Algorithm 1 Data Range Estimation

Require: Plot Area Info: $Top, Left, Bottom$. OCR results R
Ensure: $Y_{scale}, Y_{max}, Y_{min}$

Find the nearest candidate $r \in R$ as r_{max} to point $(Left, Bottom)$
where $r.r < Left - 4$ and $r.text$ is number
Find the nearest candidate $r \in R$ as r_{min} to point $(Left, Top)$
where $r.r < Left - 4$ and $r.text$ is number
$r_{min}.num$ = number($r_{min}.text$)
$r_{max}.num$ = number($r_{max}.text$)
$Y_{scale} = \frac{r_{max}.num - r_{min}.num}{r_{min}.t - r_{max}.t}$
$Y_{min} = r_{min}.num - Y_{scale}(Bottom - \frac{r_{min}.t + r_{min}.b}{2})$
$Y_{max} = r_{max}.num + Y_{scale}(\frac{r_{max}.t + r_{max}.b}{2} - Top)$

3.3. Type-specific Chart Object Detection

3.3.1 Bar Chart

In Section 3.1, we have extracted the top-left and bottom-right key points from bar images using the key point detection network. In this step, we need to match all the top-left key points to the corresponding bottom-right key points to construct the bar objects. We binarize the key point probability map by threshold value $s = 0.4$. For each top left point p_{tl}, we find the closet bottom right point p_{br} and group them together to obtain the bounding box. The distance measure is defined as a weighted distance on x-axis $\text{dist}_x(.,.)$ and y-axis $\text{dist}_y(.,.)$:

$$\text{dist}(p_{tl}, p_{br}) = \gamma \text{dist}_x(p_{tl}, p_{br}) + \nu \text{dist}_y(p_{tl}, p_{br}) \quad (2)$$

For vertical bar charts, we use $\gamma > \nu$. For horizontal bars $\nu > \gamma$. In the case of vertical bar charts, to find the corresponding bottom-right key point, we only search for the right side of the plot area for each top-left key point.

3.3.2 Pie Chart

To get the location of the pie center and arc points, we use the same key point detection network as decribed in Section 3.1. We replace the corner pooling layer by center pooling layer from [7] to capture the 360-degree neighborhood information. We filter the key points prediction probability map by threshold $s = 0.3$ to get binarized heat map.

For each sector element, the key point detection network extracts the center point p_v and arc point p_{arc}. When grouping the key points to form the sectors, we consider two cases: (1) **tight pie chart** where all the sectors are laying together to form one circle (oval) (2) **exploded pie chart** where one or more sectors are separated from each other. Previous works [6][23] can process the pie charts in the first case but fail to deal with the charts in the second case. In this work, we design a algorithm SECTOR COMBINING 2

Algorithm 2 Sector Combining

Require: center points $\{p_v\}$, arc points $\{p_{arc}\}$
Ensure: sectors defined by edge points $[p_v, p^b_{arc}, p^e_{arc}]$

if length($\{p_v\}$) == 1 **then**
  for $p_{arc} \in \{p_{arc}\}$ **do**
    find the nearest $p^*_{arc} \in \{p_{arc}\}$ in clockwise order
    new sector = $[p_v, p_{arc}, p^*_{arc}]$
  end for
else
  r^*, t = Pie Radius Estimation($\{p_v\}, \{p_{arc}\}$)
  for $p_v \in \{p_v\}$ **do**
    for $p_{arc} \in \{p_{arc}\}$ **do**
      find the nearest $p^*_{arc} \in \{p_{arc}\}$ in clock wise order
      if $\frac{dis(p_v, p_{arc}) - r^*}{r^*} < t$ and $\frac{dis(p_v, p^*_{arc}) - r^*}{r^*} < t$ **then**
        new sector = $[p_v, p_{arc}, p^*_{arc}]$
      end if
    end for
  end for
end if

to find the key points in each sectors for both cases. For the first case, we only need to sort the arc points in clock-wise order and calculate the portion of each sector. For the second case, we include the pie radius estimation step where we find the optimal radius that can link all center and arc points. The center and arc points has 1:N mapping, meaningly, one or more sectors can be attached with one center point. We check if the distance between a center point and the candidate arc points is within some threshold. If yes, then this pair belongs to the same sector, otherwise not. (Details of the pie radius estimation can be found in the supplemental material.)

3.3.3 Line Chart

The key point detection network predicts the locations of pivot points on the line. In order to group the key points according to the lines that they belong to, we attach a convolutional layer in the key point extraction branch (after conv1 in Figure 3) as the embedding layer. We enforce the feature embeddings of points in the same line to be as close as possible, and the embeddings from different lines to be as far as possible. We define the embedding loss function following the practice of [15]:

$$e^k_m = \frac{1}{N} \sum_i e^k_i, \text{where } \{e^k_i\} \text{ belong to a same line } k \quad (3)$$

$$loss_{pull} = \frac{1}{K} \sum_k \frac{1}{N} \sum_i (e^k_i - e^k_m)^2 \quad (4)$$