Unsupervised Graphic Layout Grouping with Transformers

Jialiang Zhu[1], Danqing Huang[2], Chunyu Wang[2], Mingxi Cheng[2]
Ji Li[2], Han Hu[2], Xin Geng[1], Baining Guo[1,2*]
[1]Southeast University, [2]Microsoft Research Asia
jialiang.zhu@outlook.com, dahua@microsoft.com, chnuwa@microsoft.com, mingxicheng@microsoft.com
jili5@microsoft.com, hanhu@microsoft.com, xgeng@seu.edu.cn, bainguo@microsoft.com

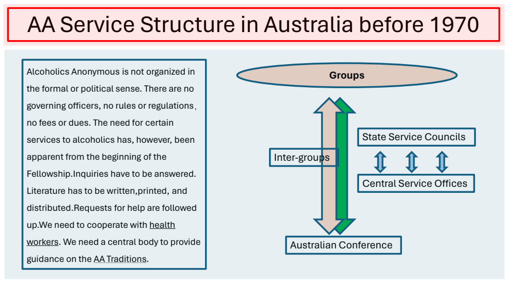
Abstract

Graphic design conveys messages through the combination of text, images and other visual elements. Unstructured designs such as overloaded social media graphics may fail to communicate their intended messages effectively. To address this issue, layout grouping offers a solution by organizing design elements into perceptual groups. While most methods rely on heuristic Gestalt principles, they often lack the context modeling ability needed to handle complex layouts. In this work, we reformulate the layout grouping task as a set prediction problem. It uses Transformers to learn a set of group tokens at various hierarchies, enabling it to reason the membership of the elements more effectively. The self-attention mechanism in Transformers boosts its context modeling ability, which enables it to handle complex layouts more accurately. To reduce annotation costs, we also propose an unsupervised learning strategy that pre-trains on noisy pseudo-labels induced by a novel heuristic algorithm. This approach then bootstraps to self-refine the noisy labels, further improving the accuracy of our model. Our extensive experiments demonstrate the effectiveness of our method, which outperforms existing state-of-the-art approaches in terms of accuracy and efficiency.

1. Introduction

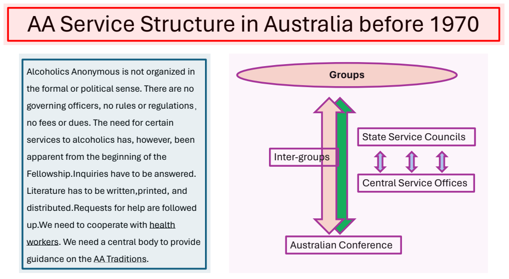
Layout grouping aims to organize the elements in a graphic design into perceptual groups, forming a hierarchical structure for effective message delivery. For example in Fig. 1, elements in the slide can be grouped at two different levels: (a) the text box along with the diagram can be assigned into one large group; or (b) besides the title box, the text box on the left and the complicated diagrams composed of many elements can be assigned into two separate groups. By parsing the group structure, we can enable many potential applications in graphic design intelligence, such

[*]Corresponding author.


AA Service Structure in Australia before 1970

Alcoholics Anonymous is not organized in the formal or political sense. There are no governing officers, no rules or regulations, no fees or dues. The need for certain services to alcoholics has, however, been apparent from the beginning of the Fellowship.Inquiries have to be answered. Literature has to be written,printed, and distributed.Requests for help are followed up.We need to cooperate with health workers. We need a central body to provide guidance on the AA Traditions.

(a) large coarse groups


AA Service Structure in Australia before 1970

Alcoholics Anonymous is not organized in the formal or political sense. There are no governing officers, no rules or regulations, no fees or dues. The need for certain services to alcoholics has, however, been apparent from the beginning of the Fellowship.Inquiries have to be answered. Literature has to be written,printed, and distributed.Requests for help are followed up.We need to cooperate with health workers. We need a central body to provide guidance on the AA Traditions.

(b) small fine-grained groups

Figure 1. An example slide with groupings in two levels, (a) large coarse groups to (b) small fine-grained groups. Elements outlined with the same color rectangle represent one group. Transparent overlays in various colors visualize distinct group regions.

as graphic design understanding and summarizing, content extraction and generation, graphic design beautification and animation creation.

Layout grouping is challenging because of the combinatorial complexity of possible layout groups. The existing methods mainly use simple heuristics such as Gestalt laws [15] to parse the layout structures [4, 19, 24]. For example, proximity assumes that elements which are close in distance are more likely to be related and united, and vice versa. However, the lack of context modeling ability makes them less effective in handling complex designs. Recently, there have been some machine learning-based attempts to tackle this issue, such as those proposed in [25, 31], which predict the relatedness between pairs of elements and iteratively merge them into groups using a pre-defined threshold. While they show promising results, they are limited to training on pairwise labeled data due to the lack of available datasets with hierarchical group annotation. Moreover, the training objective of pairwise relatedness is not directly aligned with the goal of hierarchical grouping, and relying solely on threshold merging in post-processing cannot guarantee a reasonable grouping hierarchy.

In this paper, we propose a highly performant approach for graphic layout grouping using the Transformer architecture. It learns a set of group tokens at various levels, to directly reason about the membership between each layout element and the group tokens. The group tokens are aware of the global context since they attend to all elements via the attention mechanism. Specifically, given a list of elements, our model progressively combines elements into groups with L grouping layers. We use a set prediction loss which matches the predicted groups to the target groups using a bipartite graph. However, we find that this initial model barely converges during training. To address this, we incorporate explicit positional priors by enforcing each group token to predict the bounding box of the corresponding group. The mechanism allows the group tokens to focus on specific regions, resulting in faster convergence and better grouping results.

We propose an approach to train the network in an unsupervised way. Firstly, we introduce a heuristic algorithm that utilizes the Gestalt principles to generate pseudo-labels, which recursively segments the elements based on their proximity to each other. Although the pseudo-labels may be incorrect in challenging cases, we find that they suffice to train a reasonable initial model. Then we adopt bootstrapping [5] to automatically refine the noisy labels and re-train the model. Particularly, after each epoch, we make grouping predictions using the current model. If the confidence of the predictions is larger than a threshold, we trust the predictions and use them to replace the original pseudo-labels. The threshold gradually decreases as the model becomes more accurate during training. We observe that the model can get accurate predictions on complex layouts, demonstrating the effectiveness of bootstrapping. The unsupervised training strategy allows the model to benefit from the large-scale unlabeled data.

We construct a dataset of presentation slides for evaluation. We manually annotate the slides with each annotation being jointly reviewed by two people to ensure its quality. We compare our method with the baselines on this dataset. Besides, we also send the results to a group of people for human evaluation. Both quantitative and qualitative results show that our model achieves much better results. In particular, our method shows strong performance in handling complex layouts with many elements. Moreover, we conduct extensive ablation studies and will share some interesting observations in the experiment section.

2. Related Work

2.1. Visual Grouping

We classify the existing methods into pixel-based and object-based ones. The former produces a hierarchical image representation. Some methods use hand-crafted features such as Gestalt cues (e.g., contour, texture and brightness) [22] and normalized cuts [1, 21, 26] to group pixels into superpixels. Recent approaches also use deep features for perceptual grouping. For example, [7, 9] use CNN's low resolution features for upsampling [17, 18] learn the multi-scale deep graph features with LSTM. Furthermore, end-to-end trainable models are proposed, such as DGM [16] with hierarchical graph operations and GroupViT [32] with Transformer-based layers.

Beyond pixels, directly grouping objects is studied for graphic layouts. Several work target at grouping web pages [4, 33] which leverage rich structures inherited from the HTML DOM tree. Without such information, other work [11, 19, 20, 24, 33] implement heuristic parsing rules based on Gestalt laws (e.g., proximity, similarity) to identify the groupings. For example, [14] extract features to compute pairwise distance and iteratively combine objects with thresholds. More recent work are mostly learning-based. CanvasEmb [31] first pre-trains a large-scale unsupervised representation and then predicts the pairwise object relations by fine-tuning on a small annotated data. Similarly, Shi et al. [25] also train a model to predict the relatedness of object pairs. To recover the hierarchical structure, they recursively merge objects with relatedness score larger than an empirical threshold in the post-processing step. Different from the previous work, we propose an end-to-end trainable model to progressively group objects without the heavy cost of group label annotation.

2.2. Graphic Design Layout Understanding

Some work focus on element detection from pixel-based input, which separates the layout image into different element regions and classify their role (e.g., text, table, figure). Traditional approaches [6, 12, 27] propose algorithms with primitive heuristics to recursively merge pixel lines into element regions. Later work adopt neural network to automatically extract features for better detection [8, 28]. For a more systematic summary of related work, please refer to [2]. Besides detecting the elements, their inter-relationships are also very important in parsing and understanding the layout. Visual grouping studies how elements can be organized into a hierarchical structure for more effective message communication. This paper focus on this task and will introduce more in the next sections.

3. Approach

3.1. Task Formulation

Each graphic layout is represented as a sequence of tokens $\{e_i\}_{i=0}^{N}$, where each token e_i contains a fixed set of properties $\{p_i^k\}_{k=0}^{K}$. The properties include type, position, color, font size, etc. The task is to organize the elements into groups in L different levels where each level is responsible for one granularity and contains G^l groups. Please note that $G^l > G^{l+1}$ to ensure the fine-to-coarse granularities.

According to the Gestalt psychology, grouping can be accomplished by considering but not limited to the following laws:

- **Proximity**: Objects that are closer to one another are perceived to be more related than objects that are spaced farther apart.
- **Similarity**: Objects that are similar in nature (such as size, shape, or color) tend to be grouped together.
- **Continuity**: Objects that appear to be connected are grouped together. For example, lines and arrows can be indicators to connect elements.

3.2. Grouping with Transformers

An overview of the system is shown in Fig. 2. Our model consists of an element encoder and L Transformer-based grouping layers.

Element Encoder. We represent each element with properties (e.g., type, position) in the embedding space by concatenating the property embeddings $\{\mathbf{p}^k\}_{k=0}^{K}$:

$$\mathbf{e}_i = \mathrm{MLP}(\mathbf{p}_i^1 \oplus \mathbf{p}_i^2 \oplus ... \oplus \mathbf{p}_i^K) \tag{1}$$

where $\oplus$ is the concatenation operator and MLP is a multilayer perceptron. For properties with categorical values such as type and color, we use the embedding matrix as the learning parameter. For properties with numerical values such as position and size, the positional encoding [30] is adopted.

Grouping Layers. Inspired by GroupViT [32], each layer l contains a Transformer-based block with G^l learnable group tokens $\{\mathbf{g}_i^l\}$. Element tokens $\{\mathbf{e}_i^l\}$ are concatenated with the group tokens into the self-attention for contextual representation learning:

$$\{\mathbf{e}_i^{l+1}\} \oplus \{\mathbf{g}_j^{l+1}\} = \mathrm{Transformer}(\{\mathbf{e}_i^l\} \oplus \{\mathbf{g}_j^l\}) \tag{2}$$

On top is an element-to-group cross attention which partitions elements $\{\mathbf{e}_i^l\}$ into G^l groups and obtains the new element tokens $\{\mathbf{e}_i^{l+1}\}$ in the next layer:

$$\{\mathbf{e}_i^{l+1}\} = \mathrm{CrossAttn}(\{\mathbf{e}_i^l\}, \{\mathbf{g}_j^l\}) \tag{3}$$

Specifically the CrossAttn assigns each element token $\mathbf{e}_i^l$ to a group token $\mathbf{g}_j^l$ with maximum score in the element-to-group attention matrix (i.e., $\mathrm{argmax}_{j\in G^l}(A_{i,j}^l)$).

Since the one-hot assignment operation via argmax is non-differentiable, we instead obtain the attention matrix using `Gumbel-Softmax` [10]:

$$A_{i,j}^l = \frac{\exp(W_q \mathbf{g}_j^l W_k \mathbf{e}_i^l + \gamma_j)}{\sum_{h=0}^{G^l} \exp(W_q \mathbf{g}_h^l W_k \mathbf{e}_i^l + \gamma_h)} \tag{4}$$

where W_q and W_k are the linear projection weights for group and element tokens respectively. $\{\gamma_h\}$ of groups $\{0, \ldots, h, \ldots, G^l\}$ are i.i.d random samples drawn from the `Gumbel(0, 1)` distribution. We then apply the straight-through trick in [29] to assign elements to corresponding groups:

$$\hat{A}^l = \texttt{one-hot}(A_{\text{argmax}}^l) + A^l - \texttt{sg}(A^l) \tag{5}$$

where `sg` is the stop gradient operator. With the trick, $\hat{A}_i^l$ has the one-hot value of assignment to a single group, and its gradient is equal to the one of A_i^l, which makes the attention matrix differentiable and end-to-end trainable. The elements in the next layer $l+1$ can be obtained by merging all elements in each group in layer l:

$$\mathbf{e}_i^{l+1} = \mathbf{g}_j^l + W_o \frac{\sum_i \hat{\mathbf{A}}_{ij}^l W_v \mathbf{e}_i^l}{\sum_i \hat{\mathbf{A}}_{ij}^l} \tag{6}$$

where W_v and W_o are the learnable projection weights.

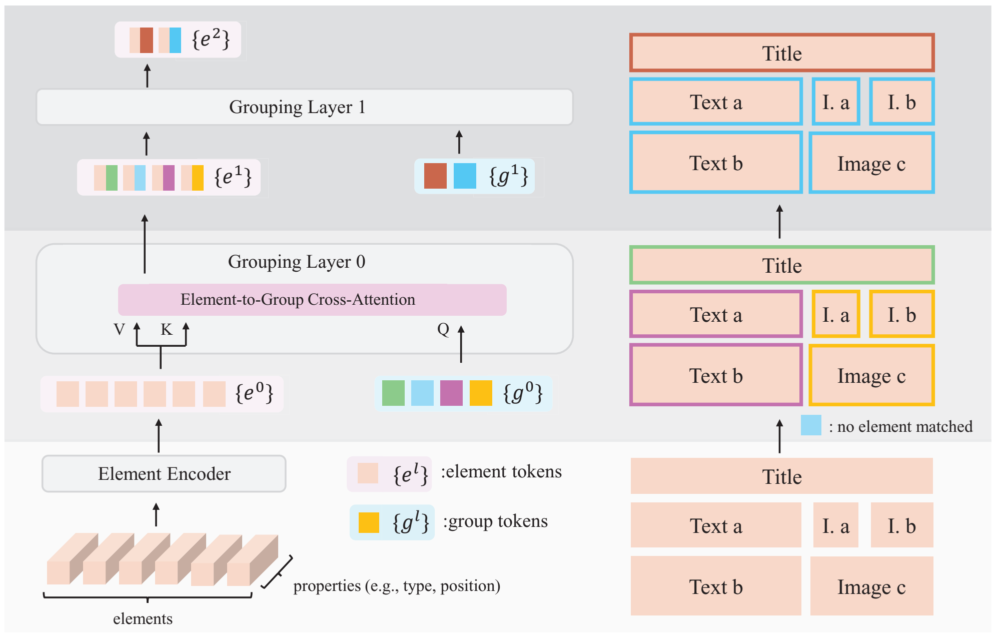
For each layer, we set the following two learning objectives.

Objective 1: Set Prediction. To train the model, we need to match the predicted groups to the target groups, which can be viewed as a set prediction problem. Inspired by recent end-to-end object detection framework [3], we conduct a bipartite graph matching with score function $s(g_j^l, \hat{g}_j^l)$ which considers the number of overlap elements and the box IoU between a predicted group and a target group. The calculation of the score and the BCE loss $\mathcal{L}_{set}$ are:

$$s(g_j^l, \hat{g}_j^l) = \beta_1 \frac{|g_j^l \cap \hat{g}_j^l|}{|g_j^l|} + \beta_2 \mathrm{IoU}(g_j^l, \hat{g}_j^l)$$

$$\mathcal{L}_{set} = [e_i^l \in g_j^l] \cdot \log(A_{ij}^l) + (1 - [e_i^l \in g_j^l]) \cdot \log(1 - A_{ij}^l) \tag{7}$$

where $\{\beta\}$ are the hyper-parameters to sum the two weights.



(a) Model architecture (b) Hierarchical grouping

Figure 2. The overview of our approach. (a): model architecture with fine-to-coarse grouping layers to progressively combine elements $\{e\}$ with learnable group tokens $\{g\}$. (b): an example of the hierarchical grouping process, elements outlined with the same color rectangle represent one group, corresponding to the "group tokens" in (a).

Objective 2: Anchor Prediction. In addition, we add an auxiliary prediction head to predict the box coordinate of each group (i.e., the minimum box covering all elements in the group) using the group token. By injecting explicit positional priors, this does not only help improve group token representation, but also eliminate the slow training convergence issue in set matching.

$$bbox_j^l = \text{sigmoid}(\text{MLP}(g_j^l))$$

$$\mathcal{L}_{anchor} = \lambda_1 ||bbox_j^l - \hat{bbox_j}^l||_1 + \lambda_2 \mathcal{L}_{iou}(bbox_j^l, \hat{bbox_j}^l) \quad (8)$$

We use a linear combination of the L_1 loss and the generalized IoU loss [23] which is scale-invariant. In total, our objective contains two losses: (1) set prediction loss; (2) group anchor loss:

$$\mathcal{L} = \lambda_3 \mathcal{L}_{set} + \lambda_4 \mathcal{L}_{anchor} \quad (9)$$

where $\{\lambda\}$ are hyperparameters for the loss weights.

3.3. Unsupervised Bootstrapped Training

Labeling of the visual groupings can be time-consuming. To the best of our knowledge, currently there are no available datasets with complete hierarchical annotation. To alleviate the data bottleneck, this paper proposes a two-stage unsupervised training without annotation cost, namely pre-training and bootstrapping.

3.3.1 Pre-Training with Pseudo-Labeling

Similar to the traditional rule-based approaches in visual grouping [11,24], we propose a heuristic labeling algorithm using Gestalt principle of proximity to generate pseudo-annotations. Intuitively, we assume that elements with closer distance tend to be in the same group as mentioned in Sec. 3.1. As shown in Algorithm 1, the main function is `SegmentRegion`. Given a list of elements, we sort them in the ascending order of top-left corner coordinate (x_l, y_t). By iterating the ordered list, we calculate the horizontal or vertical margin between two neighbouring elements. If the margin is larger than a pre-defined threshold, we segment the two elements, otherwise we keep them in the same group. Starting with all the elements in a layout, we recursively apply the function `SegmentRegion` for each segmented group and finally obtain the hierarchical labels. In this way, we are able to collect a large-scale pseudo-labels for pre-training.

3.3.2 Bootstrapping with Self-Improved Labels.

Although we can get a reasonable initial model with pre-training, pseudo-labels are noisy and might limit the model learning capability. Hence we adopt a simple but effective bootstrapping strategy to self-refine the training labels using the model predictions. For each training instance, we compute the model confidence by averaging the element-

Algorithm 1 Heuristic Label Induction.

```
Require: $N$ elements, each with box coordinate $(x_l, y_t, x_r, y_b)$,
    margin threshold $m$
 1: SEGMENTREGION($\{e_i\}_0^N$, $xAxis$, $m$)
 2: procedure SEGMENTREGION($\{e_i\}_0^G$, $d$, $m$)
 3:     Sort $\{e_i\}_0^G$ in ascending order of left-top $(x_l, y_t)$
 4:     $end_x, end_y = e_0[x_r], e_0[y_b]$
 5:     $offset = 0$
 6:     for $i = 1, \dots, G$ do
 7:         if $d = xAxis$ and $e_i[x_r] - end_x > m$ then
 8:             group elements $\{e\}_{offset}^{i-1}$
 9:             SEGMENTREGION($\{e\}_{offset}^{i-1}$, $yAxis$, $m$)
10:         else if $d = yAxis$ and $e_i[y_b] - end_y > m$ then
11:             group elements $\{e\}_{offset}^{i-1}$
12:             SEGMENTREGION($\{e\}_{offset}^{i-1}$, $xAxis$, $m$)
13:         else
14:             $end_x = max(end_x, e_i[x_r])$
15:             $end_y = max(end_y, e_i[y_b])$
```

to-group attention scores:

$$c(\{g^l\}) = \text{avg} \sum_j \frac{(\sum_{e_i \in g_j} A_{ij})}{|g_j|} \tag{10}$$

After training the model in several epochs to a stable checkpoint, our bootstrapping replaces the training labels with our model predictions if the model confidence is larger than a threshold. We empirically try different threshold strategies and will show more results in the experiments.

4. Experiments

4.1. Experimental Settings

To the best of our knowledge, there are currently no publicly available datasets, baselines, or well-defined quantitative metrics for evaluating the task of visual grouping. In this paper, we formally define the settings and establish a systematic approach for evaluating this task.

4.1.1 Datasets.

We start by collecting a large-scale dataset of public presentation slides in the *.pptx* format for training our model. To obtain grouping annotations, we utilize our proposed heuristic algorithm presented in Sec. 3.3.1 to assign pseudo-labels to each slide. We then filter out the training samples that yield empty group results. Our final training dataset consists of **369,347** slides, with each slide containing an average of approximately 2 levels of pseudo-groups. More specifically, there are 4.82 groups in each slide in level 0 (fine-grained) on average, with each group containing 2.28 shapes. In level 1 (coarse-grained), there are 2.93 groups on average, each with 3.92 shapes. For evaluation, we created a human-labeled dataset consisting of 79 slides. Two annotators were provided with guidelines and examples to group the shapes in a slide hierarchically. The evaluation set consists of slides with at most two levels of annotated groups, with an average of 4.11 groups, each with 2.06 shapes at level 0 and an average of 2.25 groups, each with 4.01 shapes at level 1. More information about the annotation process can be found in the supplementary materials.

4.1.2 Evaluation Metrics.

We propose two quantitative metrics to evaluate the system performance. Given a prediction with P levels of groups $\{g_P\}_{l=0}^{l_{P-1}}$ and a target with T levels $\{g_T\}_{l=0}^{l_{T-1}}$, we define (1) **Acc (soft)** to determine a prediction as correct if any level in the prediction matches with any level in the target. On the other hand, (2) **Acc (strict)** requires a stricter exact match where each level of group prediction must be equal to the corresponding level in the target. The metric formulas are as follows:

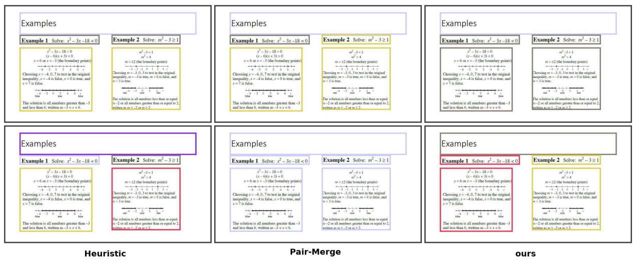
$$\text{Acc(soft)} = \frac{1}{|\{g_T\}|} \sum_{p,t \in \{g_P\},\{g_T\}} \left[\bigvee_{l_i=0}^{l_P-1} \bigvee_{l_j=0}^{l_T-1} (p_i^l = t_j^l) \right]$$

$$\text{Acc(strict)} = \frac{1}{|\{g_T\}|} \sum_{p,t \in \{g_P\},\{g_T\}} \left[\bigwedge_{i=0}^{l_T-1} (p_i^l = t_i^l) \right] \tag{11}$$

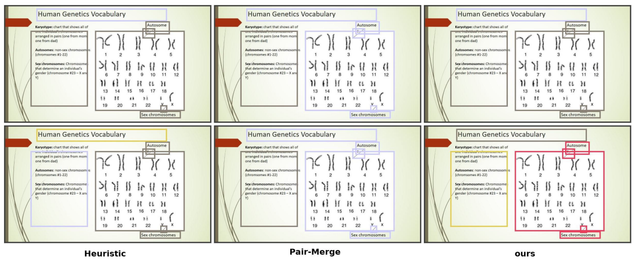
Besides, we also conduct human evaluation to see how people rate the grouping outputs by our approach and the other ones. We invite 3 participants and each participant is given 60 pairs of grouping candidates. Each pair consists of our system's prediction as well as one baseline's output of the same slide. Participants are asked to determine which hierarchical grouping is better. We calculate the win rate of our approach in the user study.

4.1.3 Baselines.

We consider the following two baselines for comparison: (1) **Heuristic**: we apply our algorithm mentioned in Sec. 3.3.1 in the evaluation set to test its performance; (2) **Pair-Merge** [25]: a Transformer-based model to predict the relatedness of pairwise shapes and then iteratively merge shape pairs with a pre-determined threshold. In their original approach, the authors collect presentations with user-created groups (without complete and hierarchical annotations) on the web. In order to train their model on our pseudo-labeled hierarchical dataset, we re-implemented their approach by keeping the same Transformer backbone and training one copy of the pairwise prediction head for each grouping level (in total pre-defined L levels).



(a)



(b)

Figure 3. Two example slides with grouping predictions from heuristic algorithm, Pair-Merge, our system, and the ground truth. Each slide contains hierarchical groupings with a coarse-level (1st row) and a fine-level (2nd row).

4.1.4 Model and Training Parameters.

For model architecture, the maximum input sequence length is 224. The token embedding size and hidden size are all 384. According to the data statistics, we set two grouping layers, with the number of group tokens 42 and 8 respectively. The first grouping layer consists of 6 attention layers, each with 6 heads, while the other grouping layers each consist of 3 attention layers, also with 6 heads. The attention dropout rate is set to 0.5. We train the model for 60 epochs with a learning rate 5×10^{-5} and batch size 64 using 4 GPUs. Adam [13] is used as optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The loss weights $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are set to 1, 1, 10, 1 respectively. We use adaptive thresholds 0.7, 0.5, 0.3 during the epoch of 40, 50, 55 for bootstrapping, which will be described more in the Sec. 4.3.2. Bounding box coordinates are normalized from 0 to 1. For the heuristic algorithm, we set the margin threshold to 0.05 in both x and y axis.

Table 1. Overall results on the human labeled evaluation dataset.

Methods	Acc (soft)	Acc (strict)	win-rate
Heuristic	44.3	12.7	80.85
Pair-Merge	60.8	0.0	94.92
ours (w/o bootstrapping)	40	15	82.86
ours	**62.5**	**22.5**	-

4.2. Overall Performance

The overall results are presented in Tab. 1. Without bootstrapping, our approach is comparable to the heuristic baseline. This is expected since the model is trained to fit the noisy labeled data provided by the heuristic algorithm. We can see a large performance increase using bootstrapping (22.5% relative gain in soft accuracy and 7.5% in strict accuracy) and achieves the best results compared to the two baselines, which indicates that our model has captured representative features and bootstrapping helps correct noisy labels. Moreover, it is interesting to see that the Pair-Merge baseline performs well in terms of soft accuracy but extremely bad in strict accuracy. We then take a deeper look and observe that this baseline tends to predict correctly in one of the grouping levels but fail to merge elements hierarchically. Therefore, its strict accuracy related to the hierarchical exact match results in 0. This indicates that a system with the pairwise relatedness learning objective might not be sensitive enough to construct a hierarchical grouping structure. While being compared, our approach outperforms the others in terms of strict accuracy, demonstrating that our multiple grouping layers effectively capture the hierarchical structure.

As for human evaluation, our approach beats the heuristic and Pair-Merge baselines as well as ours (w/o bootstrapping) with a win-rate of 80.85%, 94.92% and 82.86% respectively, which is consistent with the automatic metric results.

Case Study. We show two examples with predictions from different systems as well as the ground truths in Fig. 3. For each case, we present two levels of groupings (coarse-grained in the 1st row and fine-grained in the 2nd row respectively). We can see that our approach correctly parses the slide grouping structure in both two cases. In the first case (Fig. 3a), the heuristic algorithm wrongly treats the horizontally-aligned elements as separate groups in the coarse level, which is a commonly seen error pattern since the algorithm scans along the x/y-axis to divide elements into groups. As for the second case (Fig. 3b), Pair-Merge correctly merges the the elements to two column-wise groups, but fail to combine them into a larger group for a hierarchical structure.

4.3. Model Analysis

To give a better understanding of our model, we conduct an in-depth model investigation in this subsection.

4.3.1 Group Token Analysis

Group tokens are a crucial component of our model as they learn to attend and associate with elements. In order to gain

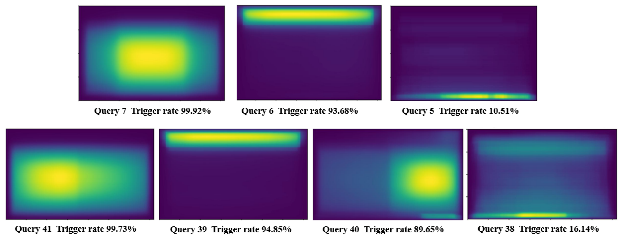


Figure 4. Heatmaps of all box predictions on 11k sampled slides from training dataset. We only visualize the active group tokens: 3 out of 8 in coarse level (1st row) and 4 out of 42 in fine-grained level (2nd row).

insight, we examine their physical meanings through visualization and conduct ablation experiments to study the impact of the number of group tokens on model performance.

Visualization. To see what the group tokens have actually learnt, we visualize the boxes predicted by different group tokens for a randomly-sampled slides ($\sim$11k) from the training dataset. As there are excessive group tokens where some are associated without any elements (i.e., not triggered), we filter those null tokens with the triggered rate less than 1% in the dataset and only show the triggered group tokens. As shown in Fig. 4, our model learns different specialization for each group token. We observe that each token has several modes of operation focusing on different regions. For example, the three group tokens in the coarse level (1st row) mainly attend in the top, center and bottom region respectively. Moreover, group tokens in different layers tend to predict boxes in different sizes. The tokens in the fine-grained level (2nd row) are responsible for smaller groups while the ones in the coarse level are more likely to favor larger groups (e.g., title + content). This observation well explains the mechanism of our hierarchical grouping model that different grouping layers handles different granularities and each group token controls its corresponding regions.

Effects of Group Token Numbers. As the group tokens have been shown to carry physical meanings via visualization, we are also curious to see if a larger number of the learnable group tokens can have a higher trigger rate and capture more meaningful patterns. Here we try different numbers of group tokens in the two corresponding group layers and show the results in Tab. 2. Moreover, we calculate the number of group tokens being triggered which has been associated with at least one element. We observe that the increasing number of group tokens does not result in better performance but instead introduce more noise. We argue that this is because there are only several key group tokens for a limited set of regions while others act as null tokens. As the group token number increases (e.g., {28, 6} to {128, 16}), the number of triggered tokens stays around {5, 3}, which further supports our argument.

Table 2. Ablation study of different numbers of group tokens in two levels $\{G^0, G^1\}$ (fine and coarse level). We also show the statistics of number of tokens being triggered (2nd column).

# Group Token	# Triggered	Acc (soft)	Acc (strict)
28 6	5 3	60.00	22.50
42 8	**7 4**	**62.50**	**22.50**
64 10	5 3	62.50	21.25
92 12	7 4	58.75	23.75
110 14	4 3	53.75	18.75
128 16	5 3	61.25	15.00

4.3.2 Effects of Bootstrapping

In this section, we investigate how different strategies of bootstrapping affect the overall results. Starting from 40 epoch, we experiment with (1) fixed bootstrapping threshold (0.7 in this study) (2) adaptive thresholds which accepts an increasing percentage of label refinement with lower thresholds (0.7, 0.5, 0.3) along training epochs. As shown in Fig. 5, both fixed (green line) and adaptive thresholds (blue line) perform better than the setting without bootstrapping (red line), indicating that our model is able to capture representative features and refine the noisy training labels correctly. Moreover, the adaptive thresholds work better than the fixed one, which means that the decreasing thresholds to include more refined labels do not deteriorate the performance but instead help denoise more training samples as our model becomes more robust during training iterations. Moreover, we calculate the percentage of re-labeled training samples during the bootstrapping epochs. As we can see, the replace rate of labels starts from 70% to 90%, which means that our model effectively utilizes the bootstrapping for data relabeling and denoising.

4.3.3 Effects of Different Learning Objectives

Our model architecture contains two prediction heads per grouping layer, set prediction and anchor prediction respectively. We would like to see how much they contribute to the overall performance. Here we try different loss weight combinations of λ_3 (set prediction), λ_4 (anchor prediction) and show the results in Tab. 3. We can see that both the two learning objectives are important, as the performance drops dramatically without any of two losses. Moreover, the setting of $\lambda_3 : \lambda_4 = 10 : 1$ achieves the best results, which indicates that the set prediction objective dominates

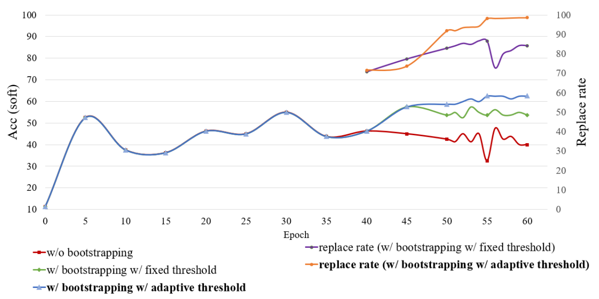


Figure 5. Accuracy curves with different bootstrapping strategies. Bootstrapping starts from epoch 40. Adaptive thresholds work the best (blue line).

the optimization while anchor prediction acts as auxiliary objective to stabilize the training.

Table 3. Results with different loss weights in terms of set prediction (λ_3) and anchor prediction (λ_4).

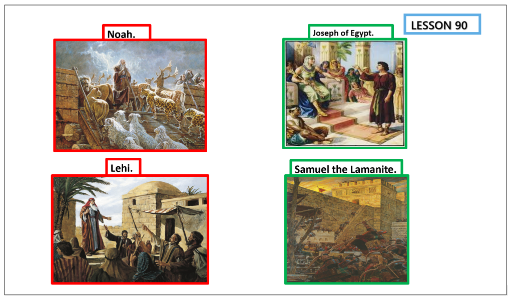
λ_3	λ_4	Acc (strict)	Acc (soft)
1	0	1.25	54.06
0	1	0.00	38.75
1	1	0.00	47.50
5	1	13.75	47.50
1	5	0.00	55.00
10	**1**	**22.50**	**62.50**
15	1	21.25	57.50

4.3.4 Error Analysis

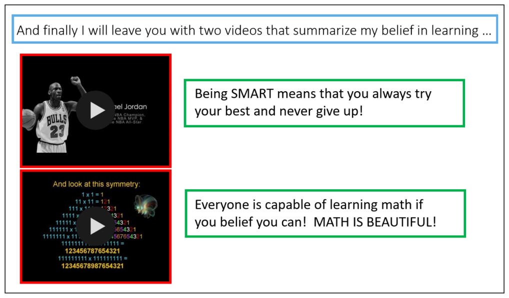
We observe two main errors that left to be solved in the future work.

Error 1: Weak Global Contextualization. Though our model has learned the interactions between elements via the attention mechanism, it is still weak at capturing repetitive patterns in a layout. As shown in Fig. 6a, there should be four image-caption groups in the slide. However, our model fails to detect the pattern and therefore predicts wrongly in this case. To address this error type, it might need further explicit modeling of global constraints to increase model capability.

Error 2: Missing Semantic Content. Currently our model only utilizes the visual properties of elements (e.g., type, position, font setting). We observe that there are a certain amount of error cases that element content (e.g., images and texts) plays an decisive role. For example in Fig. 6b, our model wrongly recognizes pictures and textboxes as two separate groups, while each textbox describes its



(a) Weak global contextualization.



And finally I will leave you with two videos that summarize my belief in learning …

Being SMART means that you always try your best and never give up!

Everyone is capable of learning math if you belief you can! MATH IS BEAUTIFUL!

(b) Missing semantic content.

Figure 6. Examples of two main error types in our approach.

horizontally-aligned picture and should be grouped as an image-caption pair. If the semantic contents of elements are taken into consideration, the ambiguity would be resolved. Multi-modality with text and vision could be a promising future direction.

5. Conclusion

This paper presents the first end-to-end trainable model for hierarchical visual grouping in graphic design layouts. Since obtaining grouping labels requires a significant amount of annotation effort, we propose a two-stage unsupervised training strategy. In the first stage, our model is pre-trained on heuristic-labeled data using Gestalt laws. In the second stage, we adopt bootstrapping to iteratively refine the noisy labels. Our experimental results demonstrate the effectiveness of our approach, and we provide extensive analysis to explain our model behaviors. Finally, we conclude by identifying two representative error types of our approach for further investigation in future work.

References

[1] Pablo Arbeláez, Jordi Pont-Tuset, Jon Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 328–335, 2014. 2

[2] Galal M. Binmakhashen and Sabri A. Mahmoud. Document layout analysis: A comprehensive survey. *ACM Comput. Surv.*, 52(6), oct 2019. 2

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, page 213–229, Berlin, Heidelberg, 2020. Springer-Verlag. 3

[4] Yu Chen, Wei-Ying Ma, and Hong-Jiang Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, page 225–233, New York, NY, USA, 2003. Association for Computing Machinery. 1, 2

[5] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, USA, 1993. 2

[6] Stefano Ferilli, Marenglen Biba, Floriana Esposito, and Teresa Maria Altomare Basile. A distance-based technique for non-manhattan layout analysis. *2009 10th International Conference on Document Analysis and Recognition*, pages 231–235, 2009. 2

[7] Raghudeep Gadde, Varun Jampani, Martin Kiefel, Daniel Kappler, and Gehler V. Peter. Superpixel convolutional networks using bilateral inceptions. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[8] Yupang Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. Layoutlmv3: Pre-training for document ai with unified text and image masking. *arXiv preprint arXiv:2204.08387*, 2022. 2

[9] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel samping networks. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[10] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 3

[11] Zhaoyun Jiang, Shizhao Sun, Jihua Zhu, Jian-Guang Lou, and Dongmei Zhang. Coarse-to-fine generative modeling for graphic layouts. In *AAAI'22*, February 2022. 2, 4

[12] R. G. Casey K. Y. Wong and F. M. Wahl. Document analysis system. In *IBM Journal of Research and Development*, volume 26, pages 647–656, November 1982. 2

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[14] Janin Koch and Antti Oulasvirta. Computational layout perception using gestalt laws. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, page 1423–1429, New York, NY, USA, 2016. Association for Computing Machinery. 2

[15] Kurt Koffka. *Principles of Gestalt psychology*, volume 44. routledge, 2013. 1

[16] Zhiheng Li, Wenxuan Bao, Jiayang Zheng, and Chenliang Xu. Deep grouping model for unified perceptual parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020. 2

[17] Xiaodan Liang, Liang Lin, Xiaohui Shen, Jiashi Feng, Shuicheng Yan, and P. Eric Xing. Interpretable structure-evolving lstm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018. 2

[18] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph lstm. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[19] Min Lu, Chufeng Wang, Joel Lanir, Nanxuan Zhao, Hanspeter Pfister, Daniel Cohen-Or, and Hui Huang. Exploring visual information flows in infographics. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery. 1, 2

[20] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Learning Layouts for Single-Page Graphic Designs. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1200–1213, 2014. 2

[21] Jordi Pont-Tuset, Pablo Arbeláez, Jonathan T. Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):128–140, 2017. 2

[22] Ren and Malik. Learning a classification model for segmentation. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 10–17 vol.1, 2003. 2

[23] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019. 4

[24] Ruth Rosenholtz, Nathaniel R. Twarog, Nadja Schinkel-Bielefeld, and Martin Wattenberg. An intuitive model of perceptual grouping for hci design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, page 1331–1340, New York, NY, USA, 2009. Association for Computing Machinery. 1, 2, 4

[25] Danqing Shi, Weiwei Cui, Danqing Huang, Haidong Zhang, and Nan Cao. Reverse-engineering information presentations: Recovering hierarchical grouping from layouts of visual elements, 2022. 2, 5

[26] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 2

[27] Hung-Ming Sun. Page segmentation for manhattan and non-manhattan layout documents via selective crla. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 116–120, 2005. 2

[28] Xin Li Xiangcheng Du Zhao Zhou Liang Xue Cheng Jin Tianlong Ma, Xingjiao Wu. Document layout analysis with aesthetic-guided image augmentaiton. *arXiv preprint arXiv:2111.13809*, 2021. 2

[29] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 3

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, 2017. 3

[31] Yuxi Xie, Danqing Huang, Jinpeng Wang, and Chin-Yew Lin. Canvasemb: Learning layout representation with large-scale pre-training for graphic design. In *Proceedings of the 29th ACM International Conference on Multimedia*, page 4100–4108, New York, NY, USA, 2021. Association for Computing Machinery. 2

[32] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18134–18144, June 2022. 2, 3

[33] Zhen Xu and James Miller. Identifying semantic blocks in web pages using gestalt laws of grouping. *World Wide Web*, 19(5):957–978, sep 2016. 2