A. Implementation Details

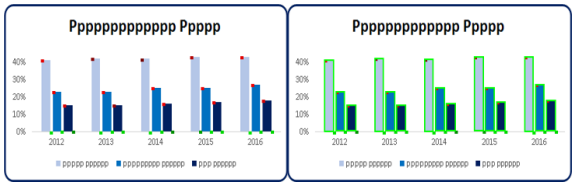
A.1. Training Details

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 1: Training Details.

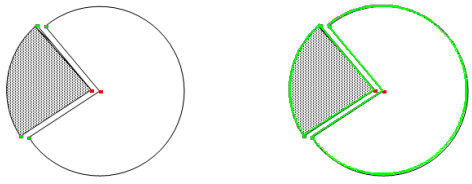
Net	Training Steps	s	Pooling	dis
Chart Type	50,000	0.5	Corner	-
Bar	50,000	0.5	Corner	-
Pie	50,000	0.4	Center	-
Line	50,000	0.4	Center	0.1
Query	20,000	-	-	

A.2. Keypoints of Different Types of Charts



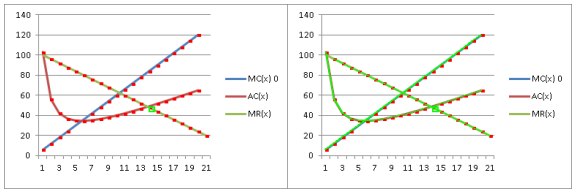
(a) Key Points (b) Bar Objects

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.



(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) Key Points (b) Lines

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.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 $\{p_v\}$, arc points $\{p_{arc}\}$
Ensure: the estimate radius r^* and threshold t

compute the all possible radius distance list $R = [r_1, ..., r_n]$
$lt = 0.05, lr = 0.2$
while $lr - lt > 0.001$ **do**
  for $r_b \in R$ **do**
    $count = \text{length}(\{r_i^*\})$ where $r_i^* \in R$ and $\frac{r_i^* - r_b}{r_b} < \frac{lt+lr}{2}$
    if $count > maxcount$ **then**
      $maxcount = \max(count, maxcount)$
      $r^* = r_b, t = lt$
    end if
  end for
  if $maxcount < \text{length}(\{p_{arc}\})$ **then**
    $lt = \frac{lr+lt}{2}$
  else
    $lr = \frac{lr+lt}{2}$
  end if
end while

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 $\{P_1, ..., P_K\}$

```
P_i = {p_i}
for p_i ∈ {p_1, ..., p_N} do
    for p_j ∈ {p_1, ..., p_N} do
        if |e_i − e_j| < dis min then
            Union(P_x, P_y), where p_i ∈ P_x and p_j ∈ P_y
        end if
    end for
end for
```

Algorithm A.3 describes the procedure that assigns the intersection points to corresponding lines. We use the pretrained 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

```
for P_k ∈ {P_1, ..., P_K} do
    sort P_k ascendingly according to x axis coordinates
    for p_i ∈ P_k do
        for c_j ∈ {c_1, ..., c_M} do
            if QUERY(p_i, c_j) then
                Union(P_k, {c_j})
            end if
        end for
    end for
    Repeat previous loop in descending order.
end for
```