A. Implementation Details

A.1. Training Details

The table below provides the configurations we used for training. $s$ is the threshold for binarizing the probability map in Section 3.3 and $dis$ is the threshold for point clustering for Algorithm A.2.

<table>
<thead>
<tr>
<th>Net</th>
<th>Training Steps</th>
<th>$s$</th>
<th>Pooling</th>
<th>$dis$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chart Type</td>
<td>50,000</td>
<td>0.5</td>
<td>Corner</td>
<td>-</td>
</tr>
<tr>
<td>Bar</td>
<td>50,000</td>
<td>0.5</td>
<td>Corner</td>
<td>-</td>
</tr>
<tr>
<td>Pie</td>
<td>50,000</td>
<td>0.4</td>
<td>Center</td>
<td>-</td>
</tr>
<tr>
<td>Line</td>
<td>50,000</td>
<td>0.4</td>
<td>Center</td>
<td>0.1</td>
</tr>
<tr>
<td>Query</td>
<td>20,000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A.2. Keypoints of Different Types of Charts

![Figure A.1](image1.png)

(a) Key Points  (b) Bar Objects

Figure A.1: From key points to bar objects. (a) Red dots show the top-left corners and green dots show the bottom-right corners of the bar objects. (b) By matching corresponding top-left and bottom-right points, we can infer the bounding boxes of the bar objects outlined in green.

![Figure A.2](image2.png)

(a) Key Points  (b) Pie Sectors

Figure A.2: From key points to pie sectors. (a) Red dots show the center points and green dots show the arc points on the pie chart. (b) By estimating the pie radius and assigning arc points to the corresponding center points, we can get the outlines of the pie sectors.

A.3. Pie Radius Estimation

The main idea of Algorithm A.1 is to find the optimal radius that can link all center and arc points while having the smallest error. We assume that each arc point associates with only one center point, while one center point can link to multiple arc points. Binary search is utilized to find the optimal radius $r^*$ and error threshold $t$ that satisfy this hypothesis. The error threshold $t$ will then be used to filter out wrong arc point to center point assignments.

**Algorithm A.1 Pie Radius Estimation**

**Require:** center points $\{p_{c}\}$, arc points $\{p_{arc}\}$

**Ensure:** the estimate radius $r^*$ and threshold $t$

1: compute the all possible radius distance list $R = [r_1, \ldots, r_n]$
2: $lt = 0.05, lr = 0.2$
3: while $lr - lt > 0.001$ do
4:     for $r_b \in R$ do
5:         $count = \text{length}(\{r_i^*\})$ where $r_i^* \in R$ and $r_i^* - r_b < lt + lr$
6:         if $count > maxcount$ then
7:             $maxcount = \max(count, maxcount)$
8:             $r^* = r_b, t = lt$
9:         end if
10:     end for
11:     if $maxcount < \text{length}(\{p_{arc}\})$ then
12:         $lt = \frac{lt + lr}{2}$
13:     else
14:         $lr = \frac{lt + lr}{2}$
15:     end if
16: end while

Figure A.3: From key points to line representations. (a) Red dots show the pivot points on the line chart. (b) By connecting points from each separate line, we can get the representations of the lines (in green) on the line chart. The tiny green box means the key point is an intersection point of two or more lines.

A.4. Line Formation from Key Points

Algorithm A.2 describes the hierarchical clustering algorithm that divides all the key points into multiple groups.
based on the lines they belong to. The clustering is performed in the embedding space of key points.

Let \( P \) denote the point set that contains all the key points in the line chart image. In the beginning, we initialize every point as an individual set \( p_i \). If two sets contain a pair of points whose embedding distance is smaller than the threshold, then we will merge the two sets into one. At the end, we will get multiple groups of key points where each group contains the points that belong to the same line.

**Algorithm A.2 Point Clustering**

**Require:** key points \( P = \{p_1, ..., p_N\} \), embeddings of key points \( \{e_1, ..., e_N\} \), threshold \( dis\ min \)

**Ensure:** grouped key points \( \{P_1, ..., P_K\} \)

1: \( P_i = \{p_i\} \)
2: for \( p_i \in \{p_1, ..., p_N\} \) do
3: for \( p_j \in \{p_1, ..., p_N\} \) do
4: if \( |e_i - e_j| < dis\ min \) then
5: Union(\( P_x, P_y \)), where \( p_i \in P_x \) and \( p_j \in P_y \)
6: end if
7: end for
8: end for

Algorithm A.3 describes the procedure that assigns the intersection points to corresponding lines. We use the pre-trained \( \text{QUERY} \) network to predict whether one intersection point \( c_j \) should be inserted to the grouped key point set \( p_k \).

**Algorithm A.3 Adding Intersection Points**

**Require:** intersection points \( \{c_1, ..., c_M\} \), previous grouped sets \( \{P_1, ..., P_K\} \)

**Ensure:** grouped key points

1: for \( P_k \in \{P_1, ..., P_K\} \) do
2: sort \( P_k \) ascendingly according to x axis coordinates
3: for \( p_i \in P_k \) do
4: for \( c_j \in \{c_1, ..., c_M\} \) do
5: if \( \text{QUERY}(p_i, c_j) \) then
6: Union(\( P_k, \{c_j\} \))
7: end if
8: end for
9: end for
10: Repeat previous loop in descending order.
11: end for