Automatic Bar Detection with Decoupled Heatmap-Embedding Learning

Kühne, Constantin and **Postnov, Kirill** and **Schulze Tast, Johann**
and **Treykorn, Felix** and **Zemann, Till**
Hasso-Plattner-Institute, Potsdam, Germany

Abstract

Charts excel in visually communicating diverse data, however chart detection algorithms often struggle to accurately obtain the presented information due to the various styles of the charts. We propose a new decoupled neural network architecture to solve this task. Our model improves upon previous Deep Learning based approaches and achieves new state-of-the-art performance for automatic bar detection on the largest public bar detection benchmark ExcelChart400K.

We predict bounding boxes of bars in bar charts by locating all corner keypoints simultaneously and matching the bottom-right and top-left keypoints via contrastive embeddings learned with our novel similarity loss. Additionally, we propose new metrics for the bar-detection domain for evaluating how well the object detection models perform with later data extraction in mind.

Our code repository is available on GitLab: https://gitlab.hpi.de/ai-in-practice/ai-in-practice-think-cell/.

1 Introduction

Extracting information from visual representations, such as bar charts, is essential for data-driven decision-making. While humans can interpret visual charts effortlessly, machines face significant challenges [21]. Automatic chart analysis, which entails extracting raw numerical data from charts, is pivotal for enabling machines to decode visual charts effectively. Bar detection, a key component of automatic bar chart analysis, is foundational and critically influences all subsequent tasks in the chart analysis process (see Figure 1).

Automatic chart analysis, particularly bar detection, has significant benefits beyond just convenience. It offers substantial time savings, especially in tasks where data analysis plays a pivotal role. Additionally, limitations that are inherent in manual data extraction, including human errors and subjectivity, are eliminated. By automating the process, individuals can quickly extract data from charts without the need for manual annotation, thereby increasing efficiency and accuracy. This improves various tasks in different domains, such as scientific document processing, chart understanding, question-answering (QA) for charts, and automatic (re-)designing of charts [3, 8].

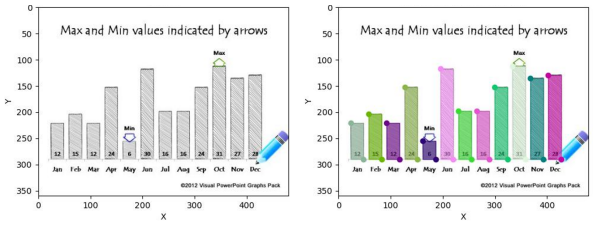

Figure 1: **Bar chart detection example.** *(left)* Original input image; *(right)* Image with predicted bounding boxes.

However, automatic bar detection poses several challenges, including variations in image quality due to factors like poor resolution or background interference, as well as diverse bar chart types (e.g., stacked bar charts) [26, 12]. Traditional rule-based approaches often require extensive domain knowledge and meticulous attention to edge cases, making them unreliable in detecting bars amidst such complexity. Addressing these challenges, therefore, necessitates the development of robust, state-of-the-art object detection or instance segmentation approaches [7].

To be practical for real-world applications, an effective automatic bar recognition solution must fulfill several key requirements. Firstly, it must demonstrate fast execution speed to accommodate the demands of real-time data analysis. Secondly, it should be compatible

with commodity hardware and allow client-side execution to ensure accessibility and scalability. Most importantly, the detection mechanism should achieve near-perfect accuracy, minimizing the need for manual intervention and post-processing.

In this work, we propose a framework for accurate bar detection comprising two separately trained networks; one for keypoint detection and the second for matching two detected keypoints to a combined bounding box. For this, the second network uses contrastive embeddings. We first outline the network architecture and training pipeline before evaluating the framework's performance by comparing it to other state-of-the-art solutions. Furthermore, we introduce two new metrics to measure keypoint detection accuracy.

2 Related Work

2.1 Object Detection

Object detection is a core problem in computer vision. Some early approaches [20, 15] extracted hard features from input images and then passed them through a window to detect objects. However, a more recent approach, as described in [22], frames object detection as a regression problem that detects bounding boxes of objects and their associated classes in a single neural network. This approach has proven to be superior in many ways compared to the earlier methods.

2.2 Object Keypoint Detection

With CornerNet [14] the authors propose an approach to object detection using a bounding box defined by paired keypoints. They employ a single convolutional neural network in form of an hourglass network [18] to detect objects as paired keypoints, eliminating the need for designing a set of anchor points such as the center point and offsets in x- and y-direction. They predict two sets of heatmaps for each top-left and bottom-right corner. One heatmap for defining the location and the other for pairing keypoints that belong to the same bounding box. The associative embedding heatmap aims to properly group the predicted keypoints. This can be done with a single model that simultaneously predict keypoints and embeddings of objects [17].

To improve the detection accuracy, they further introduced a special kind of pooling, which is referred to as corner pooling. This pooling layer aggregates maxima of features horizontally and vertically, as a fundamental building block for localizing corner points.

2.3 Keypoint Detection for Bar Charts

With ChartOCR [16], the authors tried to combine deep learning and rule-based methods to achieve high generalization ability when extracting keypoints from various chart types. This further extended the approach of CornerNet and also uses the corner pooling layers, an hourglass network as well as separate output channels for the keypoints and embeddings. Corner pooling helps with the corner detection of general objects, because one can assume that the keypoints usually lie outside of the object's boundaries. However, for bar charts, this assumption does not hold since the keypoints are located exactly in the corner of the box. As a result, corner pooling does not provide any additional value in bar detection, though the feature aggregation might confuse the network in some cases. Other works rely on transformer architectures to minimize the post-processing error occurring during grouping the predicted keypoints [26].

The problem of locating the keypoints correctly and still keeping some context awareness in the model is one of the biggest challenges of this task. To overcome this issue of combining local information with global context, Yan, Ahmed and Doermann [27] proposed a modified R-CNN architecture to fuse the local field of a CNN with transformer-based positional encodings to obtain positional context features.

2.4 Embedding Loss Functions

Previous work solves the keypoint matching problem by using contrastive keypoint embeddings with losses that pull embeddings of one box close together, and push embeddings of different boxes away from one another. This push-pull loss [16] requires multiple hyperparameters to weight matching and non-matching pairs, as well as a margin hyperparameter that defines how far apart the keypoints should be to result in a loss of zero.

Improving on the push-pull loss, Ahmed, Yan, and Doermann [2] used the multi-similarity loss [25] that encourages similar samples to be closer to one another and dissimilar samples to be further apart. For corner point matching, this means that the embedding vectors of the top left keypoint

and the bottom right keypoint of a bounding box should be similar while they should be dissimilar to the embedding vectors of all other bounding box keypoints.

2.5 Rule-based Chart Detection

Rule-based methods depend on hand-crafted thresholds and features to extract chart data [5]. They usually rely on finding connected components and then deducing boxes from the graphical components. For this, however, assumptions about chart types and alignment have to be made [11]. One prominent example is the software offered by think-cell [1]. It is capable of bar chart data extraction and uses rule-based heuristics for detecting keypoints internally. The exact methods they employ are not publicly available.

3 Methodology

Our approach diverges from previous works [16, 27, 2] by using a novel decoupled architecture comprised of two models, each optimized for a specific task within the prediction pipeline (see Figure 2). The first model is dedicated to predicting the top-left and bottom-right keypoints of bars by generating a keypoint probability heatmap for both keypoint types from the preprocessed image. Then, the predicted heatmaps are postprocessed with simple non-maximum suppression to find the local maxima.

Conversely, the second model only produces contrastive embeddings for each pixel based on the heatmap information. In doing so, it is able to produce high-quality embeddings, especially at the important spatial regions where the keypoints are located. Through the separation each model can specialize its learned features, eliminating an otherwise needed loss weighting hyperparameter to balance the competing objectives through hyperparameter tuning.

The partitioning of tasks within the pipeline enables us to use higher-resolution images for heatmap generation and lower-resolution images for embedding generation. This allows for a higher receptive field while sacrificing a high spatial resolution, which is unnecessary for embedding matching. As a result of changing the input resolutions and employing two independent models, the performance of the entire system substantially improves. To ensure that embeddings corresponding to keypoints of the same bar have high similarities, we use the linear sum assignment algorithm, which maximizes the sum of similarities in the bipartite graph between the two sets of all top-left and all bottom-right keypoints. This enables their matching and is equivalent to the algorithm used in [16].

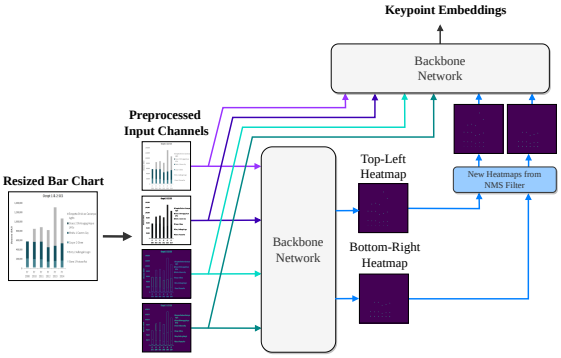

Figure 2: **Schematic illustration of the bar detection network training pipeline.** The inputs of the heatmap network are 8 preprocessed channels (see Figure 3), and the embedding network receives 10 channels as input, consisting of the 8 preprocessed channels and two additional heatmap channels, which are the post-processed prediction of the first network at inference time, and the target heatmap created from keypoint labels at training time.

3.1 Heatmap Loss

To train the heatmap network, we use a modified version of the focal loss (1) that was introduced with CornerNet [14]

$$\mathcal{L}_{heat} = -\frac{1}{M} \sum_{c,i,j}^{C,H,W} \begin{cases} \overline{p_{cij}}^{\gamma} \log(p_{cij}) & \text{if } y_{cij} = 1 \\ (1 - y_{cij})^{\beta} (p_{cij})^{\gamma} \log(\overline{p_{cij}}) & \text{otherwise} \end{cases} \tag{1}$$

where N is the number of target boxes, $M = \max(1, N)$, y_{cij} is the target probability, p_{cij} is the predicted probability and $\overline{p_{cij}} = (1 - p_{cij})$. The exponent γ of the modulating factor can be tuned to put a higher emphasis on difficult, mispredicted instances (e.g. tiny or stylized boxes). The second hyperparameter β scales the loss at the non-maximum regions of the Gaussian target probabilities. Adjusting this parameter affects the variance of the predicted distribution per keypoint, and a high beta leads to a lower loss when $y_{cij} < 1$, resulting in smaller predicted regions with non-zero probability. This is advantageous for further post-processing with non-maximum suppression as the location of the keypoint's maximum probability is constrained to a smaller region and thus more

likely correctly located. It also helps to disentangle overlapping distributions for two keypoints that are close together. A notable case is $\beta = 0, \gamma = 0$, where the function collapses to the standard cross-entropy loss. Our tuning results come to the same conclusion as CornerNet [14] that $\beta = 4$ and $\gamma = 2$ are a good combination for the two hyperparameters. Lastly, we add the $\max(1, N)$ term to avoid division by 0 when the image does not contain any bars.

3.2 Embedding loss

We propose a novel hyperparameter-free loss for learning contrastive embeddings per pixel that can be used to match predicted keypoints.

Let $\mathbf{E}_{tl}$ and $\mathbf{E}_{br}$ represent the vectors of all top-left and bottom-right embeddings for the predicted keypoints in an image, and let $\mathbf{P}, \mathbf{N} \subseteq$ **KP-Pairs** be the sets of positive and negative keypoint pairs, where a positive pair refers to a tuple of two keypoints that belong to the same box. We first concatenate the embeddings into a combined vector $\mathbf{E} = [\mathbf{E}_{tl}; \mathbf{E}_{br}]$. Next, we create the similarity matrix $\mathbf{S} = \mathbf{E}\mathbf{E}^\top$. We use dot-product similarities instead of angular cosine similarities because the cosine similarities are in the range $S_{ij} \in [0, 1]$, which does not match the domain ($\mathbb{R}$) of the softmax function. Continuing, we set every element on the main diagonal and upper-right triangle of $\mathbf{S}$ to zero, so that every unordered pair of keypoints is present exactly once. By zeroing out the main diagonal, we also exclude self-similarities. Subsequently, we extract the normalized similarities by first computing the softmax over rows and columns independently, and then taking the element-wise average of these two softmax matrices. This lets each keypoint construct a distribution over all other keypoints with their probability of matching. Combining both row-wise and column-wise softmax probabilities lets us treat each keypoint within the context of being a 'reference' and a 'target' keypoint respectively, and without conflating all similarities into a single probability distribution.

$$\mathbf{S}_{\text{norm}}[i,j] = \frac{1}{2}\left(\frac{e^{s_{ij}}}{\sum_{k=1}^{|\mathbf{E}|} e^{s_{kj}}} + \frac{e^{s_{ij}}}{\sum_{k=1}^{|\mathbf{E}|} e^{s_{ij}}}\right) \tag{2}$$

Using the normalized similarity-matrix, we formulate the embedding loss as the mean cross-entropy loss, so that positive keypoint pairs $(i,j) \in \mathbf{P}$ are pushed towards a probability of one and negative pairs $(i,j) \in \mathbf{N}$ are pushed towards a probability of zero. We have to include the negative pairs in the loss because averaging the two softmax distributions does not guarantee each row and column to be a valid probability simplex.

$$\mathcal{L}_{\text{embed}} = -\left(\frac{1}{|\mathbf{P}|}\sum_{(i,j)\in\mathbf{P}} \log(\mathbf{S}_{\text{norm}}[i,j]) + \frac{1}{|\mathbf{N}|}\sum_{(i,j)\in\mathbf{N}} \log(1 - \mathbf{S}_{\text{norm}}[i,j])\right) \tag{3}$$

3.3 Backbone models

During our experiments, we evaluated several backbone models on the heatmap and embedding tasks. These include the standard U-Net [23], an Attention U-Net (AttnU-Net) [19], a Recurrent Residual U-Net (R2U-Net) [4], a Recurrent Residual Attention U-Net (R2AttU-Net) that combines the previous two architectures, the R2AttU-Net with dilated convolutions and a U-shaped Swin Transformer (Swin U-Net) [6], which is a variation of the Vision Transformer [9] that uses shifted window attention and U-Net style skip connections between the downsampling and upsampling path.

We hypothesize that solving the heatmap task only requires small receptive fields, so any U-Net variant should work well here. However, for embedding predictions, models with extended receptive fields, such as the R2AttU-Net with dilated convolutions and Swin U-Net theoretically offer an advantage by learning globally coherent embeddings that result in correct keypoint matchings. Models with smaller receptive fields might therefore fail to produce high-quality embeddings. For example, a top-left corner of a stacked bar might not be able to computationally take into account which bottom-right corner of the stacked bars below is the correct one that should produce a matching embedding.

3.4 Preprocessing

By integrating additional informative channels as part of our preprocessing, we observe a slight positive impact on the model's performance. The preprocessed input consists of the normalized image, a contrast-enhanced image via histogram equalization, and edge detection results using the Sobel and Canny filters (see Figure 3). First, we normalize the image by subtracting the mean and dividing it by the standard deviation, channel-wise. Next, we

apply histogram equalization on the original image to improve the visibility of features by spreading out the most frequent intensity values. This is particularly useful when the original image has low contrast and boxes have similar colors as the background. Finally, the Sobel and Canny edge detectors are used to extract box boundaries. The simple Sobel filter is useful for emphasizing the most significant edges, while the Canny edge detector can capture a wider range of edges and is robust against noise. The inclusion of edge detection filters has been shown to increase the performance of CNN-based models for classification tasks [13, 24]. It also slightly improved the heatmap segmentation and embedding predictions during our experiments.

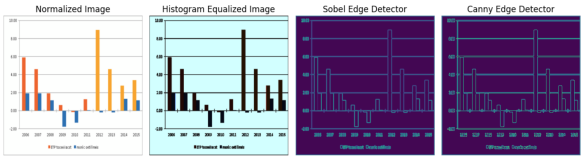

Figure 3: **Preprocessed input channels:** Normalized image *(3 channels)*, histogram-equalized image *(3 channels)*, Sobel edge detection result *(1 channel)* and Canny edge detection result *(1 channel)*.

3.5 Augmentations and other Regularization Measures

We employ various standard regularization measures to prevent overfitting, including batch normalization, an L2 weight decay of 3e-6, and random image augmentations (see Figure 11 for examples). Our suite of random augmentations consists of random crops where no bounding box is cropped out of the image, vertical/horizontal flips, adjustments in hue, saturation, value, brightness and contrast. Additionally, it encompasses Gaussian blur, Gaussian noise, random gamma adjustments, JPEG image compression, and RGB value shifts. These augmentations simulate a wide range of real-world conditions to improve the model's robustness and generalization ability. Additionally, we train with small batch sizes of 8, which also contributes a small regularization effect.
During our experiments, we tracked the validation loss on 6,121 validation images to ensure that our model generalizes well. Notably, out of all regularization techniques, the random augmentations improved the model's generalization ability the most, even though the training dataset (with 158,140 images) is already relatively large.

3.6 Training Setup

We train the heatmap and embedding network individually for 20 epochs using a base learning rate of 0.001 and an exponential learning rate decay factor of 0.9 per epoch. Both networks are trained via the Adam optimizer, and the embedding network training additionally employs the Sharpness-Aware Minimization (SAM) [10] optimization method in conjunction with Adam. The inclusion of SAM leads to better generalization capabilities but only works for training the embeddings as the heatmap loss landscape is likely too sharp for this method to find a good solution.
To train our models and tune hyperparameters, we use A40 and A100 GPUs with 40GB and 80GB of memory respectively. When training on consumer GPUs with 24GB of memory, we recommend halving the batch size.
Furthermore, to ease the replication of our results, we include a comprehensive list of all hyperparameter settings in Table 2.

4 Evaluation

4.1 Dataset

We train, tune, and evaluate our models on the bar charts "bardata(1031)" of the publicly available ExcelChart400K dataset. This benchmark contains images of vertical bar charts split into a train (158,140 images), validation (6,121 images) and test (6,262 images) set. The annotated bounding boxes are given in the COCO format (x1, y1, width, height), where a bar is defined via its top-left corner (x1,y1) and its bottom-right corner (x1+width, y1+height).

The width and height of images follow a log-normal distribution since both are constrained to only positive values (see Figure 4). The training set has a mean height of 415 with a standard deviation of 168 and a mean width of 735 with a standard deviation of 361.
To use an image in our models, they need to be of uniform shape. Taking into account the size distribution, we compromised on a size of 448x448 pixels. We opted to stretch images rather than employing resizing with padding to maintain the aspect ratio. This decision stems from observed instances where resizing with padding resulted in small objects within the images being hard to detect, thus potentially compromising model performance.

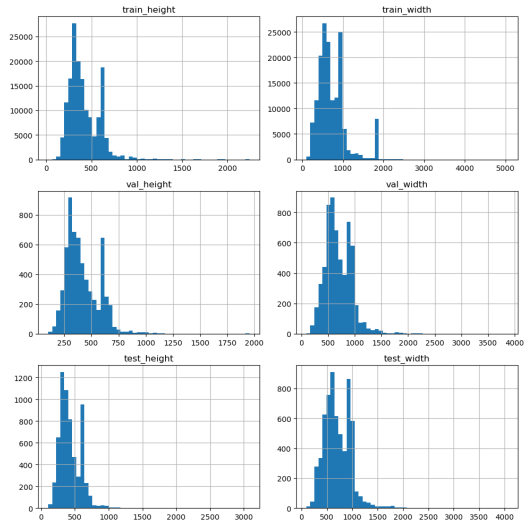

Figure 4: **Image height and width frequency distributions** of the ExcelChart400K dataset (grouped by train, validation and test splits).

4.2 Metrics

We use six different metrics to evaluate all models, including task-specific ones from previous works to ensure comparability and our own metrics that suit our use cases. Each metric is defined in detail below.

"Perfect Images" and "Perfect Boxes": We introduce two new metrics - "Perfect Images" and "Perfect Boxes" - that specifically focus on precise and near-perfect prediction of keypoints with the task of data extraction as a later step in mind. These metrics are more concerned with the error of the y coordinate of a keypoint than the x coordinate, as the y errors are more important for the correct extraction of data points of each (vertical) bar.

To be considered "perfect," a predicted bounding box must have a maximum deviation from the ground truth of 20 pixels for both top-right and bottom-left keypoints' x coordinate and a maximum deviation of 2 pixels for their y coordinate respectively. The "Perfect Images" metric builds on the "Perfect Boxes" metric by counting the images that do not have any incorrect bounding boxes under the criteria of the "Perfect Boxes" metric, i.e., all bounding boxes in an image must be "perfect" to be counted as a "Perfect Image."

To account for different image sizes, we additionally scale the maximum allowed deviations for x and y values, so they are within 2px and 20px before resizing the image (Section 4.1).

Intersection over Union (IoU): IoU calculates the extent of overlap between predicted bounding boxes and the actual (ground truth) bounding boxes. It is defined as the intersection area of the predicted bounding box and its corresponding ground truth divided by their area of union:

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where A and B denote the two bounding boxes. To account for multiple boxes in an image, we first calculate the IoU for each bounding box pair and then take their mean. A perfect overlap would result in an IoU score of 1.0.

ChartOCR-Score: In [16], the authors propose three evaluation metrics for various chart types. We adopt their bar chart metric to evaluate our method and ensure comparability to previous work.

Too Many/Few Predicted Boxes: To assess the confidence of a model, we also report the percentage of predicted bounding boxes when the number of predicted boxes does not match the number of target boxes. Let $\mathrm{N}_{\mathrm{preds}}, \mathrm{N}_{\mathrm{gt}}$ be the numbers of predicted and ground truth bounding boxes respectively. We then calculate the scores for N images as

$$too\_many = \frac{1}{N} \sum_{i}^{N} \max(0, (\mathrm{N}_{\mathrm{preds}}[i]/\mathrm{N}_{\mathrm{gt}}[i]) - 1)$$

and

$$too\_few = \frac{1}{N} \sum_{i}^{N} |\min(0, (\mathrm{N}_{\mathrm{preds}}[i]/\mathrm{N}_{\mathrm{gt}}[i]) - 1)|.$$

4.3 Experimental Setup

For our experiments, we evaluated our proposed architecture by comparing it with three baseline methods. The first baseline method, think-cell [1], is a rule-based off-the-shelf commercial product capable of bar detection and chart data extraction. The second method, referred to as CACHED [27], employs a framework that combines the Cascade R-CNN with SwinT-FPN. Lastly, ChartOCR [16] employs a single Stacked Hourglass-Net as the backbone network to detect and match keypoints in various chart types. Evaluation of these methods was performed using the test set derived from the ExcelChart400K dataset, using our predefined metrics. We independently reproduced the baseline results (with the exception of think-cell) to mitigate any inconsistencies.

4.4 Results

4.4.1 Quantitative Results

As shown in Table 1, our proposed model outperforms all other methods in terms of accuracy. Notably, with 80.25% the portion of perfectly predicted images is significantly higher than all other compared baselines. This means that not only does our method recognize most overall keypoints correctly (with a perfect boxes score of 96.83%), but the wrong predictions also center around a few difficult images, while the majority of bar charts are recognized without any error.

The closest competitor is ChartOCR, which slightly outperforms our approach in the IoU metric with a score of 0.90 where the difference is less than 0.001. However, ChartOCR falls behind in every other benchmark, including the ChartOCR-score, which was first introduced in their paper.

The rule-based model currently used by think-cell is only competitive in the "Too many Predicted Boxes" metric, although this comes at a significant cost of missed boxes, with a score of 13.49% being the highest among all baselines. Nevertheless, it performs well in the ChartOCR score, suggesting that once keypoints are predicted, they are predicted correctly.

CACHED lags behind in all measured metrics except for the IoU, which is almost constant for every approach at around 0.89. However, it should be noted that this metric only measures the overlap of bounding boxes and does not meaningfully assess how well a model recognizes keypoints, which is more important for data extraction tasks.

4.4.2 Qualitative Evaluation

We visualize the prediction results from our method for qualitative evaluation (see Figure 8 and Figure 9). We randomly selected images from both correctly and incorrectly predicted charts. Our method has demonstrated an ability to precisely locate each bar element in a variety of images, even in charts that have a large number of targets to identify. However, Figure 9 shows that our method sometimes struggles with challenging backgrounds or interfering elements, as well as with bars that are below a certain size.

4.5 Discussion

The results demonstrate that the proposed approach has successfully learned to detect bounding boxes in a wide range of scenarios and does so in a fast fashion at 1.09 seconds per image on commodity hardware without a GPU. It is currently the most reliable model for this task in various complex cases. The framework is capable of extracting the bounding boxes of images with diverse backgrounds, a large number of bars, stacked bars, and images prone to human error. Moreover, it can also detect bars that are partially occluded (see Figure 8).

It's important to note that the ChartOCR metric proposed in [16] does not take into account whether the model predicted too few or too many boxes. This is one of the reasons we introduced the new "Perfect Images" metric. Therefore, when comparing the ChartOCR-score, one should also consider the "Too Many/Few Predicted Boxes" as the current metric doesn't accurately describe the problem of perfectly predicting the bounding boxes of an entire image.

Despite our theoretical assumptions that models

Table 1: Quantitative evaluation and comparison to related work. Metrics are computed on the ExcelChart400K test split, which consists of 6,262 images. All metrics are described in Section 4.2. Furthermore, not listed in the table due to the code not being publically available for evaluation, the authors of ChartDETR [26] report a ChartOCR-Score of 0.904 with a ResNet-50 backbone and 0.923 with a HRNet-w32 backbone.

Metrics	**Methods**			
	think-cell (Rule-Based)	**CACHED** (Cascade R-CNN + SwinT-FPN)	**ChartOCR** (Stacked Hourglass-Net)	**Ours** (2x decoupled R2AttU-Net)
Perfect Images (%) ↑	63.5580	38.5340	67.3427	**80.2460**
Perfect Boxes (%) ↑	83.7186	83.5549	94.5368	**96.8290**
ChartOCR-Score ↑	0.8913	0.8175	**0.9224**	**0.9309**
IoU ↑	0.8878	0.8917	**0.9030**	**0.9023**
Too Many Predicted Boxes (%) ↓	1.6585	7.4910	1.1920	**0.9223**
Too Few Predicted Boxes (%) ↓	13.4960	8.5970	3.3951	**1.5745**

with a global receptive field yield better results, we did not observe an empirically significant improvement. This could be due to insufficient training data for a Vision Transformer-based model like the Swin U-Net or an already large enough receptive field of the other models. Ultimately, the R2AttU-Net, which combines recurrent convolutions that result in a larger receptive field, and a local attention mechanism to filter information propagation, showed superior performance for both learning the heatmaps and embeddings. Other U-Net variants were only a few percentage points off, with the R2AttU-Net with dilated convolutions performing on par for the embedding task.

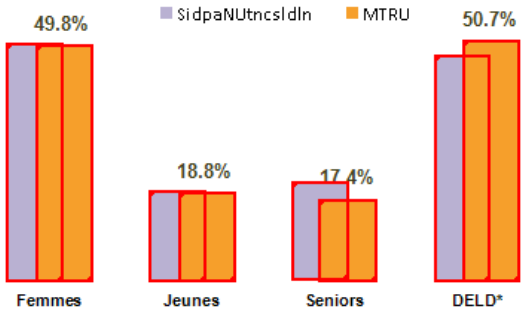

Figure 5: **Example with occluded bottom-right corners.** Our model correctly infers the width of the purple bars to predict the occluded keypoints.

Our model has learned to perform simple reasoning, making it able to infer the proportions of a bar based on information from surrounding bars. For example, when groups of bars usually consist of two bars next to each other, the model can correctly predict both bars, even when one of them is almost entirely occluded. Furthermore, the model can infer the most likely position of an occluded corner keypoint based on the size and placement patterns (e.g. bars are always placed in groups of two) of other bars. It can also take the style of bars into account, for example, in an image bars might always be the same width. Thus for an occluded bar, the model would use the typical width and height of the bars to predict the occluded keypoints (see Figure 5).

5 Limitations and Future Work

It should be noted that, despite the proposed approach being effective, there are some limitations and potential areas for improvement. One of these limitations is that if a large image contains very small bars, the predictions of the bounding boxes for these bars are more prone to errors. This is because the image is rescaled to 448 x 448 pixels, which also rescales its bounding boxes. Consequently, the size of very small boxes is reduced to a point that is currently undetectable by the model. To address this, future work might experiment with larger image input sizes to decrease the effects of this limitation.

The model sometimes fails to detect objects in obscure images with objects that rarely occur in front of bars. This commonly happens when there are lines in images that cut obliquely through bars or the whole image (as seen in the bottom right image of Figure 9). In such cases, the bounding boxes created by the model may use the intersection of these lines and the bounding boxes as keypoints, rather than the actual corner points of the bars. However, this issue does not occur for regular horizontal lines. We suggest that this issue might be addressed by including more examples of these error cases in the dataset. Moreover, utilizing a much larger dataset could also aid in making Vision Transformer models viable and potentially superior to our current U-Net-based backbone model.

The ExcelChart400K dataset comprises solely images with vertical bars. This inherently means our models are specifically trained for this type of bar chart. Even though we do use random flipping in our augmentations which could enable the models to learn how to predict images with horizontal bars, it requires further examination to confirm if this is the case.

It could be beneficial to extend the current architecture by taking into account the legends of images. These legends often contain valuable and crucial information, such as the number of bars or the color of those bars in the image. Such information can be utilized by the model to determine the required number of bars or specific colors to look for. Extracting this information is a challenging task that would require extensive effort to yield significant improvements. Additionally, the current training methodology for the embeddings involves a loss function that is built upon the cross-entropy loss. While this is a good starting point, it can be improved by using a more sophisticated loss function, such as the focal loss used in heatmap training. This would emphasize the challenging keypoint pairs, likely leading to better outcomes in difficult situations, as already demonstrated by the heatmap loss.

The next steps could include generalizing the methodology to other chart types such as pie charts or line charts, as demonstrated in [16, 26]. This would require the use of the resilient keypoint and embedding architectures mentioned but would necessitate additional data sets and experimentation, as well as a fresh approach to determining which data points should be extracted and which keypoints should be matched.

6 Business Opportunity

Identifying bars in charts can present a range of obstacles, including the varying types of bar charts, diverse backgrounds, and resolutions, as well as overlapping bars or text. Nonetheless, our solution has demonstrated encouraging outcomes and holds the potential for commercialization. With this in mind, we suggest a plan for advancing our model into a valuable product and explore monetization strategies.

6.1 Product and Value Proposition

The current state of our product offers a console-based end-to-end pipeline that takes a raw image file as input and accurately detects all bounding box coordinates. In the next phase, we plan to expand the prototype to include the detection of basic chart elements outside the plot area and integrate OCR. The resulting product would thereby be capable of automatically extracting numerical information from charts without requiring any human intervention (customer gain). This solution solves various customer pains, as presented in Figure 6. For instance, it increases efficiency and reduces opportunity costs by providing quick execution and near-perfect bar detection. Furthermore, we can address even more customer jobs such as data mining, automatic redesigning of charts, or document processing.

6.2 Target Market

Our primary aim is to target companies that invest a significant amount of time in creating presentations, showcasing their work, and visualizing data. This market has the potential to be enormous and includes various industries such as consultancies, marketing agencies, or news agencies. We are looking to expand our reach by forming alliances with educational institutions. By doing so, we can provide educational licenses to students, which would be free of charge. Although this may not lead to the biggest revenue stream, it would bind the students

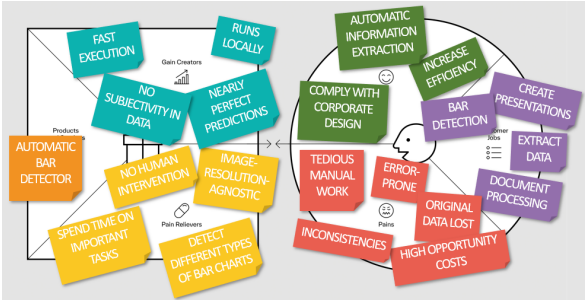

Figure 6: **Value Proposition Canvas** showcasing how our solution addresses the customers' jobs and needs.

as potential future customers. We also believe that there is a significant opportunity in offering our product to individual customers. The market for individual consumers is potentially as large as the entire user base of tools like PowerPoint.

Our Go-to-Market strategy is centered around think-cell, a company that offers tools for creating and editing aesthetic charts and diagrams. We aim to expand our customer base by leveraging the size and reach of think-cell's existing clientele. By integrating our product into their suite, we expect to jump-start our growth.

6.3 Business Model

The success of our venture relies heavily on our product's superior performance compared to other state-of-the-art solutions. Therefore, our experienced ML engineers, who optimize and develop our models, are our most valuable resource. These engineers will also account for the largest portion of our cost structure, along with office space and computing resources. To cover these costs, we have devised the following pricing strategy:

- We sell end-user licenses to companies. As these companies scale and more employees utilize our product, their demand for additional licenses will increase. Ultimately, as the companies we serve grow, so will our own business. As an incentive for more end-users with our product in each company, the price per license will decrease depending on the amount of end users.

- We offer a pay-per-use model. There is no minimum upfront fee and we charge users for each API call made. It is meant for individual users who have variable utilization of our product.

It will be essential for us to collaborate with major cloud service providers to have access to computing and cloud resources. In the future, we

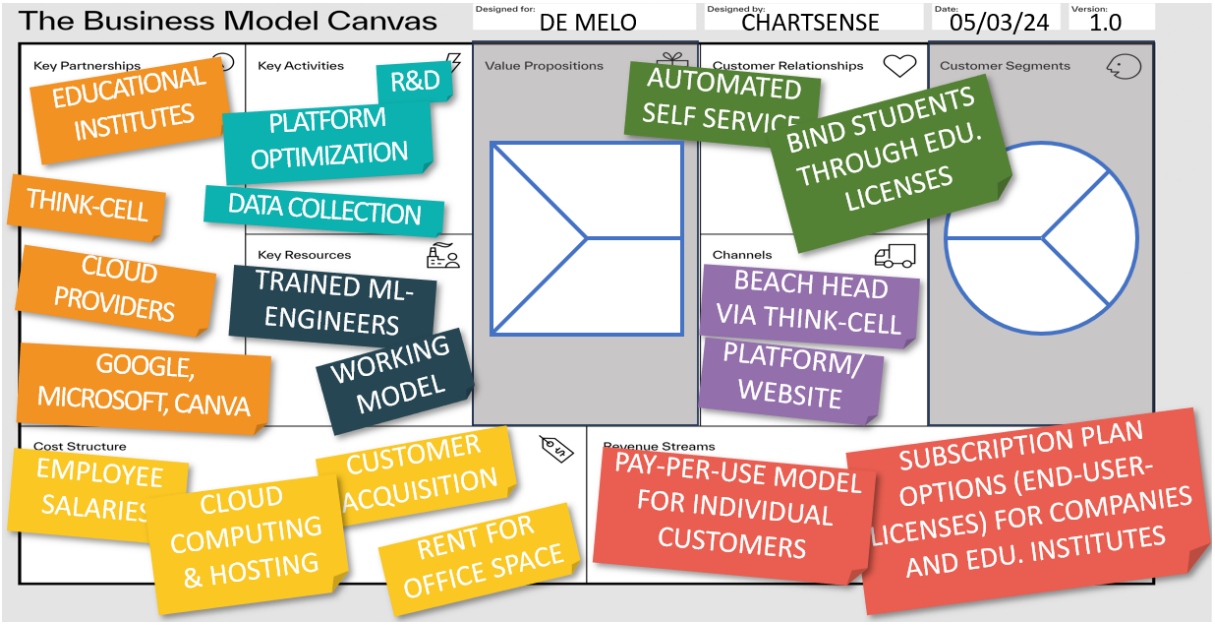

Figure 7: **Business Model Canvas** showcasing our strategic planing and developments.

can expand our business by partnering with companies that offer the leading software for creating presentations, such as Microsoft and Google.

The complete Business Model canvas is provided in Figure 7.

7 Conclusion

In this report, we propose a novel decoupled pipeline for precise bar detection. We combine a Recurrent Residual Attention U-Net with preprocessing and additionally augment the training images in conjunction with other regularization techniques to boost generalization competence. Furthermore, we introduce new metrics that focus on customer satisfaction to evaluate the accuracy of predictions. Our approach has been tested against the current best models and shows superior or matching performance in all metrics. Specifically, the number of images where all bars are predicted perfectly is significantly higher than the current state-of-the-art.

Our framework demonstrates strong prediction capabilities by accurately detecting bars in various challenging images. It also displays simple reasoning abilities. However, the current version of the model is mainly trained and thus limited to vertical bar charts and struggles with tiny bars that suffer from rescaling. These limitations could be addressed in future work.

Overall, our solution has great potential to be turned into a valuable product, and we have identified a promising monetization strategy and business model for it.

References

[1] How to work with think-cell's internal datasheet | think-cell. Accessed: 04.03.2024.

[2] Saleem Ahmed, Pengyu Yan, David S. Doermann, Srirangaraj Setlur, and Venugopal Govindaraju. 2023. Spaden : Sparse and dense keypoint estimation for real-world chart understanding. In *IEEE International Conference on Document Analysis and Recognition*.

[3] Rabah A. Al-Zaidy, Sagnik Ray Choudhury, and C. Lee Giles. 2016. Automatic summary generation for scientific data charts. In *AAAI Workshop: Scholarly Big Data*.

[4] Md Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, Tarek M. Taha, and Vijayan K. Asari. 2018. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation.

[5] Abhijit Balaji, Thuvaarakkesh Ramanathan, and Venkateshwarlu Sonathi. 2018. Chart-text: A fully automated chart image descriptor. *CoRR*, abs/1812.10636.

[6] Hu Cao, Yueyue Wang, Joy Chen, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, and Manning Wang. 2021. Swin-unet: Unet-like pure transformer for medical image segmentation.

[7] Zhi-Qi Cheng, Qi Dai, Siyao Li, Jingdong Sun, Teruko Mitamura, and Alexander G. Hauptmann.

2023. Chartreader: A unified framework for chart derendering and comprehension without heuristic rules.

[8] Komal Dadhich, Siri Daggubati, and Jaya Sreevalsan-Nair. 2021. Barchartanalyzer: Digitizing images of bar charts. pages 17–28.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale.

[10] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-aware minimization for efficiently improving generalization.

[11] Jinglun Gao, Yin Zhou, and Kenneth E. Barner. 2012. View: Visual information extraction widget for improving chart images accessibility. In *2012 19th IEEE International Conference on Image Processing*, pages 2865–2868.

[12] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. 2017. Chartsense: Interactive data extraction from chart images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 6706–6717, New York, NY, USA. Association for Computing Machinery.

[13] Anchal Kalra and Roshan Lal. 2016. A hybrid approach using sobel and canny operator for digital image edge detection. pages 305–310.

[14] Hei Law and Jia Deng. 2018. Cornernet: Detecting objects as paired keypoints. *CoRR*, abs/1808.01244.

[15] D.G. Lowe. 1999. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2.

[16] Junyu Luo, Zekun Li, Jinpeng Wang, and Chin-Yew Lin. 2021. Chartocr: Data extraction from charts images via a deep hybrid framework. *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1916–1924.

[17] Alejandro Newell and Jia Deng. 2016. Associative embedding: End-to-end learning for joint detection and grouping. *CoRR*, abs/1611.05424.

[18] Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937.

[19] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. 2018. Attention u-net: Learning where to look for the pancreas.

[20] C.P. Papageorgiou, Michael Oren, and Tomaso Poggio. 1998. General framework for object detection. volume 6:, pages 555 – 562.

[21] Jorge Poco and Jeffrey Heer. 2017. Reverse-engineering visualizations: Recovering visual encodings from chart images. *Computer Graphics Forum (Proc. EuroVis)*.

[22] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2015. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation.

[24] Cesar Torres, Claudia I. Gonzalez, and Gabriela E. Martinez. 2022. Fuzzy edge-detection as a preprocessing layer in deep neural networks for guitar classification. *Sensors*, 22(15).

[25] Xun Wang, Xintong Han, Weiling Huang, Dengke Dong, and Matthew R. Scott. 2019. Multi-similarity loss with general pair weighting for deep metric learning. *CoRR*, abs/1904.06627.

[26] Wenyuan Xue, Dapeng Chen, Baosheng Yu, Yifei Chen, Sai Zhou, and Wei Peng. 2023. Chartdetr: A multi-shape detection network for visual chart recognition.

[27] Pengyu Yan, Saleem Ahmed, and David Doermann. 2023. *Context-Aware Chart Element Detection*, page 218–233. Springer Nature Switzerland.

A Appendix

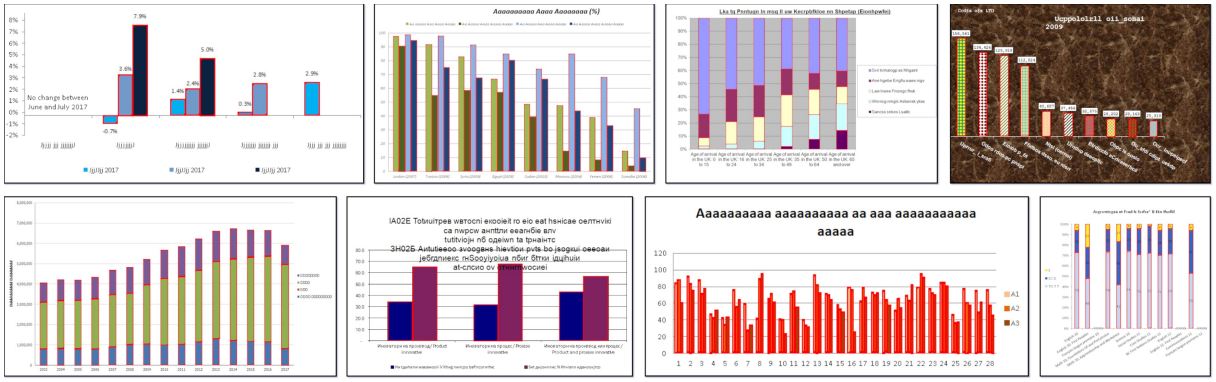

Figure 8: **Examples of bar chart detection results.**

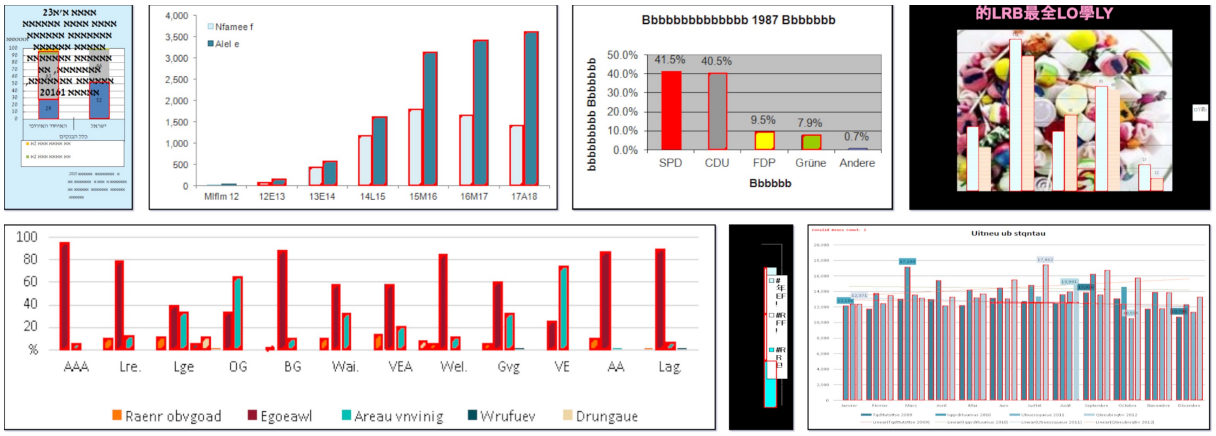

Figure 9: **Failure cases of bar chart detection results.**

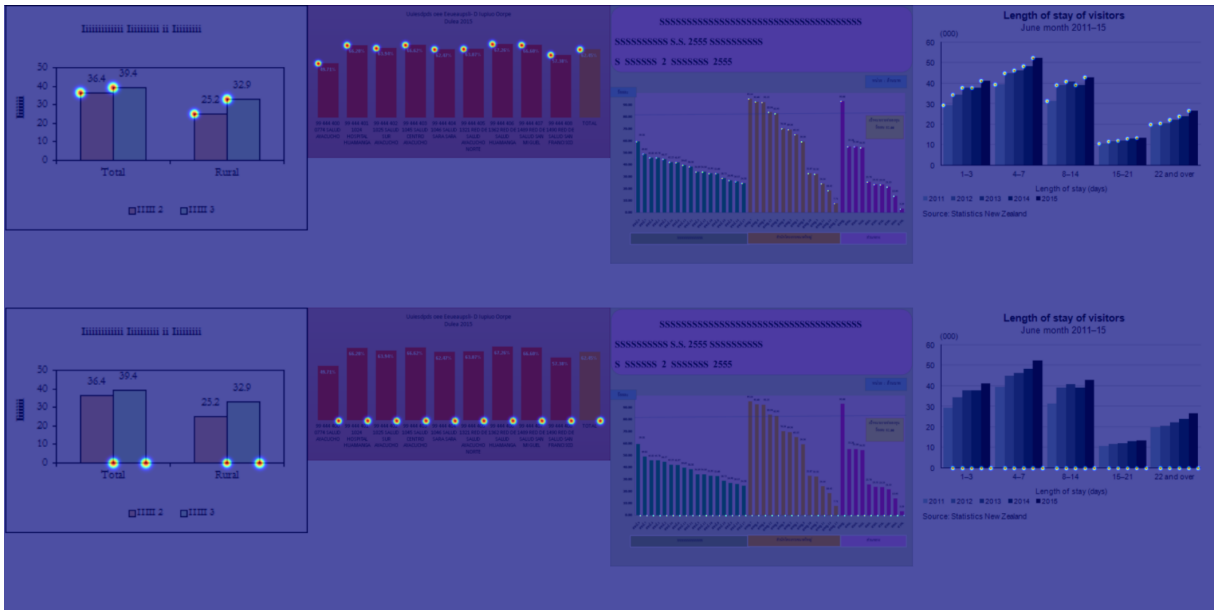

Figure 10: **Examples of predicted keypoint probability heatmaps** (overlayed on the input images) using the R2AttU-Net.

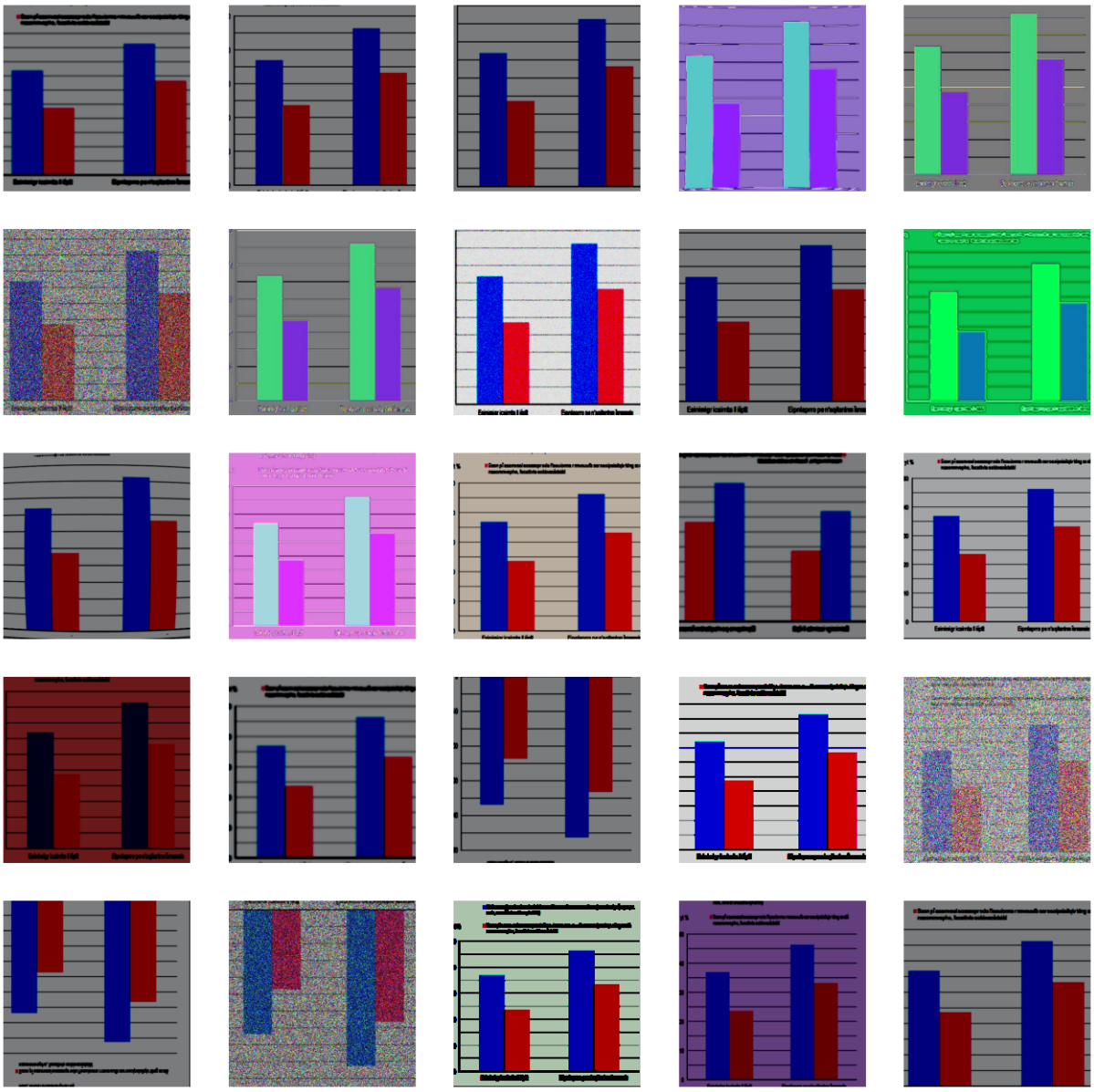

Figure 11: **Examples of random augmentations** applied on a single image.

Table 2: **Comprehensive List of Hyperparameter Values with Descriptions.**

Hyperparameter	**Description**	**Value**
Model	Recurrent Residual U-Net with attention (combines [4, 19])	R2AttU-Net
Rescaled image size	Size of input images and preprocessed channels	Heatmap-Net: (448x448) Embedding-Net: (224x224)
Number of Channels	Number of channels in the U-Net downsampling path The upsampling path follows the same list in reverse	$[32, 64, 128, 256, 512]$
Recurrent steps per convolution	Number of times a recurrent convolution is applied	3
Max. learning rate	Base learning rate for the optimizer	0.001
Exponential learning rate decay	Exponential decay rate of the learning rate scheduler	0.9
Epochs	Number of training epochs	20
Weight decay	Weight decay (L2 penalty) for regularization	3e-6
Batch size	Number of images per training batch	8
Optimizer	Sharpness-Aware Minimization [10] is used in conjunction with the Adam optimizer, which leads to better generalization at the cost of doubling the training time	Adam (+ SAM for embeddings)
Focal loss beta	Beta parameter for the focal loss, where $\beta > 0$ reduces the loss at non-maximum regions of the soft Gaussian targets	4
Focal loss gamma	Gamma parameter for the focal loss, where $\gamma > 0$ puts higher focus on hard, misclassified examples	2
NMS threshold	Threshold for non-maximum suppression to extract keypoints from the predicted probability heatmap	0.4
Embedding size	Number of dimensions of the embedding space	32
Gradient Clipping	Gradient norm clipping to prevent exploding gradients	Clip at L2-Norm=1