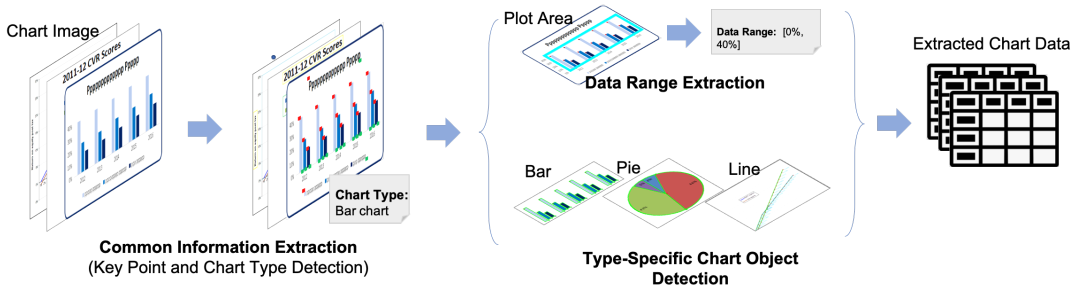

Figure 2: ChartOCR framework outline. There are three major steps: common information extraction, data range extraction and type specific chart object detection. **Common information extraction** aims to detect the keypoints and the chart type. **Data range extraction** infers the range of the data that the chart represents. **Type specific detection** focuses on extracting the chart objects (eg. bars, lines and etc).

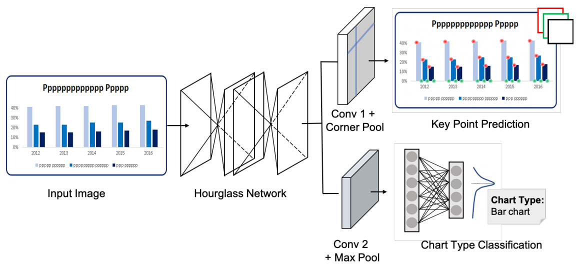

Figure 3: Common information extraction network. We use a hourglass network to provide pixel-level probability map of the keypoint locations. Corner pooling layer is applied on the penultimate layer of keypoint detection branch to increase the receptive field on along the horizontal and vertical direction. (Best viewed in color)

for key point proposal. The output for key point detection network is a probability map that highlights the pixels in key point locations. The probability map has 3 channels to predict the locations of *top-left*, *bottom-right* and *background*. The size of the output probability map is the same as input image. The penultimate layer of key point detection network is a *corner pooling* layer adopted from the CornerNet[15]. Corner pooling layer performs max-pool on the horizontal and vertical direction respectively, which helps the convolutional layers to better localize key point locations. We follow the same setting of CornerNet[15] and define the loss functions as the summation of probability map loss and the smooth L1 loss for keypoint coordinates.

Chart Type Classification We add an additional convolutional layer to the direct output of Hourglass Net and convolve the key point feature map into a smaller size e.g. (32×32). Then we apply max-pooling on it to obtain a one-dim vector. We then feed the intermediate feature vector to fully connected (FC) layers to predict the chart type of the input image. The last FC layer of this branch has `softmax` activation and this branch is trained with cross-entropy loss. Let N be the number of samples in the batch, C be the number of classes. y_{ic} is 1 if and only if ith sample belong to class c, and $\mathbf{p}$ is the prediction of probability distribution. Then the loss can be written as

$$L_{CE}(\mathbf{y}, \mathbf{p}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic} log(p_{ic}) \tag{1}$$

3.2. Data Range Extraction

Data range extraction helps us to convert the detected key points from image pixel space to the numerical readings. The data range extraction applies to line and bar charts. For pie chart, the summation of all the sectors should be 100% by default, thus the data range extraction can be skipped.

We use Microsoft OCR API [1] to extract the text from the image. The extracted text comes from legend, title and axis-labels. For data range extraction, we need to identify the numbers that are associated with y-axis only. To separate out those y-axis labels, we assume that those numbers are always on the left-hand side of the plot area. Thus we only need to locate the plot area, then based on its position, the y-axis labels can be filtered out easily. The plot area is also defined by the top-left and bottom right corners, so we could follow the similar routine as keypoint detection described in 3.1 to locate the plot area. Once we have the plot area location and the OCR result, we design the Data Range Estimation algorithm 1 to get the data range of the chart. In this algorithm, we first use the detected corner points to identify the plot area, then find the recognized numbers that are on the left-hand side of plot area, and finally use the top and the bottom numbers to calculate the data range and pixel range to map the points to the actual data value.

[1]Microsoft OCR API: https://dev.cognitive.azure.cn/docs/services/5adf991815e1060e6355ad44/operations/587f2c6a154055056008f200