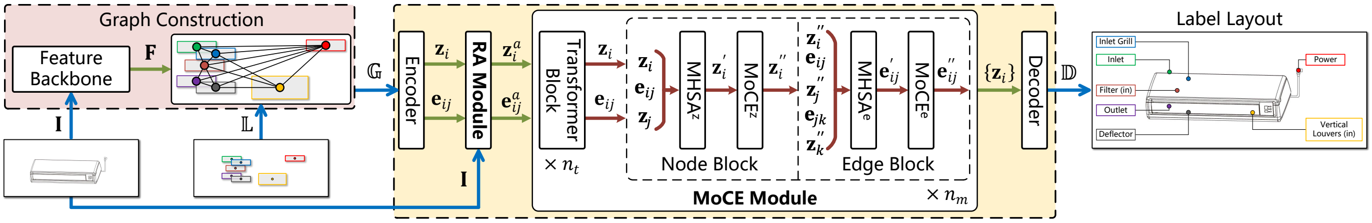


Fig. 3: Overview of the LPCE architecture. First, the encoder initializes each node representation $\mathbf{z}_i$ and edge representation $\mathbf{e}_{ij}$ of the input graph $\mathbb{G}$. The RA module produces the augmented node representation $\mathbf{z}_i^a$ and edge representation $\mathbf{e}_{ij}^a$ based on style features from reference samples. Next, the MoCE module employs multiple cluster-guided experts to refine the features of diverse label layouts. Finally, the decoder predicts the displacements $\mathbb{D}$ of the labels. The label layout is generated by combining the anchor positions and displacements of all labels. For clarity, some edges are omitted in the graph $\mathbb{G}$.

3.2 Graph Transformer Block

An off-the-shelf graph Transformer block is used to learn node and edge representations [46]. We adjust its hyperparameters for LPCE. The Transformer block contains a node block and an edge block. Both blocks consist of a multi-head self-attention (MHSA) layer and an FFN:

$$\begin{aligned}
\mathbf{z}_i' &= \mathrm{MHSA}^z\left(\mathbf{z}_i, \left\{\left(\mathbf{z}_j, \mathbf{e}_{ij}\right)\right\}_{j\in\mathbb{N}_i}\right) + \mathbf{z}_i \\
\mathbf{z}_i'' &= \mathrm{FFN}^z\left(\mathbf{z}_i'\right) + \mathbf{z}_i' \\
\mathbf{e}_{ij}' &= \mathrm{MHSA}^e\left(\mathbf{e}_{ij}, \mathbf{z}_i'', \mathbf{z}_j'', \left\{\left(\mathbf{e}_{jk}, \mathbf{z}_k''\right)\right\}_{(j,k)\in\mathbb{N}_{ij}}\right) + \mathbf{e}_{ij} \\
\mathbf{e}_{ij}'' &= \mathrm{FFN}^e\left(\mathbf{e}_{ij}'\right) + \mathbf{e}_{ij}'
\end{aligned} \tag{1}$$

where $\mathbf{z}_i$ and $\mathbf{e}_{ij}$ denote the hidden representations of node i and edge (i, j), respectively. $\mathbb{N}_i$ denotes the first-order neighbor node set of node i, and $\mathbb{N}_{ij}$ denotes the neighbor edge set of edge (i, j). The neighbors of an edge are defined as edges that share at least one associated node with it. The FFN consists of two linear layers, with a ReLU activation function applied between them. The difference between the two blocks lies in their attention mechanism. In the node MHSA layer, a node-level attention projects QKV vectors from each node representation, simultaneously conditioned on the edge representation between two nodes. In the edge MHSA layer, an edge-level attention projects QKV vectors from each edge representation and associated node representations, then adds the QKV vectors of edges and nodes. Therefore, mutual dependencies between node and edge representations are formed across the two blocks.

4 METHODOLOGY

The proposed LPCE, as illustrated in Fig. 3, contains four modules: an encoder, a MoCE module, an RA module, and a decoder.

- **Encoder.** The encoder integrates label information and image features. It takes the graph $\mathbb{G}$ as input and fuses the geometric and visual attributes of nodes and edges into initial representations. This is achieved via linear projections applied to each node and edge, defined as $\mathcal{E}^z : \left(\mathbf{z}_i^g, \mathbf{z}_i^v\right) \mapsto \mathbf{z}_i \in \mathbb{R}^d$ and $\mathcal{E}^e : \left(\mathbf{e}_{ij}^g, \mathbf{e}_{ij}^v\right) \mapsto \mathbf{e}_{ij} \in \mathbb{R}^d$, where d is the dimension of representations.
- **RA Module.** Reference label layouts are retrieved for each input sample based on image similarity. The features of these reference layouts are injected into the initial node and edge representations of each sample. We describe the details in Sec. 4.1.
- **MoCE Module.** This module consists of n_t standard Transformer blocks (Eq. (1)) and n_m ones equipped with the MoCE layer. The standard blocks update the node and edge representations. The MoCE layer is composed of multiple FFNs that further refine the representations. Each expert consists of a pair of FFNs. The gating function clusters input samples based on their graph representations, guiding each sample to its matching expert according to the clustering results. The details are described in Sec. 4.2.
- **Decoder.** The decoder takes the updated node representations as input, and predicts the displacement $\hat{\mathbf{d}}_i$ of each node by an FC layer, $\mathcal{D} : \mathbf{z}_i \mapsto \hat{\mathbf{d}}_i$. The label layout is determined by adding the displacements to the anchor positions of all labels.

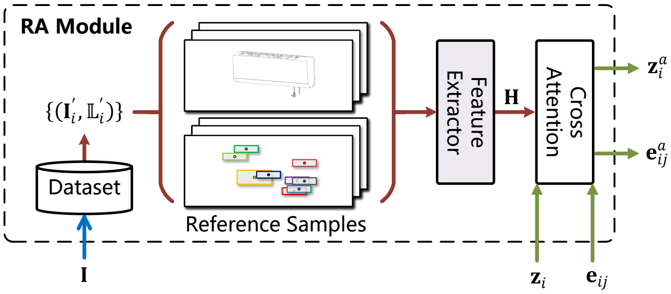


Fig. 4: Architecture of the RA module. References $\{(\mathbf{I}_i', \mathbb{L}_i')\}_{i=1}^{K_r}$ for each input image $\mathbf{I}$ are retrieved based on visual similarity and encoded into style features $\mathbf{H}$. The features are injected into each node representation $\mathbf{z}_i$ and edge representation $\mathbf{e}_{ij}$ of the input sample via cross-attention.

4.1 Retrieval Augmentation

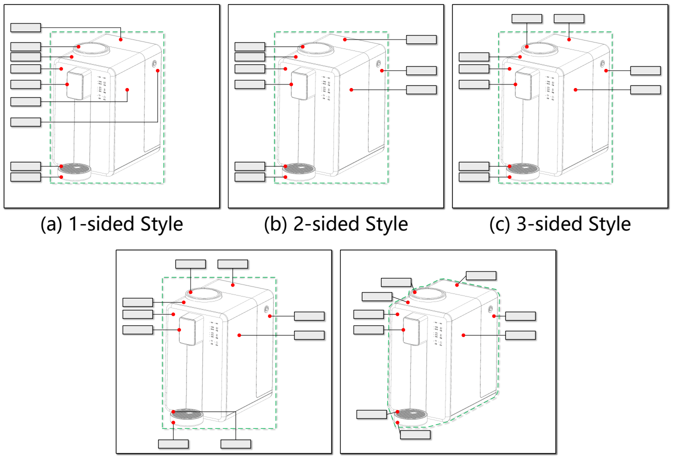
To produce visually appealing label layouts, labels are typically required to be placed in designated regions. Various layout styles are thus formed. Absorbing stylistic experience from existing layouts presents a significant opportunity for promoting the quality of label placement. Therefore, retrieval augmentation is introduced to enrich the initial node and edge representations via reference layout information, as shown in Fig. 4. First, reference layout samples are retrieved from a given dataset. Then, a reference feature extractor encodes these samples into style features. Finally, the reference features are injected into the node and edge representations, respectively, by a cross-attention mechanism.

4.1.1 Reference Retrieval

Given the input image $\mathbf{I}$ as the query, we retrieve a set of reference layout samples from a label placement dataset. These samples are represented by their images and raw label information $\left\{\left(\mathbf{I}_i', \mathbb{L}_i'\right)\right\}_{i=1}^{K_r}$, where K_r denotes the number of samples. We adopt DreamSim [17] to measure image similarity between the query and candidate samples, as it aligns well with human perception of semantic similarity. Since images with similar semantic content tend to exhibit similar label layouts [5], using a perceptual metric helps improve the relevance of retrieved layouts. Compared to traditional metrics that focus primarily on low-level features, DreamSim captures higher-level semantics, which better supports the retrieval augmentation. Top-K_r samples are thus retrieved based on image similarity between $\mathbf{I}$ and $\mathbf{I}'$. To prevent ground-truth leakage, all samples from the training set serve as the retrieval source for both training and testing, with the exception of the query itself during training.

4.1.2 Feature Encoding

We define five layout styles based on the rectangular and contour boundary labeling [5]. For rectangular labeling, depending on the number of sides involved, we distinguish 1-sided, 2-sided, 3-sided, and 4-sided layouts. A 1-sided layout is typically used when there are only a few



(a) 1-sided Style

(b) 2-sided Style

(c) 3-sided Style

(d) 4-sided Style

(e) Contour Style

Fig. 5: Five label layout styles of an illustration. (a)-(e): 1-sided, 2-sided, 3-sided, 4-sided, and contour styles.

labels. The specific side for label placement depends on the available space. In a 2-sided layout, the sides are either opposite or adjacent. When the number of labels increases or there is sufficient space, 3-sided and 4-sided layouts are favored. This allows labels to be placed closer to their target elements, thus reaching a balance between aesthetics and readability. Contour labeling creates a layout that roughly matches the shape of the target object, achieving label placement that blends in with the object. Five corresponding label layout styles are thus developed based on the above layouts, as shown in Fig. 5.

To embed reference layouts into style features, we design a reference feature extractor based on style classification. It takes the graph $\mathbb{G}'$ of a reference sample $(\mathbf{I}', \mathbb{L}')$ as input, and predicts its style. The extractor, as shown in Fig. 6, consists of four components: encoder, Transformer module, feature fusion module, and predictor. 1) The encoder follows the same architecture as the one in Sec. 4, and also generates the initial node and edge representations. 2) Then, the representations are updated by n_s standard Transformer blocks (Eq. (1)). 3) The updated node representations are aggregated into the graph-level representation $\mathbf{h}_{\mathbb{G}'} \in \mathbb{R}^d$ using Transformer-based adaptive pooling [24]. While visual features have been incorporated into both node and edge representations, they are limited to local patch features and do not capture global image features. To capture the image-level features, the extracted feature maps $\mathbf{F}'$ of the image $\mathbf{I}'$ are condensed into a feature vector $\mathbf{h}_{\mathbf{I}'} \in \mathbb{R}^d$ using adaptive average pooling and an FC layer. Both feature types $(\mathbf{h}_{\mathbb{G}'}, \mathbf{h}_{\mathbf{I}'})$ are taken into account for identifying the layout style. We introduce a fusion function, defined as $\mathcal{F} : [\mathbf{h}_{\mathbb{G}'}, \mathbf{h}_{\mathbf{I}'}] \mapsto \mathbf{h} \in \mathbb{R}^d$, to combine the features. It is implemented by an MLP. 4) Finally, by taking the fused features $\mathbf{h}$ as input, the predictor, defined as $\mathcal{P} : \mathbf{h} \mapsto \hat{\mathbf{y}} \in \mathbb{R}^5$, estimates the probability distribution $\hat{\mathbf{y}}$ of the input sample across the five styles. This is implemented by an FC layer.

In addition, we adopt a focal loss to guide the extractor training:

$$\mathcal{L} = -\alpha_c (1 - \hat{y}_c)^{\gamma} \log(\hat{y}_c) \tag{2}$$

where $c \in \{1, 2, 3, 4, 5\}$ denotes the index of the ground-truth category of the input sample, $\hat{y}_c$ is the c-th entry of $\hat{\mathbf{y}}$, and the focusing parameter γ is set to 2. The class balancing factor α_c is computed based on the normalized inverse frequency of each class:

$$\alpha_c = \frac{1 - \frac{n_c}{N}}{\sum_{c=1}^{5} \left(1 - \frac{n_c}{N}\right)} \tag{3}$$

where n_c denotes the number of samples in the c-th class, and N is the total number of training samples. We pre-train the feature extractor and remove its predictor. The fused features are treated as style features, thereby avoiding the issue of varying numbers of labels across different samples. Each sample $(\mathbf{I}', \mathbb{L}')$ is embedded into a feature vector $\mathbf{h}$, and these features are then stacked into a matrix $\mathbf{H} \in \mathbb{R}^{K_r \times d}$.

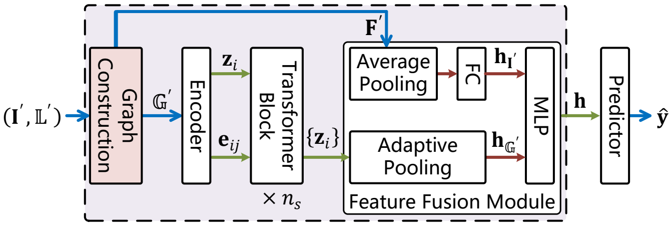


Fig. 6: Architecture of the reference feature extractor. First, given the graph $\mathbb{G}'$ of a reference sample $(\mathbf{I}', \mathbb{L}')$ as input, the encoder generates the initial node and edge representations. Next, the representations are updated by the Transformer module. Then, the feature fusion module integrates the graph-level representation $\mathbf{h}_{\mathbb{G}'}$ and the image-level feature $\mathbf{h}_{\mathbf{I}'}$ into the style feature $\mathbf{h}$.

4.1.3 Feature Injection

Both node and edge representations are augmented by injecting the reference feature $\mathbf{H}$. Taking the node representation $\mathbf{z}_i$ as an example, the cross-attended feature between $\mathbf{z}_i$ and $\mathbf{H}$ is computed to generate the augmented representation $\mathbf{z}_i^a \in \mathbb{R}^d$:

$$\mathbf{z}_i^a = \mathbf{H}^{\top} \cdot \mathrm{softmax}\left(\frac{\mathbf{H} \cdot \mathbf{z}_i}{\sqrt{d}}\right) + \mathbf{z}_i \tag{4}$$

where a residual connection is applied after the attention block. In the cross-attention mechanism, the node representation acts as the query vector, and the reference feature serves as both the key and value vectors. Analogously, the augmented representation $\mathbf{e}_{ij}^a$ is obtained for each edge representation $\mathbf{e}_{ij}$. This design allows the placement of given labels to absorb style features from the reference layouts.

4.2 Mixture of Cluster-guided Experts

Multiple experts are employed to learn fine-grained characteristics for various label layouts, each of which is responsible for a group of layouts. The gating function routes input samples to their matching experts by clustering their graph representations.

4.2.1 MoCE Layer

We design a MoCE layer to replace the FFNs in both the node and edge blocks. Specifically, it consists of two sub-layers: a node MoCE sub-layer and an edge MoCE sub-layer, each corresponding to the FFN originally used in the respective block. The node sub-layer integrates $K_e + 2$ FFNs, as shown in Fig. 7. K_e FFNs $\left\{\mathrm{FFN}_k^z\right\}_{k=1}^{K_e}$ serve as experts, each dedicated to learning features for a group of samples, thus generating more distinctive node representations. This enables finer-grained characterization of each sample. The remaining two FFNs placed before (FFN_a^z) and after (FFN_b^z) the expert FFNs provide a smoothing buffer around them, damping gradient spikes during sparse activation. This structure prevents abrupt output shifts, stabilizing MoCE training [68]. The edge sub-layer also consists of $K_e + 2$ FFNs, $\left\{\mathrm{FFN}_a^e, \left\{\mathrm{FFN}_k^e\right\}_{k=1}^{K_e}, \mathrm{FFN}_b^e\right\}$, with roles similar to those in the node sub-layer. Expert activation is shared across the two sub-layers: each expert is composed of a pair of FFNs, $\left(\mathrm{FFN}_k^z, \mathrm{FFN}_k^e\right)$, and the activation is determined from the node side.

4.2.2 Cluster-based Gating

We design a cluster-based gating function to determine which expert the input sample $(\mathbf{I}, \mathbb{L})$ should be assigned to. Specifically, we first extract the graph representation of the sample for clustering purposes. The input node representations to the experts, obtained from FFN_a^z (the FFN before the expert FFNs), are aggregated using global additive pooling and then passed through a nonlinear transformation to produce the graph representation $\mathbf{h}_{\mathbb{G}} \in \mathbb{R}^{d_h}$:

$$\mathbf{h}_{\mathbb{G}} = \mathrm{MLP}\left(\mathrm{GAP}\left(\left\{\mathrm{FFN}_a^z\left(\mathbf{z}_i'\right)\right\}_{i=1}^{n_l}\right)\right) \tag{5}$$

where $\mathbf{z}_i'$ denotes the output representation of the node MHSA layer, and GAP denotes the global additive pooling operation. The nonlinear transformation is implemented by a multi-layer perceptron (MLP) with

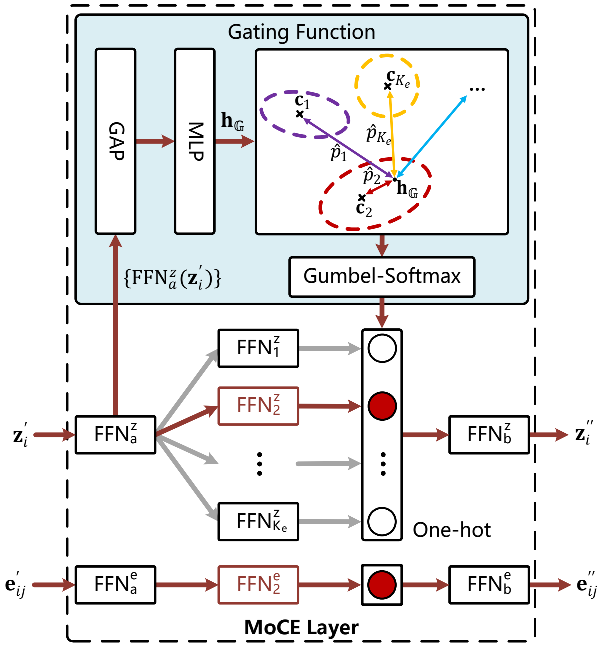


Fig. 7: Architecture of the MoCE layer. It comprises a node and an edge sub-layer. Each expert consists of two FFNs $\left(\mathrm{FFN}_k^z, \mathrm{FFN}_k^e\right)$—one from each sub-layer—responsible for refining node and edge representations. Two FFNs, placed before (FFN_a) and after (FFN_b) the expert FFNs, are used to ensure training stability. Expert activation is determined from the node sub-layer via the cluster-based gating function, which assigns each input to a specific expert by clustering its graph representation $\mathbf{h}_{\mathbb{G}}$ and shares the assignment across both sub-layers. In this example, the second expert $\left(\mathrm{FFN}_2^z, \mathrm{FFN}_2^e\right)$ is activated. For clarity, inactive experts in the edge sub-layer and their one-hot entries are omitted.

two hidden layers. Before generating the clustering feature (*i.e.*, the graph representation), the learning of node and edge representations undergoes the following process. The geometric and visual attributes of nodes and edges of the input sample are separately fused into initial node and edge representations in the encoder. Subsequently, in the RA Module, the style features of retrieved reference layouts are injected into the initial node and edge representations. Finally, in the MoCE Module, standard Transformer blocks update the node and edge representations. Therefore, the resulting clustering feature effectively incorporates geometric features, visual features, and retrieved layout features.

Next, we compute the assignment probability $\hat{\mathbf{p}}$ of the input sample to the experts. To achieve this, we introduce K_e learnable parameters $\left\{\mathbf{c}_k \in \mathbb{R}^{d_h}\right\}_{k=1}^{K_e}$ as centroids for the K_e clusters, respectively. The centroids are initialized by K-means clustering. The similarity between the sample's representation and each centroid is computed to obtain the assignment probability. A hard Gumbel-Softmax operation is then applied, converting the probability into a hard decision, assigning the sample to the cluster with the highest probability, thus activating the corresponding expert. This ensures precise expert selection while maintaining differentiability. Due to the heavy-tailed nature of the Cauchy distribution [40], it is effective at handling noise and outliers, which helps maintain training stability. Therefore, we use the Cauchy distribution as a kernel to measure the similarity between the graph representation $\mathbf{h}_{\mathbb{G}}$ and the k-th centroid $\mathbf{c}_k$:

$$\hat{p}_k = \frac{\left(1 + \|\mathbf{h}_{\mathbb{G}} - \mathbf{c}_k\|^2\right)^{-1}}{\sum_{k'=1}^{K_e}\left(1 + \|\mathbf{h}_{\mathbb{G}} - \mathbf{c}_{k'}\|^2\right)^{-1}} \tag{6}$$

where $\hat{p}_k$ is the k-th element of $\hat{\mathbf{p}}$, indicating the probability that the input sample belongs to the k-th cluster. Regarding the training stability of the gating function, we further adopt a small batch size to allow more frequent updates of the cluster centroids during the training of LPCE. This helps mitigate extreme cases in the clustering process and reduces the model's sensitivity to the random initialization of centroids.

Finally, the node representations output from the MHSA layer are refined by the MoCE layer:

$$\mathbf{z}_i'' = \mathrm{FFN}_b^z\left(\mathrm{FFN}_{k^*}^z\left(\mathrm{FFN}_a^z\left(\mathbf{z}_i'\right)\right)\right) + \mathbf{z}_i' \tag{7}$$

where k^* denotes the index of the activated expert for the input sample. The edge representations are refined in a similar way, $\mathbf{e}_{ij}'' = \mathrm{FFN}_b^e\left(\mathrm{FFN}_{k^*}^e\left(\mathrm{FFN}_a^e\left(\mathbf{e}_{ij}'\right)\right)\right) + \mathbf{e}_{ij}'$.

The MoCE layer achieves sparsity through the cluster-based gating function, similar to the commonly used sparse gating mechanism [51]. This mechanism calculates a weighted sum of outputs from the top-k experts, with the activation of experts determined by the softmax operation. The proposed cluster-based gating is equivalent to activating the top-1 expert. Additionally, the MoCE layer avoids a load balancing issue common in sparsely-gated MoE models. The issue refers to the uneven distribution of workload across experts, with some being frequently utilized and others seldom or never engaged. To address this, a balancing loss is typically introduced to promote an even distribution of tokens across experts within each batch [9]. In contrast, the proposed gating function directs each sample to its matching expert by clustering graph representations, ensuring all experts are effectively trained.

5 Experiments

We evaluate the performance of our LPCE by comparing it with various methods for label placement. Additionally, we conduct ablation studies and a user study for further evaluation and analysis. The findings from these experiments may open potential pathways for future research and optimization of our method. The implementation details of LPCE are provided in the supplementary material.

5.1 Evaluation Benchmarks & Metrics

The experiments are conducted on a label placement dataset, SWU-AMIL [46]. This dataset contains 869 label layouts across 11 categories. All layouts are collected from commercial illustrations. These layouts are split into 695/174 samples for training and testing, respectively. Based on the five layout styles defined in Sec. 4.1, we add a style annotation to each sample. The numbers of the five classes of samples in the training and testing splits are {88, 87, 386, 107, 27} and {18, 22, 102, 19, 13}, respectively. Additionally, by applying slight local perturbations to the anchor coordinates, we augment the number of training samples to 6,950.

We use the following four metrics to evaluate label placement quality: 1) Probability of Correct Keypoints (PCK@τ) [46, 64], where $\tau = 0.05$ is a threshold tolerance factor; 2) Intersection over Union (IoU) between predicted and ground-truth label positions, calculated based on their respective boxes [48]; 3) Overlap between labels, computed as the average IoU over all pairs of label boxes [36]; 4) Label visibility (LV), measured by the proportion of labels not occluded by any others [29]. Higher PCK, IoU, and LV values indicate better performance, while lower Overlap also reflects better results. Calculations of these metrics are explained in the supplementary material. For all models, the mean values and standard deviations of these metrics over 5 runs are reported. To evaluate model efficiency, we profile the number of model parameters and the time required to generate a label layout. We report the number of model parameters without the feature backbone, as the backbone remains consistent across all models. We calculate the mean generation time across all test samples.

5.2 Evaluation Results

LPCE is compared with four groups of methods: (1) GNNs: GCN2 [10], GATv2 [8], GPS [47], and ASC [20]; (2) MoE for GNNs: GMoE [58]; (3) Learning-based label placement methods: LPGT [46]; (4) Rule-driven label placement methods: SO [23], and SAL [26]. The experimental settings are listed below:

- For fair comparison with GNN models, we maintain the overall architecture of LPCE and replace the Transformer block with the GNN layer from each model. Each expert is a GNN layer. To avoid the over-smoothing problem, we apply residual connections

Table 1: Comparison of label layout quality (PCK@0.05 (%)) and efficiency on the SWU-AMIL dataset. “Time” denotes the average generation time of a label layout. “Params” refers to the number of parameters in a model, excluding the feature backbone. PCK results are averaged over 5 restarts (LPGT results are taken from [46]). Time statistics are calculated 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)	Params
GCN2$_{rm}$ [10]	43.21	63.74	38.00	68.83	67.13	45.00	63.77	48.27	24.29	**89.72**	21.08	57.64 ± 0.37	**9.64**	**1.9M**
GATv2$_{rm}$ [8]	42.26	62.90	40.67	70.84	68.60	52.50	65.62	48.99	26.00	88.61	18.80	58.33 ± 0.65	12.63	4.1M
GPS$_{rm}$ [47]	44.43	76.37	40.67	71.11	67.02	60.00	64.30	48.40	26.00	88.06	19.26	58.88 ± 0.59	11.24	5.9M
ASC$_{rm}$ [20]	42.82	68.22	40.67	70.14	66.73	62.50	61.99	48.27	26.57	89.39	22.95	58.04 ± 0.34	10.61	2.4M
GMoE$_r$ [58]	42.18	57.93	38.00	70.17	65.75	55.00	64.20	47.82	**32.57**	89.61	23.85	57.98 ± 0.70	15.35	3.7M
LPGT [46]	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	11.4M
LPGT* [46]	45.11	77.00	**48.00**	**74.45**	**73.55**	77.50	59.43	48.04	31.72	84.83	30.12	60.26 ± 0.32	22.25	11.4M
LPCE	**51.68**	**83.44**	44.67	74.21	73.31	70.00	**65.66**	49.99	21.43	88.89	25.98	**62.65 ± 0.31**	14.49	6.9M

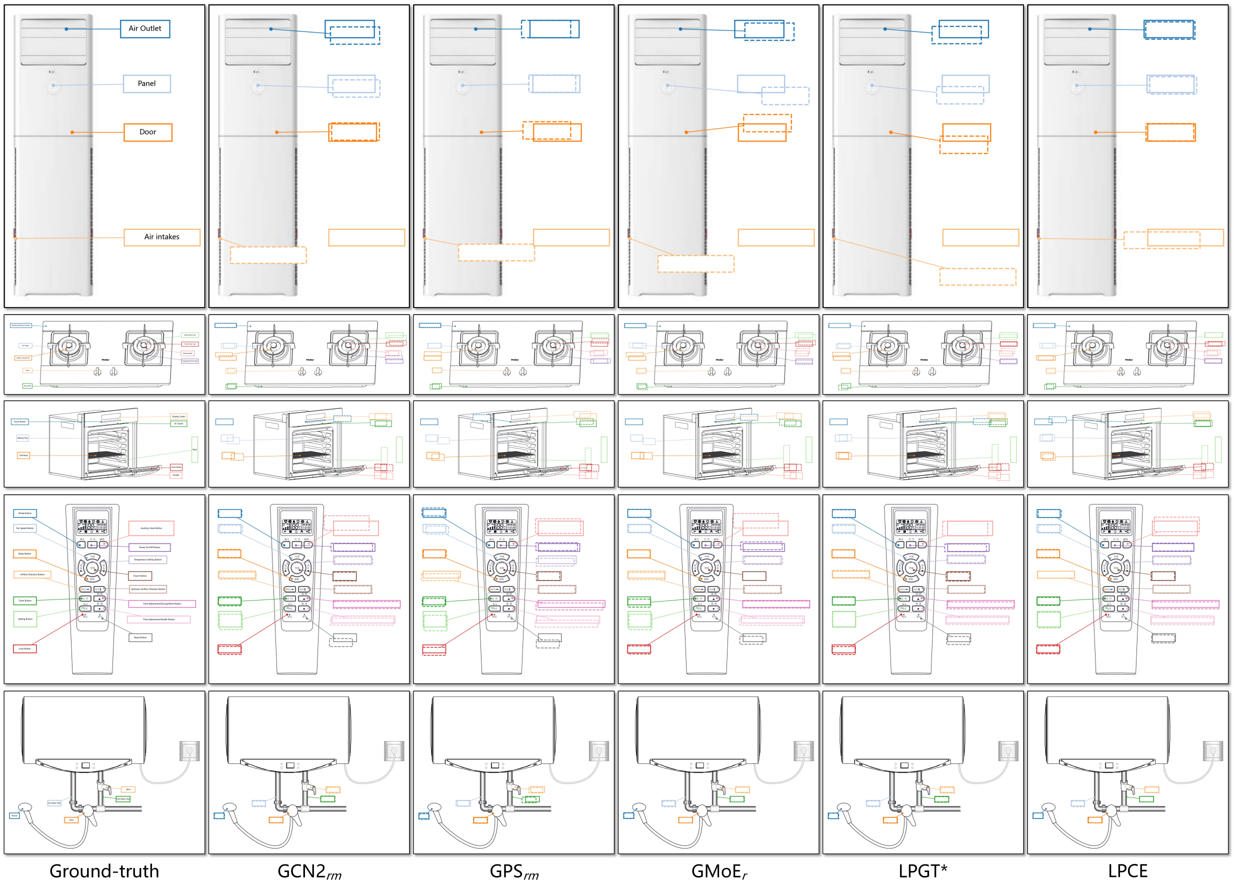


Fig. 8: Visualization results on the SWU-AMIL dataset. Solid boxes represent ground-truth label positions, while dashed boxes indicate predicted ones. Different colors distinguish labels, and label texts are omitted in predictions for clarity.

Table 2: Comparison of label layout quality (IoU, LV, Overlap (%)) on the SWU-AMIL dataset.

Model	IoU ↑	LV ↑	Overlap ↓
GCN2$_{rm}$ [10]	29.96 ± 0.34	85.94 ± 0.30	0.40 ± 0.02
GATv2$_{rm}$ [8]	30.06 ± 0.27	86.38 ± 0.98	0.40 ± 0.04
GPS$_{rm}$ [47]	28.85 ± 0.41	85.56 ± 0.80	0.46 ± 0.06
ASC$_{rm}$ [20]	30.27 ± 0.51	85.87 ± 0.46	0.41 ± 0.04
GMoE$_r$ [58]	30.16 ± 0.50	86.18 ± 0.51	0.41 ± 0.04
LPGT* [46]	30.15 ± 0.87	87.74 ± 0.54	0.40 ± 0.11
LPCE	**31.95 ± 0.43**	**89.39 ± 0.52**	**0.30 ± 0.03**

after each GNN layer. For models implemented based on attention mechanisms (GATv2) or Transformer architectures (GPS), we set the number of heads in their multi-head attention to 16, which is consistent with LPCE. Additionally, GCN layers [32] are utilized as the message passing layers in both GPS and ASC. These GNN models are denoted by their names with the subscript *rm*.

- Following the standard configuration of GMoE, we define each expert as a GCN layer. Additionally, we incorporate a retrieval augmentation module, in which the reference feature extractor is implemented based on GCN layers. This new model is denoted as GMoE$_r$.
- The publicly reported results of LPGT were obtained on the training set of SWU-AMIL. We retrain it on the augmented training data, and denote the model as LPGT*.

Quantitative Results From the quantitative results illustrated in Tabs. 1 and 2, LPCE consistently outperforms the compared models, achieving the highest PCK (62.65%), IoU (31.95%), and LV (89.39%), as well as the lowest Overlap (0.30%). These results demonstrate LPCE’s superiority in balancing spatial precision and visual clarity

Table 3: Ablation studies of the RA and MoCE modules of LPCE on the SWU-AMIL dataset.

Model	ac	d.wash	d.cabi.	fridge	g.stove	oven	panel	remote	washer	w.heat	w.puri	Avg.	Time(ms)	Params
Baseline	45.51	82.44	44.67	73.62	73.36	47.50	61.01	48.29	17.14	87.89	**30.89**	60.35 ± 0.81	**12.98**	**5.5M**
Baseline$_r$	49.03	81.49	44.00	74.19	70.54	**70.00**	61.88	49.81	15.72	87.33	25.68	61.16 ± 0.61	13.81	6.2M
Baseline$_m$	48.97	**83.81**	**45.33**	74.14	**75.00**	55.00	62.56	48.24	20.00	87.11	30.72	61.57 ± 0.35	13.77	6.2M
Baseline$_{rm}$	**51.68**	83.44	44.67	**74.21**	73.31	**70.00**	**65.66**	**49.99**	**21.43**	**88.89**	25.98	**62.65 ± 0.31**	14.49	6.9M

Table 4: Ablation studies of the RA and MoCE modules of the comparison GNN models on the SWU-AMIL dataset.

Model	ac	d.wash	d.cabi.	fridge	g.stove	oven	panel	remote	washer	w.heat	w.puri	Avg.	Time(ms)	Params
GCN2	40.76	58.49	34.00	66.97	63.33	30.00	57.70	44.45	29.71	81.39	20.14	54.25 ± 0.48	6.38	1.1M
GCN2$_r$	38.43	59.65	34.00	66.47	62.30	25.00	60.82	44.36	25.43	84.00	21.93	54.27 ± 0.43	9.08	1.5M
GCN2$_m$	41.69	58.96	33.33	67.60	61.62	42.50	59.52	46.77	19.14	86.50	20.69	55.44 ± 0.44	9.05	1.6M
GCN2$_{rm}$	43.21	63.74	38.00	68.83	67.13	45.00	63.77	48.27	24.29	89.72	21.08	57.64 ± 0.37	9.64	1.9M
GATv2	39.74	57.65	36.67	66.95	60.30	32.50	60.39	45.05	21.14	85.11	23.77	54.79 ± 0.56	11.03	2.5M
GATv2$_r$	42.25	54.38	34.00	66.04	61.54	47.50	63.49	46.26	33.71	84.94	24.36	55.84 ± 0.17	11.75	3.2M
GATv2$_m$	41.19	62.95	36.00	70.11	71.21	32.50	63.02	46.54	34.57	88.36	19.80	57.30 ± 0.28	11.76	3.5M
GATv2$_{rm}$	42.26	62.90	40.67	70.84	68.60	52.50	65.62	48.99	26.00	88.61	18.80	58.33 ± 0.65	12.63	4.1M
GPS	31.91	65.73	40.00	50.22	60.46	37.50	47.93	33.53	29.14	7.53	15.14	39.34 ± 3.49	9.12	3.1M
GPS$_r$	38.17	55.21	37.33	66.04	63.22	50.00	62.44	46.85	20.29	78.22	21.30	54.23 ± 1.14	9.66	3.5M
GPS$_m$	40.17	74.49	44.67	67.50	67.98	65.00	62.97	47.16	20.57	83.81	17.10	56.13 ± 0.81	10.26	5.6M
GPS$_{rm}$	44.43	76.37	40.67	71.11	67.02	60.00	64.30	48.40	26.00	88.06	19.26	58.88 ± 0.59	11.24	5.9M
ASC	40.48	51.40	36.67	66.81	61.05	47.50	60.27	45.09	24.57	77.78	22.97	54.34 ± 0.87	8.76	1.4M
ASC$_r$	41.12	55.84	33.33	68.95	61.86	60.00	59.95	46.55	30.00	83.61	24.69	55.99 ± 0.41	10.05	1.7M
ASC$_m$	42.88	62.87	36.67	69.99	66.22	60.00	59.01	46.83	21.71	88.83	22.45	57.06 ± 0.46	10.00	2.1M
ASC$_{rm}$	42.82	68.22	40.67	70.14	66.73	62.50	61.99	48.27	26.57	89.39	22.95	58.04 ± 0.34	10.61	2.4M
GMoE	42.93	61.06	35.33	67.32	66.85	47.50	61.74	45.57	32.29	91.56	22.19	56.71 ± 0.31	14.34	3.0M
GMoE$_r$	42.18	57.93	38.00	70.17	65.75	55.00	64.20	47.82	32.57	89.61	23.85	57.98 ± 0.70	15.35	3.7M

Table 5: Ablation studies of the architecture of the expert on the SWU-AMIL dataset.

Model	ac	d.wash	d.cabi.	fridge	g.stove	oven	panel	remote	washer	w.heat	w.puri	Avg.	Time(ms)	Params
Trans.	47.53	**86.34**	41.33	72.63	70.53	70.00	62.67	49.61	20.00	88.50	**26.86**	60.75 ± 0.31	16.62	10.1M
FC	50.30	83.11	**46.00**	73.00	68.60	**72.50**	62.35	49.24	**29.14**	85.50	26.12	61.12 ± 0.18	**14.28**	**6.4M**
FFN	**51.68**	83.44	44.67	**74.21**	**73.31**	70.00	**65.66**	**49.99**	21.43	**88.89**	25.98	**62.65 ± 0.31**	14.49	6.9M

across multiple layout quality metrics. In addition, it achieves top-2 PCK values in multiple categories, including air conditioner (51.68%), dishwasher (83.44%), fridge (74.21%), panel (65.66%), and remote (49.99%), which collectively account for 80.55% of the training samples and 79.89% of the test samples, demonstrating better adaptability and robustness. Moreover, LPCE maintains a competitive inference speed (14.49 ms) compared to other advanced methods and has a relatively smaller number of parameters (6.9M). The augmented training data has improved PCK for LPGT by 0.99% (from 59.27% to 60.26%), enabling LPGT* to achieve the closest performance to LPCE among all methods compared. However, LPGT* has considerably more parameters (11.4M) and slower inference speed (22.25 ms), making it less practical than LPCE. Although GMoE$_r$ achieves relatively high performance in certain categories (*e.g.*, 32.57% PCK on washer), its average result still lags behind LPCE by 4.67%. Furthermore, while GNN models (GCN2$_{rm}$, GATv2$_{rm}$, GPS$_{rm}$, and ASC$_{rm}$) exhibit relatively lower inference latency and smaller model sizes, they show weaker performance. The performance of SO and SAL is reported as unsatisfactory in [46], with PCK values of 7.14% and 3.52%, respectively. Overall, LPCE not only achieves superior label placement accuracy but also ensures an effective trade-off between model complexity and computational efficiency, making it advantageous for real-world label placement applications.

Qualitative Results The predicted label layouts of several samples are visualized in Fig. 8. Compared to GCN2$_{rm}$, GPS$_{rm}$, GMoE$_r$, and LPGT*, LPCE generates more visually appealing layouts across different categories and numbers of labels. Some label positions predicted by the comparison methods are inappropriate, resulting in occlusion of target elements. In particular, all models generate decent label layouts for the water heater sample (last row). However, for the oven sample (third row), all models' predicted layouts exhibit label overlap on a vertically oriented label. We believe the likely reason is that vertically oriented labels were rarely encountered during training, leading to poor placement performance across all models for this type of label. To address this issue, we plan to incorporate prior knowledge of label orientation (horizontal or vertical) as conditional input in future studies.

5.3 Ablation Study

5.3.1 Key Components

We conduct ablation studies to shed light on the impact of RA and MoCE. We implement a baseline model by removing the RA module and replacing the MoCE-Transformer block with the standard one. The experiments are conducted by incorporating each of these two key designs into the baseline model individually, as shown in Tab. 3. The baseline model achieves an average placement accuracy of 60.35%. Integrating only the RA module (Baseline$_r$) improves the performance to 61.16%, indicating that it contributes positively to the overall performance. Similarly, introducing solely the MoCE module (Baseline$_m$) also enhances the baseline performance, reaching 61.57%. Notably, the combination of RA and MoCE (Baseline$_{rm}$, *i.e.*, LPCE) achieves the highest accuracy of 62.65%, clearly surpassing that of each individual module. This result underscores the complementary nature of RA and MoCE, as their joint use leads to superior performance compared to using each module independently.

Furthermore, Tab. 4 highlights the flexibility and effectiveness of RA and MoCE when applied to other GNN models. For each model, the baseline is denoted by its name. When separately added to the baseline, the two modules produce variants denoted by the subscripts r and m, respectively. Applying both modules to GCN2 improves the average accuracy from 54.25% to 57.64%, while in GATv2, the performance

Table 6: Ablation studies of the arrangement of the MoCE module on the SWU-AMIL dataset.

Model	ac	d.wash	d.cabi.	fridge	g.stove	oven	panel	remote	washer	w.heat	w.puri	Avg.	Time(ms)	Params
T-T-T	49.03	81.49	44.00	74.19	70.54	70.00	61.88	49.81	15.72	87.33	25.68	61.16 ± 0.61	13.81	6.2M
M-T-T	46.79	83.18	46.67	73.73	70.89	62.50	61.48	47.35	22.86	91.28	22.39	60.40 ± 0.67	14.69	6.9M
T-M-T	45.29	83.31	46.67	72.02	71.98	57.50	60.17	49.92	24.29	92.22	24.65	59.91 ± 0.52	14.59	6.9M
T-T-M	51.68	83.44	44.67	74.21	73.31	70.00	65.66	49.99	21.43	88.89	25.98	62.65 ± 0.31	14.49	6.9M
M-M-T	46.76	83.11	44.67	72.94	72.14	72.50	62.21	48.67	21.43	89.33	21.56	60.29 ± 0.59	15.07	7.6M
M-T-M	49.17	83.63	42.67	74.21	70.49	62.50	61.93	50.24	22.86	88.50	21.75	61.16 ± 0.33	15.25	7.6M
T-M-M	49.55	84.09	46.67	72.67	71.57	67.50	61.90	51.34	24.29	89.17	23.90	61.21 ± 0.33	15.27	7.6M
M-M-M	45.71	82.20	44.00	72.85	66.38	67.50	62.19	51.21	23.43	88.83	22.21	60.08 ± 0.64	16.25	8.3M

Table 7: Ablation studies of the gating function on the SWU-AMIL dataset.

Model	ac	d.wash	d.cabi.	fridge	g.stove	oven	panel	remote	washer	w.heat	w.puri	Avg.	Time(ms)	Params
Softmax (w/o balance)	47.38	82.40	42.00	73.45	71.90	60.00	62.72	**50.87**	23.43	88.33	25.18	60.96 ± 0.14	**13.91**	**6.7M**
Softmax (with balance)	47.58	83.11	**45.33**	73.37	72.40	**70.00**	63.63	50.85	**24.86**	**89.22**	24.13	61.25 ± 0.42	14.04	**6.7M**
Cluster	**51.68**	**83.44**	44.67	**74.21**	**73.31**	**70.00**	**65.66**	49.99	21.43	88.89	**25.98**	$\mathbf{62.65 \pm 0.31}$	14.49	6.9M

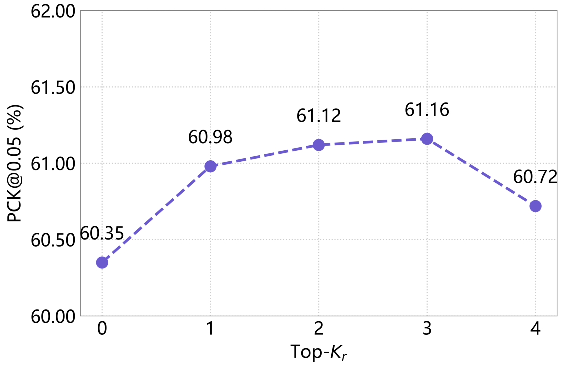


Fig. 9: Ablation studies of the number of reference samples.

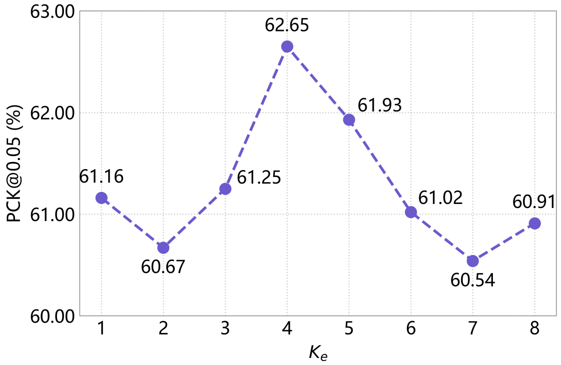


Fig. 10: Ablation studies of the number of experts.

increases from 54.79% to 58.33%. Similar trends can be observed in GPS (39.34% to 58.88%) and ASC (54.34% to 58.04%). Applying each module individually also results in performance gains across all baseline models. Even in the case of GMoE, a strong baseline, the RA module leads to a gain from 56.71% to 57.98%. The performance gains introduced by MoCE underscore the concept's effectiveness in [58], namely using a GNN layer as an expert. These consistent improvements across different architectures demonstrate that RA and MoCE are not only effective in the proposed method but also serve as general-purpose enhancements that can be readily integrated into a wide range of GNN frameworks to boost label placement performance.

5.3.2 Number of References

To evaluate how the number of reference samples K_r affects label layout quality, we train the baseline model of LPCE using the top-K_r reference samples, where $K_r \in \{1,2,3,4\}$. As shown in Fig. 9, the performance increases steadily from 60.98% at $K_r = 1$ to a peak of 61.16% at $K_r = 3$, indicating that incorporating more reference samples can enhance overall layout quality. Moreover, RA enhances the performance even with a single reference sample compared to the baseline (60.98% vs. 60.35%). However, further increasing the number to 4 leads to a performance drop to 60.72%, likely due to the inclusion of less relevant or noisy references. Based on this trend, we select $K_r = 3$ as the default setting, which offers the best balance between reference diversity and relevance.

5.3.3 MoCE Architecture

Number of Experts We conduct an ablation study to explore the effect of the number of experts K_e on label placement quality, where $K_e \in \{1,2,\dots,8\}$. As illustrated in Fig. 10, using only one expert yields a PCK of 61.16%, which essentially corresponds to the model without MoCE, *i.e.*, Baseline$_r$. As the number of experts increases, performance initially improves, peaking at 62.65% when $K_e = 4$. However, further increasing K_e beyond 4 leads to performance degradation, likely due to over-fragmentation of learning capacity. These results suggest that $K_e = 4$ is the optimal choice, which allows the model to capture fine-grained characteristics of diverse label layouts more effectively.

Expert Architecture To evaluate the impact of expert architecture, we compare three designs in which each expert is composed of a pair of Transformer blocks, FC layers, or FFN layers. In all cases, the gating function operates on the input node representations to the experts for clustering and expert assignment. As shown in Tab. 5, using FFNs as experts achieves the best performance, which corresponds to the design in LPCE. In contrast, the FC-based expert design (which uses the decoder's FC layers as experts) yields a lower accuracy of 61.12%, though it is more lightweight in terms of parameters (6.4M) and inference time (14.28 ms). The Transformer-based expert design, while having the largest model size (10.1M) and slowest inference (16.62 ms), performs the worst with a PCK of 60.75%, indicating that additional adjustments are required for the attention mechanism to optimize performance. These results demonstrate that the FFN-based experts offer a strong balance between learning capacity and computational efficiency.

MoCE Module Arrangement We conduct an ablation study to investigate how the number and position of MoCE-Transformer blocks affect model performance, as summarized Tab. 6. LPCE contains three Transformer blocks, each of which can be either standard (denoted as T) or equipped with the MoCE layer (denoted as M). Accordingly, each configuration can be represented as a three-letter code indicating the type of block used at each of the three positions. The baseline setup T-T-T, which uses three standard Transformer blocks ($n_t = 3$, $n_m = 0$), corresponds to the model without MoCE, *i.e.*, Baseline$_r$ in Tab. 3. T-T-M, the setting used in LPCE, achieves the best performance when incorporating one MoCE-Transformer block ($n_t = 2$, $n_m = 1$), outper-

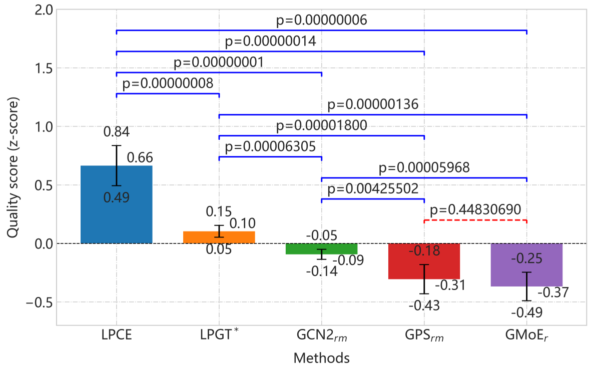


Fig. 11: User study results: quality scores with 95% confidence intervals. Statistically significant differences between method pairs are marked by solid blue brackets with associated p-values; red dashed brackets indicate non-significant differences.

forming the other two alternatives (M-T-T 60.40% and T-M-T 59.91%, respectively). When MoCE is used in two blocks ($n_t = 1$, $n_m = 2$), T-M-M achieves the highest PCK of 61.21%, slightly outperforming the baseline, while the other two configurations M-M-T and M-T-M perform similarly or slightly worse. Finally, using MoCE in all three blocks (M-M-M, *i.e.*, $n_t = 0$, $n_m = 3$) results in a performance drop to 60.08%. In summary, these results demonstrate that placing MoCE-Transformer blocks at deeper levels enables the model to capture high-level and fine-grained features more effectively. However, overusing or placing them at shallower levels may lead to diminishing returns or even performance degradation.

MoCE Stability To validate the role of the two auxiliary FFNs [68] placed before and after the expert FFNs in MoCE, we train a variant with them removed, while keeping the rest of the LPCE architecture unchanged. This removal leads to a significant performance drop, from 62.65 ± 0.31 to 59.79 ± 1.03. Moreover, the increased standard deviation indicates higher training instability. These observations confirm the effectiveness of the two additional FFNs in stabilizing training.

Gating Function We compare two softmax-based gating variants, with and without a balancing loss, against our cluster-based gating, as shown in Tab. 7. The softmax-based gating function replaces the MLP in Eq. (5) with an FC layer that linearly predicts a 4-dimensional vector. A hard Gumbel-Softmax operation is then applied to convert this vector into a one-hot representation, thereby assigning the input sample to the activated expert. While this design is straightforward, it achieves a lower PCK of 60.96%, and suffers from imbalanced expert usage during training. To address this load imbalance issue, we further introduce the balancing loss. For more details, please refer to [9]. This variant encourages more uniform expert activation. As a result, it improves the performance to 61.25%, but still lags behind the cluster-based gating.

5.4 User Study

We conducted a user study to compare user preferences across different methods (GCN2$_{rm}$, GPS$_{rm}$, GMoE$_r$, LPGT*, and LPCE) using 11 diverse samples from the SWU-AMIL test set, with an average of 9.0 labels per sample (ranging from 4 to 15). Participants were presented sequentially with pairs of label layouts for the same sample and asked to choose their preferred one, using a psychophysical paired comparison technique [56] and the two-alternative forced choice (2AFC) paradigm. In cases where participants perceived no clear difference between the two layouts, the pair was treated as a tie, and a preference was assigned at random with equal probability. Each sample featured 10 layout pairs, and to mitigate learning effects and fatigue, the sequence and side placement of layouts were randomized. A total of 29 participants (16 males and 13 females, average age 21.9, ranging from 18 to 34) were recruited through a university forum. Each participant evaluated 110 pairs, resulting in 108 scores per layout, with an average evaluation duration of 9 minutes and 58 seconds. Preferences were quantified into a count matrix $\mathbf{C}$ and analyzed for statistical significance using Thurstone's Law of Comparative Judgment [44, 56]. A two-tailed test at $\alpha = 0.05$ was used to assess the null hypothesis of *no clear user preference between the evaluated methods*.

The results, shown in Fig. 11, indicate that LPCE achieves the highest quality score with a positive z-score, demonstrating performance well above the average level in terms of user preference. LPGT* also scores positively, ranking second, and is perceived as slightly better than average. In contrast, GCN2$_{rm}$, GPS$_{rm}$, and GMoE$_r$ all receive negative z-scores, indicating below-average perceived quality. There is no statistically significant difference between GPS$_{rm}$ and GMoE$_r$ ($p = 0.448$). These results confirm a statistically significant user preference for LPCE over all competing methods ($p < 0.05$).

5.5 Summary

The quantitative comparison results demonstrate that LPCE enhances label placement quality by jointly refining layout features and incorporating prior experience from existing layouts. Moreover, it achieves a good balance between computational efficiency and model complexity. The qualitative results and the user study indicate that the layouts generated by LPCE are more aesthetically pleasing. Findings from ablation studies, based on variations in the key designs of LPCE, confirm the effectiveness of these designs in influencing overall performance.

6 Discussion

Our method shows encouraging results but still has limitations. The cluster-based gating function is not explicitly supervised with a separate loss. Instead, it is optimized indirectly through end-to-end training with the overall network objectives, which may limit the clustering quality. Moreover, the current expert design does not incorporate domain-specific knowledge such as layout styles, which has become an emerging focus in recent MoE research. We plan to introduce cluster-level supervision to enhance expert assignment and explore style-aware or domain-specific experts to improve interpretability and label placement quality.

LPCE's scalability on larger datasets and in real-time scenarios also requires attention. To address potential limitations in inference speed and computational load, we propose two directions: 1) parameter sharing and distillation, by sharing weights between the main and RA-module encoders, reusing identical Transformer blocks across modules, and applying knowledge distillation to train a compact student model that mimics LPCE's outputs while retaining accuracy; and 2) graph sparsification, by pruning low-importance edges from the fully connected graph to reduce computational overhead while preserving essential structure.

Although our model is proposed for label placement, its core design ideas have broader implications for the visualization research community. First, the MoCE architecture offers a general strategy for handling heterogeneous design scenarios. Instead of relying on a single model, our use of specialized sub-models for different layout patterns can be extended to tasks like adaptive chart annotation and infographic layout generation. Second, the RA strategy provides a practical way to leverage prior design knowledge in data-driven systems. By incorporating reference layouts, the model improves prediction quality based on past examples. This approach could inform tools that adapt to user intent, such as chart recommenders or intelligent authoring interfaces.

7 Conclusion

In this paper, we propose the idea of *mixture of cluster-guided experts* for label placement, with multiple experts collaboratively refining the characteristics of label layouts. Building on the idea, we develop a Label Placement Cluster-guided Experts (LPCE) network to learn optimal label positions. In LPCE, a MoCE layer is built by integrating multiple FFNs in a graph Transformer block, where each expert consists of a node FFN and an edge FFN. A cluster-based gating function is designed to direct each input sample to an appropriate expert through representation clustering. In addition, a retrieval augmentation strategy is introduced into LPCE. The reference layouts are retrieved to promote the quality of predicted layouts. Experimental results demonstrate the effectiveness and efficiency of LPCE.

Acknowledgments

This work was supported by Natural Science Foundation of Chongqing, China (No. CSTB2023NSCQ-MSX0881) and Fundamental Research Funds for the Central Universities (No. SWU-KR22032).

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. *J. WSCG*, 13(1):1–8, 2005. 2

[3] A. Asai, S. Min, Z. Zhong, and D. Chen. ACL 2023 Tutorial: Retrieval-based language models and applications. In *Proc. of ACL*, pp. 41–46. ACL, Stroudsburg, 2023. doi: 10.18653/v1/2023.acl-tutorials.6 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 1, 3

[6] A. Blattmann, R. Rombach, K. Oktay, J. Müller, and B. Ommer. Retrieval-augmented diffusion models. In *Proc. of NeurIPS*, pp. 15309–15324. MIT Press, Cambridge, 2022. doi: 10.48550/arXiv.2204.11824 2

[7] P. Bobák, L. Čmolík, and M. Čadík. Reinforced Labels: Multi-agent deep reinforcement learning for point-feature label placement. *IEEE Trans. Vis. Comput. Graph.*, 30(9):5908–5922, 2024. doi: 10.1109/TVCG.2023.3313729 1

[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 5, 6

[9] W. Cai, J. Jiang, F. Wang, J. Tang, S. Kim, and J. Huang. A survey on mixture of experts. *CoRR*, abs/2407.06204, 2024. doi: 10.48550/arXiv.2407.06204 5, 9

[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: doi/abs/10.5555/3524938.3525099 5, 6

[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] 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

[13] G. Cipriano and M. Gleicher. Text scaffolds for effective surface labeling. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1675–1682, 2008. doi: 10.1109/TVCG.2008.168 2

[14] L. Čmolík and J. Bittner. Real-time external labeling of ghosted views. *IEEE Trans. Vis. Comput. Graph.*, 25(7):2458–2470, 2019. 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

[16] W. Fedus, B. Zoph, and N. Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(120):1–39, 2022. doi: doi/abs/10.5555/3586589.3586709 2

[17] S. Fu, N. Tamir, S. Sundaram, L. Chai, R. Zhang, T. Dekel, and P. Isola. DreamSim: Learning new dimensions of human visual similarity using synthetic data. In *Proc. of NeurIPS*, pp. 50742–50768. MIT Press, Cambridge, 2023. doi: doi/10.5555/3666122.3668330 3

[18] S. Gedicke, A. Jabrayilov, B. Niedermann, P. Mutzel, and J.-H. Haunert. Point feature label placement for multi-page maps on small-screen devices. *Comput. Graph.*, 100:66–80, 2021. doi: 10.1016/J.CAG.2021.07.019 2

[19] T. Götzelmann, K. Ali, K. Hartmann, and T. Strothotte. Adaptive labeling for illustrations. In *Proc. of Pacific Graphics*, pp. 64–66. Citeseer, 2005. 2

[20] 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 5, 6

[21] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. REALM: retrieval augmented language model pre-training. In *Proc. of ICML*, pp. 3929–3938. PMLR, New York, 2020. doi: doi/abs/10.5555/3524938.3525306 2

[22] J. Hao, M. Yang, Q. Shi, Y. Jiang, G. Zhang, and W. Zeng. FinFlier: Automating graphical overlays for financial visualizations with knowledge-grounding large language model. *IEEE Trans. Vis. Comput. Graph.*, 2024. doi: 10.1109/TVCG.2024.3514138 2

[23] 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, 5

[24] D. Horita, N. Inoue, K. Kikuchi, K. Yamaguchi, and K. Aizawa. Retrieval-augmented layout transformer for content-aware layout generation. In *Proc. of CVPR*, pp. 67–76. IEEE, New York, 2024. doi: 10.1109/CVPR52733.2024.00015 2, 4

[25] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Comput.*, 3(1):79–87, 1991. doi: 10.1162/NECO.1991.3.1.79 2

[26] 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 2, 5

[27] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. *CoRR*, abs/2401.04088, 2024. doi: 10.48550/arXiv.2401.04088 2

[28] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Comput.*, 6(2):181–214, 1994. doi: 10.1162/NECO.1994.6.2.181 2

[29] J. P. Kern and C. A. Brewer. Automation and the map label placement problem: A comparison of two GIS implementations of label placement. *Cartographic Perspectives*, (60):22–45, 2008. 5

[30] S. Kim, D. Lee, S. Kang, S. Lee, and H. Yu. Learning topology-specific experts for molecular property prediction. In *Proc. of AAAI*, pp. 8291–8299. AAAI, Menlo Park, 2023. doi: 10.1609/aaai.v37i7.26000 2

[31] 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 2

[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 6

[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, 2019. 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, 2021. 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, 2019. doi: 10.1109/TVCG.2018.2854737 2

[36] S. Lakhanpal, S. Chopra, V. Jain, A. Chadha, and M. Luo. Refining text-to-image generation: Towards accurate training-free glyph-enhanced image generation. In *Proc. of WACV*, pp. 4372–4381. IEEE, New York, 2025. doi: 10.1109/WACV61041.2025.00429 5

[37] 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

[38] F. Lu, M. Chen, H. Hsu, P. Deshpande, C. Y. Wang, and B. MacIntyre. Adaptive 3D ui placement in mixed reality using deep reinforcement learning. In *Proc. of CHI*, pp. 1–7. ACM, New York, 2024. doi: 10.1145/3613905.3651059 1, 2

[39] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proc. of SIGKDD*, pp. 1930–1939. ACM, New York, 2018. doi: 10.1145/3219819.3220007 2

[40] L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *J. Mach. Learn. Res.*, 9(11):2579–2605, 2008. Available at http://jmlr.org/papers/v9/vandermaaten08a.html. 5

[41] B. Niedermann, M. Nöllenburg, and I. Rutter. Radial contour labeling with straight leaders. In *Proc. of PacificVis*, pp. 295–304. IEEE, New York, 2017. doi: 10.1109/PACIFICVIS.2017.8031608 2

[42] V. Pavlovec. Mixed Labeling: Integrating of internal and external labeling. Master thesis, Czech Technical University in Prague, 2019. 2

[43] V. Pavlovec and L. Čmolík. Rapid Labels: Point-feature labeling on gpu. *IEEE Trans. Vis. Comput. Graph.*, 28(1):604–613, 2022. doi: 10.1109/TVCG.2021.3114854 1, 2

[44] 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 9

[45] 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

[46] J. Qu, P. Zhang, E. Che, Y. Chen, and H. Ling. Graph transformer for label placement. *IEEE Trans. Vis. Comput. Graph.*, 31(1):1257–1267, 2025. doi: 10.1109/TVCG.2024.3456141 1, 2, 3, 5, 6, 7

[47] 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: doi/10.5555/3600270.3601324 5, 6

[48] 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

[49] C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. Susano Pinto, D. Keysers, and N. Houlsby. Scaling vision with sparse mixture of experts. In *Proc. of NeurIPS*, pp. 8583–8595. MIT Press, Cambridge, 2021. doi: doi/10.5555/3540261.3540918 2

[50] T. Ropinski, J.-S. Praßni, J. Roters, and K. H. Hinrichs. Internal labels as shape cues for medical illustration. In *Proc. of VMV*, pp. 203–212. Citeseer, Aka GmbH, 2007. 2

[51] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously Large Neural Networks: The sparsely-gated mixture-of-experts layer. In *Proc. of ICLR*, 2017. doi: 10.48550/arXiv.1701.06538 5

[52] L. Shen, H. Li, Y. Wang, T. Luo, Y. Luo, and H. Qu. Data Playwright: Authoring data videos with annotated narration. *IEEE Trans. Vis. Comput. Graph.*, 2024. doi: 10.1109/TVCG.2024.3477926 2

[53] S. Sheynin, O. Ashual, A. Polyak, U. Singer, O. Gafni, E. Nachmani, and Y. Taigman. KNN-Diffusion: Image generation via large-scale retrieval. In *Proc. of ICLR*, 2023. doi: 10.48550/arXiv.2204.02849 2

[54] K. Sun, Z. Zhu, and Z. Lin. AdaGCN: Adaboosting graph convolutional networks into deep models. In *Proc. of ICLR*, 2021. doi: 10.48550/arXiv.1908.05081 2

[55] 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

[56] K. Tsukida, M. R. Gupta, et al. How to analyze paired comparison data. *UWEE Technical Report 206*, 2011. Available at https://vannevar.ece.uw.edu/techsite/papers/refer/UWEETR-2011-0004.html. 9

[57] I. Vollick, D. Vogel, M. Agrawala, and A. Hertzmann. Specifying label layout style by example. In *Proc. of UIST*, pp. 221–230. ACM, New York, 2007. doi: 10.1145/1294211.1294252 1

[58] H. Wang, Z. Jiang, Y. You, Y. Han, G. Liu, J. Srinivasa, R. Kompella, Z. Wang, et al. Graph Mixture of Experts: Learning on large-scale graphs with explicit diversity modeling. In *Proc. of NeurIPS*, pp. 50825–50837. MIT Press, Cambridge, 2023. doi: doi/10.5555/3666122.3668333 2, 5, 6, 8

[59] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei. AM-GCN: Adaptive multi-channel graph convolutional networks. In *Proc. of SIGKDD*, pp. 1243–1253. ACM, New York, 2020. doi: 10.1145/3394486.3403177 2

[60] 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

[61] 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, 2023. doi: 10.1109/TVCG.2022.3209475 1

[62] J. Woodward and J. Ruiz. Analytic review of using augmented reality for situational awareness. *IEEE Trans. Vis. Comput. Graph.*, 29(4):2166–2183, 2023. doi: 10.1109/TVCG.2022.3141585 2

[63] X. Wu, S. Huang, and F. Wei. Mixture of LORA experts. In *Proc. of ICLR*, 2024. doi: 10.48550/arXiv.2404.13628 2

[64] 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 5

[65] 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 2

[66] Y. Zhang, R. Cai, T. Chen, G. Zhang, H. Zhang, P.-Y. Chen, S. Chang, Z. Wang, and S. Liu. Robust mixture-of-expert training for convolutional neural networks. In *Proc. of ICCV*, pp. 90–101. IEEE, New York, 2023. doi: 10.1109/ICCV51070.2023.00015 2

[67] 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

[68] B. Zoph, I. Bello, S. Kumar, N. Du, Y. Huang, J. Dean, N. Shazeer, and W. Fedus. ST-MoE: Designing stable and transferable sparse expert models. *CoRR*, abs/2202.08906, 2022. doi: 10.48550/arXiv.2202.08906 4, 9

Mixture of Cluster-guided Experts for Retrieval-Augmented Label Placement

Supplemental Material

Pingshun Zhang, Enyu Che, Yinan Chen, Bingyao Huang, Haibin Ling and Jingwei Qu

1 Symbol Notations

For clarity, Tab. 1 summarizes the key symbols used in this paper along with their respective meanings.

2 Additional Experiments

2.1 Implementation Details

We use pre-trained ResNet-101 as the feature backbone. The spatial sizes of input images are set to $H = W = 256$. The feature maps of layers `conv4-23` and `conv5-3` are concatenated to form the features $\mathbf{F}$. Therefore, the dimension of the visual features is $d_v = 3072$. We set the numbers of Transformer blocks and MoCE-Transformer blocks to $n_t = 2$ and $n_m = 1$, respectively. The hidden dimensions of the node and edge representations are both set to $d = 288$. The QKV vector dimensions in the node and edge MHSA layers are both set to 18. Both MHSA layers have 16 attention heads and an output dimension of 288. The hidden dimensions of the node and edge FFNs are set to 96 and 48, respectively. Specifically, the hidden dimensions of the expert FFNs are all set to 96. The dimension of the graph representation is set to $d_h = 300$. For the reference feature extractor, we set the number of Transformer blocks to $n_s = 3$, with other settings consistent with the above LPCE configurations.

AdamW optimizer is used for training the feature extractor and LPCE. The hyperparameters are $\varepsilon =$ 1e-8, $(\beta_1, \beta_2) = (0.9, 0.999)$, and weight decay of 0.01. Initial learning rates are set to 1e-4 and 8e-5 for training the extractor and LPCE, respectively, both halved at regular intervals. Batch sizes are 16 and 8, respectively. Before training LPCE, we pre-train the reference feature extractor using the focal loss (Eq. (2)) and freeze it afterward. Then, three losses proposed in [46] namely displacement loss, distance loss, and overlap loss, are used to guide the training of LPCE, with original weights of 1.0, 0.2, and 0.008, respectively. All experiments are run on a single NVIDIA GeForce RTX 4090 GPU and an Intel Core i9-13900k CPU. LPCE and other models are implemented using PyTorch and PyTorch Geometric library. We will release our algorithm upon the publication of this paper.

2.2 Calculations of Metrics

The calculations of the three metrics—PCK, IoU, and Overlap—are introduced as follows.

Table 1: Symbol notations.

Symbol	Meaning
$\mathbb{L}, \mathbb{D}$	sets of label information, displacements
$\mathbb{G}, \mathbb{V}, \mathbb{E}, \mathbb{N}$	complete graph, sets of node vectors, edge vectors, neighbors
$\mathbb{R}$	sets of real numbers
$\mathbf{I}, \mathbf{F}, \mathbf{H}, \mathbf{C}$	input image, convolutional feature maps, style feature matrix, count matrix
$\mathbf{a}, \mathbf{b}, \mathbf{d}$	anchor, box size, displacement
$\mathbf{z}, \mathbf{e}, \mathbf{h}$	node representation, edge representation, style feature
$\mathbf{y}, \mathbf{p}, \mathbf{c}$	probability distribution, assignment probability, centroid parameters
$\mathcal{E}, \mathcal{D}$	encoder, decoder
$\mathcal{F}, \mathcal{P}, \mathcal{L}$	fusion function of style features, predictor, focal loss function
K	number of entities
n, d	quantity, dimensionality
g, v	superscripts or subscripts of geometric feature, visual feature
z, e, a	superscripts of node, edge, augmented feature
r, m, e	subscripts of RA, MoCE, expert

- PCK@τ is defined as the ratio of successfully placed labels to the total number of labels [46, 64]:

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

  where h_l and w_l are the height and width of the label layout, $\mathbf{l}_i = \mathbf{a}_i + \mathbf{d}_i$ and $\hat{\mathbf{l}}_i = \mathbf{a}_i + \hat{\mathbf{d}}_i$ denote the ground-truth and predicted label positions, respectively, computed by adding the displacements to the anchor position. The indicator function $\mathbb{1}$ returns 1 if the distance between predicted position $\hat{\mathbf{l}}_i$ and ground-truth position $\mathbf{l}_i$ is smaller than $\tau \cdot \max \left(h_l, w_l \right)$.

- IoU is calculated based on predicted and ground-truth label boxes [48]:

$$\mathrm{IoU} = \frac{1}{n_l} \sum_{i=1}^{n_l} \frac{\left| \mathbf{o}_i \cap \hat{\mathbf{o}}_i \right|}{\left| \mathbf{o}_i \cup \hat{\mathbf{o}}_i \right|} \tag{2}$$

  Each label box is represented by its center and size, where the center corresponds to the label position, *i.e.*, $\mathbf{o}_i = \left[\mathbf{l}_i, \mathbf{b}_i \right]$ and $\hat{\mathbf{o}}_i = \left[\hat{\mathbf{l}}_i, \mathbf{b}_i \right]$.

- Overlap is computed as the average IoU over all pairs of label boxes [36]:

$$\mathrm{Overlap} = \frac{2}{n_l \left(n_l - 1 \right)} \sum_{1 \leq i < j \leq n_l} \frac{\left| \hat{\mathbf{o}}_i \cap \hat{\mathbf{o}}_j \right|}{\left| \hat{\mathbf{o}}_i \cup \hat{\mathbf{o}}_j \right|} \tag{3}$$

2.3 Qualitative Results

The predicted label layouts of several samples from the SWU-AMIL dataset by GCN2$_{rm}$ [10], GPS$_{rm}$ [47], GMoE$_r$ [58], LPGT* [46], and LPCE are visualized in Fig. 1.

To demonstrate the generalizability of LPCE, we test it on several unseen samples from [14, 15, 42]. We manually annotate them and provide anchor points for prediction. The LPCE model, trained on the SWU-AMIL dataset, is directly applied to generate label layouts.

- *Pingshun Zhang, Enyu Che, Yinan Chen, Bingyao Huang and Jingwei Qu are with College of Computer and Information Science, Southwest University. E-mail: z2211973606@email.swu.edu.cn, enyuche@gmail.com, out1147205215@outlook.com, bhuang@swu.edu.cn, qujingwei@swu.edu.cn.*
- *Haibin Ling is with Department of Artificial Intelligence, Westlake University. E-mail: linghaibin@westlake.edu.cn.*
- *Jingwei Qu is the corresponding author.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

The predicted and corresponding human-labeled layouts for four unseen samples are visualized in Fig. 2. LPCE performs well on the "Human Head" and "Wheel Fork" samples, producing clear and well-aligned layouts. For the "Digestive System" sample, although the label "Gullet" is placed on the right side rather than the left as in the human-labeled layout, the overall layout remains acceptable and readable. The "Spheres" sample poses challenges in terms of its dense and uniform structure. In this case, LPCE's output shows some label overlaps and occlusions of target elements. In summary, these results demonstrate that LPCE generalizes well to unseen samples without retraining.

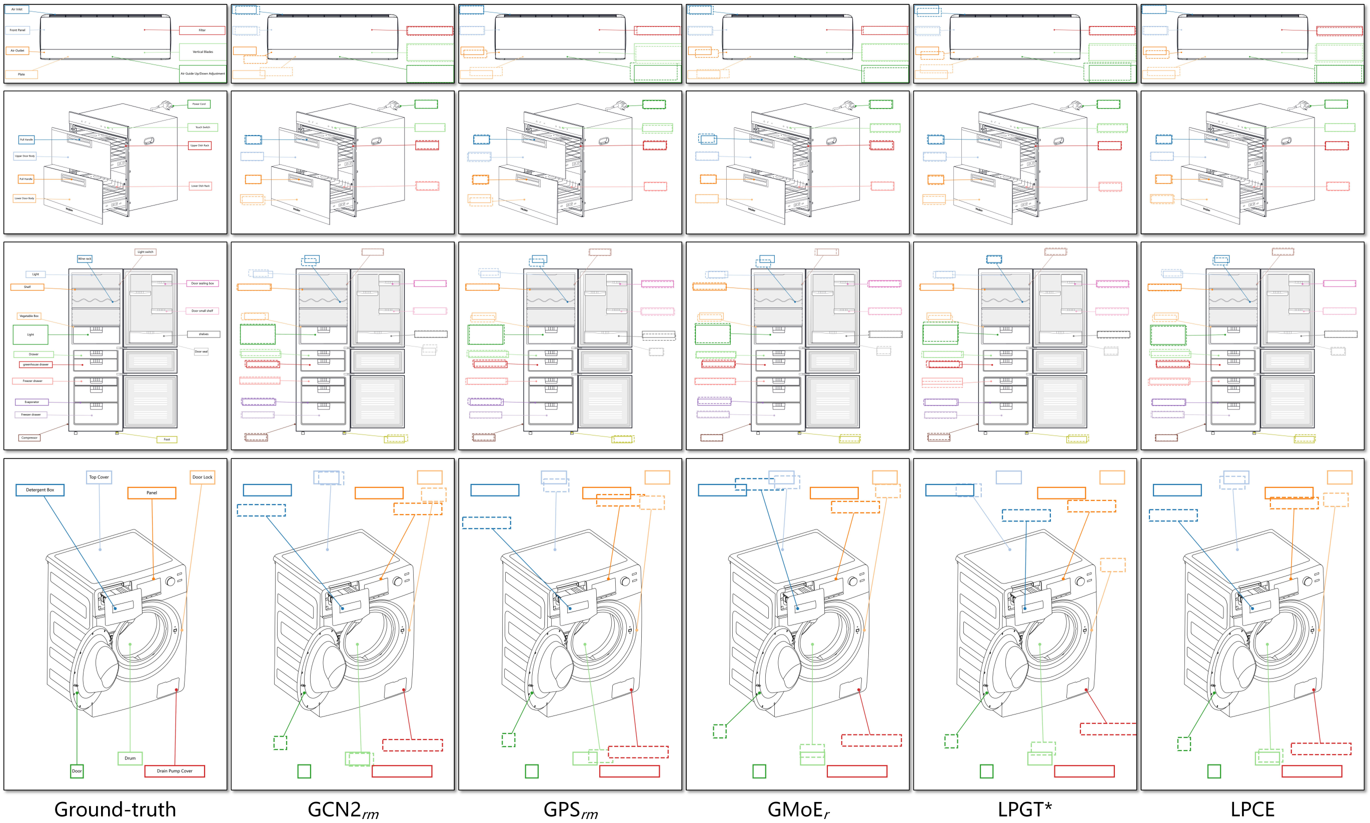
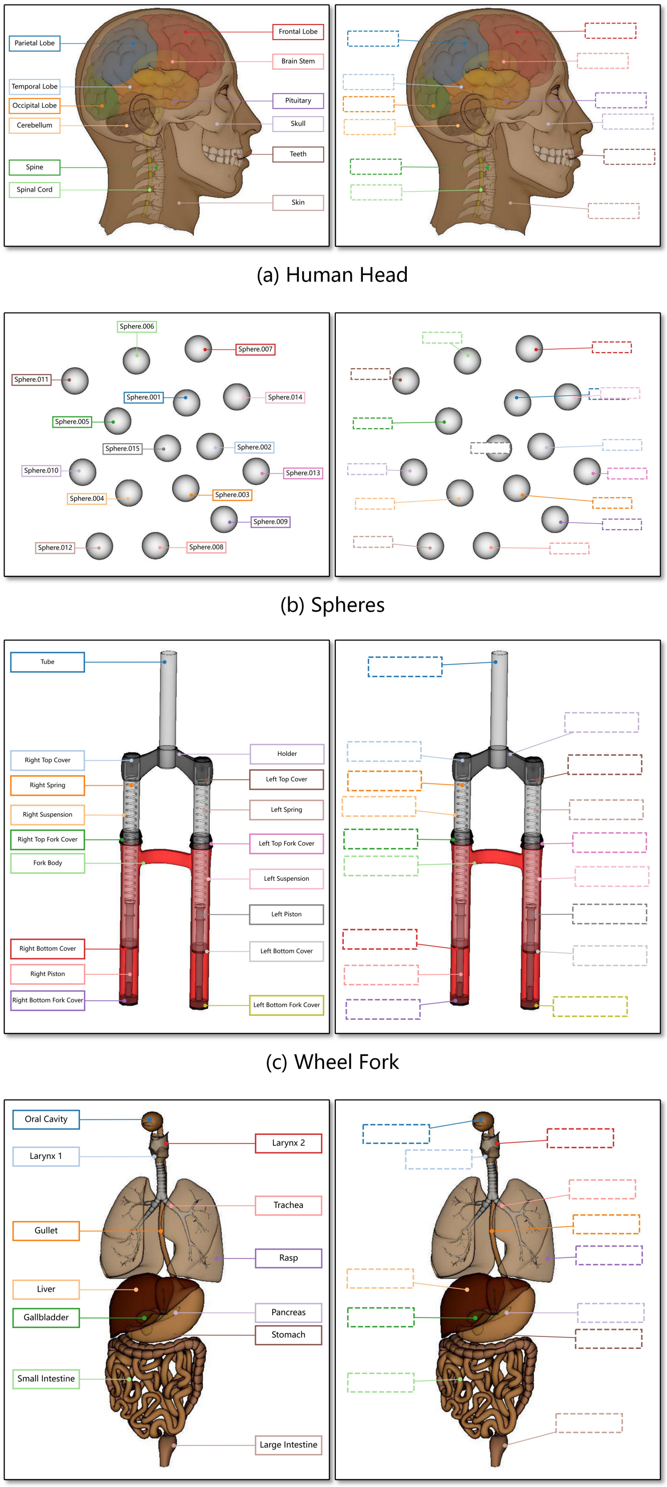


Fig. 1: Visualization results on the SWU-AMIL dataset. Solid boxes represent ground-truth label positions, while dashed boxes indicate predicted ones. Different colors distinguish labels, and label texts are omitted in predictions for clarity.



(a) Human Head

(b) Spheres

(c) Wheel Fork

(d) Digestive System

Fig. 2: Visualization results on some unseen samples from [14, 15, 42]. Solid boxes represent human-labeled label positions, while dashed boxes indicate positions predicted by LPCE. Different colors distinguish labels, and label texts are omitted in predictions for clarity.