Text Localization in Scientific Figures using Fully Convolutional Neural Networks on Limited Training Data

Morten Jessen
stu85924@mail.uni-kiel.de
Kiel University, Germany
Kiel, Schleswig-Holstein, Germany

Falk Böschen
stu212915@mail.uni-kiel.de
Kiel University, Germany
Kiel, Schleswig-Holstein, Germany

Ansgar Scherp
ansgar.scherp@essex.ac.uk
University of Essex, UK
Colchester, UK

ABSTRACT

Text extraction from scientific figures has been addressed in the past by different unsupervised approaches due to the limited amount of training data. Motivated by the recent advances in Deep Learning, we propose a two-step neural-network-based pipeline to localize and extract text using Fully Convolutional Networks. We improve the localization of the text bounding boxes by applying a novel combination of a Residual Network with the Region Proposal Network based on Faster R-CNN. The predicted bounding boxes are further pre-processed and used as input to the of-the-shelf optical character recognition engine Tesseract 4.0. We evaluate our improved text localization method on five different datasets of scientific figures and compare it with the best unsupervised pipeline. Since only limited training data is available, we further experiment with different data augmentation techniques for increasing the size of the training datasets and demonstrate their positive impact. We use Average Precision and F1 measure to assess the text localization results. In addition, we apply Gestalt Pattern Matching and Levenshtein Distance for evaluating the quality of the recognized text. Our extensive experiments show that our new pipeline based on neural networks outperforms the best unsupervised approach by a large margin of 19-20%.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; • **Applied computing → Optical character recognition**.

KEYWORDS

neural networks, training data augmentation, OCR

ACM Reference Format:
Morten Jessen, Falk Böschen, and Ansgar Scherp. 2019. Text Localization in Scientific Figures using Fully Convolutional Neural Networks on Limited Training Data. In *ACM Symposium on Document Engineering 2019 (DocEng '19), September23–26, 2019, Berlin, Germany.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3342558.3345396

1 INTRODUCTION

Scientific figures are the visual representation of information or data from scientific experiments. They are created with the intention of providing a quick and clear way to provide contextualized knowledge to the reader. Graphical elements and textual elements are used to illustrate complicated concepts or complex relationships to make them easier to understand. It is known that the information encoded in scientific figures are often not repeated directly in the text [3]. This makes extracting and contextualizing information from scientific figures even more important and renders into a mandatory requirement for fully automated analyses of scientific literature.

In this paper, we focus on the task of automated localization and extraction of text from scientific figures by proposing a supervised method based on Deep Learning. The special difficulties associated with the task of localizing and extracting the text are to distinct between graphical elements like axes, markings inside a graph, etc., and text with different fonts, sizes, emphases, resolution, and orientation [2]. In the past, several supervised approaches [27, 33] and unsupervised approaches [2, 6, 40, 47] have been proposed to extract text from scientific figures and similar images. Unsupervised approaches have been very successful in the past [2] since there are only very few annotated scientific figures available for training supervised models. The lack of training data is one of the main hindrances when using supervised approaches like neural networks [33]. On the contrary, neural networks lately demonstrated to provide very effective solutions for image analysis tasks. Particularly, Fully Convolutional Networks (FCNs) are very good at solving image processing tasks and have surpassed other machine learning algorithms in many—if not most—current problems like semantic segmentation [4]. Nonetheless, the recent successes of FCNs in object detection tasks raise the question of whether or not they will perform similarly well for text localization when only limited training data is available.

To address this question, we choose a Residual Network [22] due to its recent successes for image processing tasks [4, 21]. Residual Networks are characterized by skip connections, i. e., layers of the network have short-cuts to other layers by jumping over some layers. This allows to jointly learn from different levels of abstraction of the image such as directly on convolutions over the input image as well as candidate regions for text elements proposed by intermediate levels. As basis, we use the Faster R-CNN [37] architecture but adopt its Region Proposal Network (RPN) with improvements made by He et al. [20]. We test different versions of the classic ResNet to determine the optimal architecture for our problem. We also address the question of how to deal with the lack of training data. The five datasets of scientific figures used in this paper contain only 633 images in total, which is low for training a complex neural network. We adopt multiple approaches to solve this problem in an attempt to find the optimal solution for text extraction from our datasets. First, we pre-train our neural network on one of the available larger natural images datasets, although these datsets contain fundamentally different images of, e. g., street scenes with a few text elements. Second, we compare the combination of all five datasets of scientific images with using each individual dataset. Lastly, we apply different data augmentation methods to increase our training set without adding additional new data. Especially the data augmentation approach shows very promising results. To complete the text extraction, we use the Long Short Term Memory (LSTM)-based optical character recognition (OCR) Engine Tesseract 4.0[1] to extract text from the bounding box predictions for text recognition.

The remainder of this paper is structured as follows: In Section 2, we discuss the related work with respect to text extraction from natural images and scientific figures. We provide a brief overview of our approach in Section 3. In Section 4, we describe the experimental setup that we used for evaluating our approach and answering our research questions. The experiments and their results are presented in Section 5 and discussed in Section 6, before we conclude.

2 RELATED WORK

First, we review recent top-performing general purpose object detection algorithms. Subsequently, we narrow the focus on neural networks that extract text from natural images. Finally, we briefly address the very few works that have applied neural networks on scientific figures and define the baseline for our work.

General Purpose Object Detection Algorithms. In terms of general purpose object detection algorithms, Dai et al. [9] presented the so called R-FCN, which is a region-based fully convolutional network. In contrast to Fast R-CNN and Faster R-CNN, this network shares almost all computation on the entire image – instead of repeating calculations in per-region subnetworks. The authors introduce position-sensitive score maps to avoid a conflict between translation-invariance in image classification and translation variance in object detection. Lin et al. [31] developed RetinaNet. In contrast to R-CNN, RetinaNet is a one-stage detector. RetinaNet uses a new Focal Loss that focuses training on a small set of hard input images and prevents the easy negatives from overwhelming the detector. This is done to avoid a foreground-background class imbalance during training that usually makes one-stage solutions inferior to two-stage solutions. Another interesting development are the You Only Look Once (YOLO) architectures [35, 36] by Redmon et al. The initial YOLO [35] approach repurposes classifiers to perform detection instead. Object detection is reformulated into a single regression problem that uses the input pixels to determine the bounding boxes' coordinates and the classification. YOLO is, like R-FCN, a one-step approach that is optimized for speed. The network can process images in real-time at 45 frames per second, which can be increased to 155 frames by using a variant called Fast YOLO. Because of this speed optimization, YOLO is prone to making localization errors, but, according to the authors, avoids false positives and outperforms methods like R-CNN in natural images and artworks. The second version YOLOv2 and YOLO9000 [36] improved these concepts further. YOLO9000 can detect over $9,000$ object classes and YOLOv2 offers easy trade-offs concerning speed and performance, both of which it can be fine-tuned for. Additionally, the authors propose a method to train the network for object detection and classification at the same time, which allows YOLO9000 to predict detections for object classes that do not have labeled detection data. The main focus of all YOLO architectures is the real-time application. Similarly, the Single Shot MultiBox Detector (SSD) [32] proposed by Liu et al. is a real-time optimized, one-stage object detector as well. The network separates the output space into multiple default bounding boxes. During prediction time, the SSD generates scores for each default bounding box and adjusts the bounding box to better match the detected object shape.

Huang et al. further explore the concept of interconnecting layers with their Densely Connected Convolutional Neural Networks by connecting all layers with each other [23]. The input of a layer consists of the output of *all* previous layers of the network. According to the authors, this new design tackles the vanishing gradient problem and increases the chance of feature reuse, which makes the architecture more efficient. Finally, Veit et al. [45] propose a similar idea to Stochastic Depth [24], but instead of dropping layers during training, they dropped layers of the trained network. Interestingly, the experiments show that dropping layers of a trained ResNet keeps the performance of the whole network similar, but dropping layers of a trained VGG neural network results in a dramatic loss of performance. This shows one of the interesting properties of ResNet, namely, that is has multiple independent paths. The identity paths create multiple paths through the network and the majority of these paths remains intact when removing single layers.

Text Extraction from Natural Images. In the past, different works have applied neural networks for text extraction from natural images such as street scenes and city scenes. The major difference to scientific figures is that the natural images typically only contain very little text such as a street sign or licence plate and at the same time there are significantly larger data sets available for training the neural models. Zhang et al. [54] applied FCNs with text line hypotheses to extract text from natural images. First, the FCN is used to predict map regions in the picture. These regions are then used to create text line hypotheses. Finally, they use an additional FCN classifier to predict the centre of the individual characters in order to remove the false hypotheses. The approach produced state-of-the-art performances on several text detection benchmarks, including ICDAR2015 and ICDAR2013. Borisyuk et al. [1] used a state-of-the-art Faster-RCNN object detection network to localize text in natural images. Their neural network replaces the common convolutional body of ResNet [22] with a ShuffleNet-based architecture [53]. Huang et al. [24] designed a network to tackle the problem that, because of its depth, ResNet needs a long time to train. To this end, they created Stochastic Depth, a method that randomly drops layers during training. This method is very similar to Dropout [42], which is a method used to avoid overfitting, but instead of dropping parts of a layer, Stochastic Depth drops the whole layer. The authors show in their experiments, that combining ResNet with Stochastic Depth improves the results compared to

[1] https://github.com/tesseract-ocr/tesseract

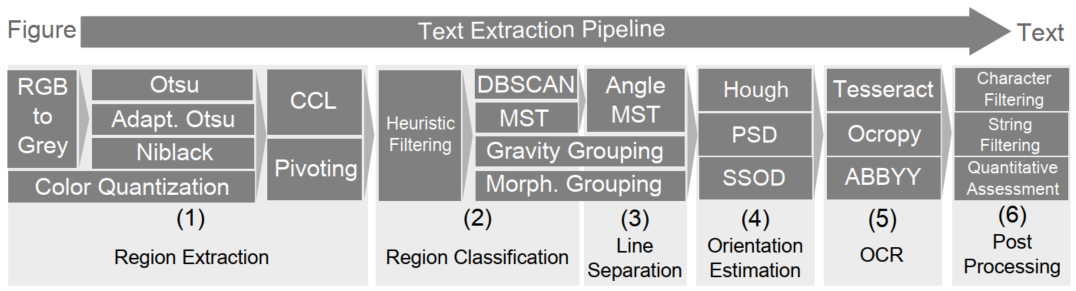

Figure 1: An overview of the components defined in the generic text extraction pipeline of Böschen et al. [2].

the classic ResNet approach and decreases training time. Yan et al. [48] presented a text extraction approach from natural images specifically designed for minority languages. They used the Uyghur language that is spoken in the Xinjiang Uyghur Autonomous Region of Western China by 10-25 million people. They propose an effective Uyghur language text detection system for images with complex backgrounds.

Text Extraction from Scientific Figures. In a recent study, Böschen et al. have compared 32 different unsupervised and supervised approaches for text extraction from scientific figures [2]. Based on a modular design of an adaptive text extraction pipeline, the existing approaches have been mapped into a common framework. To this end, the authors have identified the most common components used for text extraction such as connected component labeling [10, 39], different variants of density-based and agglomerative clustering [7, 15, 16], and the use of a existing off-the-shelf optical character recognition engine for finally extracting the text from the figures. A detailed depiction of the components identified and used in this pipeline are depicted in Figure 1. Their best performing configuration is the BS-4OS-O pipeline, which uses the Single String Orientation Detection (SSOD) method to estimate the orientation of the extracted text elements.

Poco and Heer [33] propose a pipeline for data extraction from scientific figures for the purpose of reverse engineering the original data from the scientific papers. For the identification of text pixels and their discrimination from non-text pixels, the authors rely on the convolutional neural network framework Darknet [34]. The actual identification and extraction of the textual elements is then again conducted using a minimum spanning tree and an of-the-shelf OCR engine as it has been extensively investigated in the survey by Böschen et al. [2]. The results of TencentAiLab, the winner of the ICDAR 2017 DeTEXT challenge [49], suggest that it uses a modern neural network architecture like Faster R-CNN for text extraction from scientific figures. However, no paper has been published by the date of submitting this paper, which makes a comparison impossible.

Summary. Given the discussions above, we use the BS-4OS-O pipeline of Böschen et al. as baseline in this paper. Furthermore, despite the lack of a paper on TencentAiLab, we compare our results also with this approach by using the AP50 measure that is used in the DeTEXT challenge. Overall, we contribute to the extensive survey conducted by Böschen et al. by adding a new text extraction pipeline that bases on the best performing neural networks discussed above. Regarding the backbones for our new text extraction pipelines, we chose ResNet [19] and the ResNeXt variant since they are currently the most effective and popular architectures. We apply our new text extraction pipelines on all datasets previously used by Böschen et al. and add a fifth dataset.

3 TEXT EXTRACTION USING RESNETS

Our approach for text extraction from scientific figures relies on two steps: First, a neural network is used for improved text localization, i. e., better predicting the bounding boxes of the text. Second, we apply an of-the-shelf OCR engine like Tesseract for text recognition from the bounding boxes detected in the first step. Thus, the focus of this paper is to improve the first step of localizing the text elements using neural networks versus the existing supervised and unsupervised approaches.

In this section, we introduce the general concepts of the residual neural networks (ResNets) that we use and explain their role in the context of text extraction. Figure 2 shows the architecture of our approach. Our text localization uses a modified Faster R-CNN architecture [38] to predict bounding boxes using the Region Proposal Network (RPN) and classifies the boundary box predictions (BBP) into those which contain text and those which do not. Subsequently, the text recognition step with the chosen OCR engine Tesseract is briefly described.

3.1 Improved Text Localization Methods

Our text localization method is based on the Faster R-CNN [38] architecture, which improves the original Fast R-CNN [13] by replacing the region proposal bottleneck caused by methods like Selective Search [43] and Edge Boxes [55]. Like Ren et al. [38], we use a Region Proposal Network (RPN) in combination with a Feature Pyramid Network. Computing the region proposals with a deep network has the advantages of being able to reuse parts of the object detection network and take advantage of GPUs [38]. We adopt the proposed Region Proposal Network but use some of the improvements made by He et al. [21] for Mask R-CNN [21]. This approach has previously been used by Chen [5] and Jaderberg [26] for extracting text from natural images. However, it has yet not been adapted for text extraction from scientific figures. Jaderberg also used shared convolutional layers to allow their network to localize text and classify characters within the same network architecture. Tests by He et al. [22] have shown that this method works

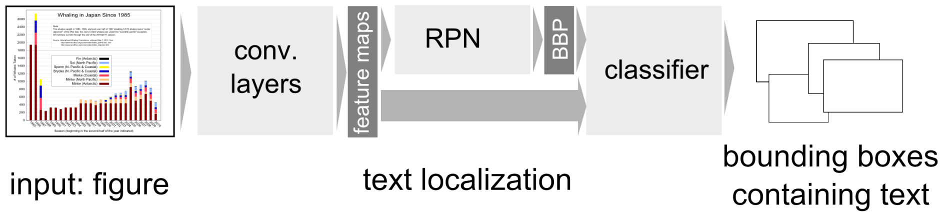

Figure 2: Architecture of the Faster R-CNN Object Detection Network (FRCNN) with the Region Proposal Network (RPN) integrated between the convolutional layers and the classifier. Both, the classifier and the RPN use the CNN's output feature map. We replace the original RPN that prepares the bounding box predictions (BBP) with improvements made by He et al. [21].

and that ResNets can gain accuracy by increasing the depth of the network up to 1, 000 layers. One of the very convenient features of the ResNet is that it consists of four different building blocks, which allows to easily construct networks of varying depth by simply stacking building blocks of the same shape on top of each other.

As second method, we use ResNexT [46], a new version of ResNet. The novelty of this design is the addition of aggregated residual transformations to the original architecture. Xie et al. [46] present an architecture that uses the strategy of repeating layers on which VGG [41] and ResNet have been built and use it to implement a split-transform-aggregate strategy. For ResNeXt, the ResNet path with width 64 is split into 32 paths with width 4. All paths contain the same topology. This adds an additional way, next to depth and width, to change the general architecture of the network without adding much complexity. Xie et al. [46] show that this approach can help to decrease the validation error and generally improves the results. We also replaced RoIPool [13] in the Faster R-CNN architecture with RoIAlign [21] as it was proposed for Mask R-CNN [21]. Mask R-CNN [21] is an improvement on the Faster R-CNN architecture and replaces the Zeiler and Fergus model [52] and Simonyan and Zisserman model [41], which are used in Faster R-CNN with a Residual Network (ResNet) [22]. RoIPool quantizes a floating-number Region of Interest (RoI) to a discrete number area of the feature map and then subdivides the area into spatial bins – quantizing again. RoIAlign avoids harsh quantization by using bilinear interpolation to compute the exact values of the input features at four regularly sampled locations in each RoI bin and aggregates the result (using maximum or average pooling) [21].

3.2 Text Recognition

Like in the survey by Böschen et al. [2], we employ an of-the-shelf component for performing the actual text extraction task. We use the OCR engine Tesseract 4.0, which is pre-trained for the English language, to extract the text from the bounding boxes that we detect in the scientific figures. We apply a multi-step text recognition process that stops as soon as the confidence in the prediction as provided by the OCR engine Tesseract is above 95%. We start with the originally predicted bounding box with an added white border and then, if our confidence limit is not met, continue with the pre-processed and modified images as input for Tesseract. We resize the image, use thresholding mixed with Gaussian blur to improve text segmentation in the image and rotate the image. While thresholding and resizing are only used to improve extraction results, rotating the image is necessary for our pipeline, because our text localization network does not detect the text's rotation. The rotation is split into multiple steps: First, we use only multiples of 90°and halve the steps until we reach 15°steps. Finally, if after all these steps the confidence is still $< 90\%$, we repeat the process, but set Tesseract from its standard mode to single-character mode. We do this because our experiments showed that most of the previously unreadable text bounding boxes contained single characters, which cannot be read in Tesseract's standard mode.

4 EXPERIMENTAL SETUP

We present our experimental setup which includes a detailed description of the datasets and the pre-processing applied on the datasets. Furthermore, this section explains the experimental procedure and evaluation measures to assess the text extraction results.

4.1 Datasets

Scientific Figures Datasets. We use five datasets of scientific figures for our evaluations. The first four datasets have also been used in the extensive comparisons of text extraction methods by Böschen et al. [2]. Thus, they allow us to conduct a fair comparison to the new methods based on neural networks proposed in this paper.

The first dataset is *CHIME-R*, a set of 115 figures from the Internet as well as figures scanned from paper. The gold standard was created by Yang Li [50]. Related to this dataset is *CHIME-S*, a set of 85 synthetically generated figures. The gold standard was created by Zhao Jiuzhou [28]. As third datset, we use *DeGruyter* with 120 figures from academic books provided by DeGruyter. The *EconBiz* dataset contains 121 scientific figures, which were randomly extracted from a corpus of 288, 000 open access publications in the economics domain. The gold standard for the DeGruyter and EconBiz datasets were created by volunteers in our research group. In addition, the fifth dataset is the *DeTEXT* dataset [51]. It contains 192 biomedical figures and is part of the ICDAR Robust Reading competition [49]. Example images for all five datasets are depicted in Figure 3.

Please note, while all datasets contain scientific figures, their nature and characteristics are rather different. CHIME-R and CHIME-S contain only three types of charts, namely bar, pie, and line charts. The scientific figures in the other three datasets DeGruyter, EconBiz, and DeTEXT are more complex and versatile. In addition to bar, pie, and line chars, they also contain scatter plots, flow/process charts,

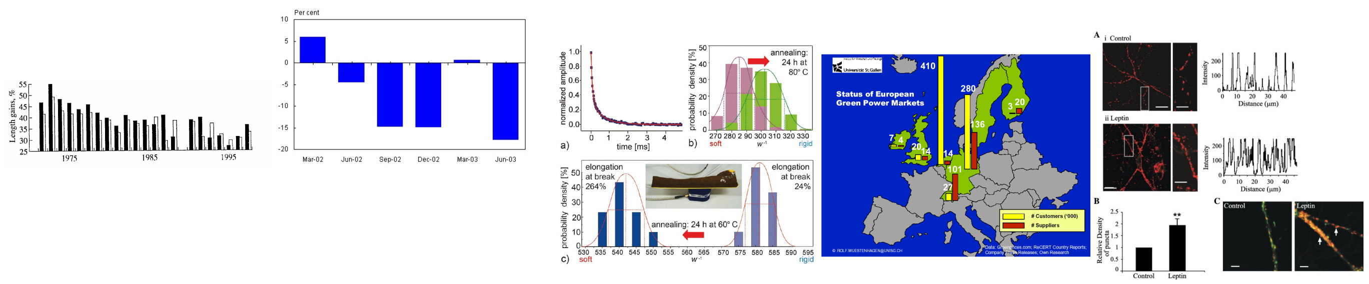

Figure 3: Example scientific figures from CHIME-R, CHIME-S, DeGruyter, EconBiz, and DeTEXT (left to right).

as well as histograms. They also contain figures with multiple types of charts. Furtermore, EconBiz contains map-based figures and DeTEXT includes many medical figures (real-life and abstracted diagrams) and map-based figures as well. Finally, DeGruyter, EconBiz, and DeTEXT also combine multiple figures in one larger figure, where each subfigure again may use different chart types.

Natural Images with Text Elements Datasets. The datasets of scientific figures are rather small for training neural networks. Thus, we additionally experiment with two larger datasets of natural images with some text elements in the captured scenes for pre-training the networks (cf. the discussion in Section 2). Pre-training neural networks on larger datasets is a very common method to remedy the lack of available training data. The idea behind it is to prepare the network to detect general image structures, which are then fine-tuned with the training on the actual datasets.

The first dataset used for pre-training is *COCO-Text* [44]. It is based on the MS-COCO Dataset [30], which is a general object detection dataset and consists of more than 200, 000 labeled natural images with more than 1, 500, 000 individual object annotations in 80 object categories. We used version 1.3 of COCO Text which uses the MS-COCO images but adds bounding box annotations for text occurrences in these images. The annotations consist of 145, 000 text annotations for 63, 686 images, which result in an average of 2.28 text annotations per image. We only used a smaller subset of those images which contain text that is in English, machine written, and legible (given the annotations). This reduces the dataset to 17, 237 images, which we split into 13, 966 training images and 3, 271 test images. The training images have in total 63, 545 text annotations while the test images have in total 14, 644 text annotations.

As second dataset for pre-training, we use the *Total-Text-Dataset* [8], which is designed for text recognition tasks. The Total-Text-Dataset contains natural images like MS-COCO and specifically focuses on text extraction. It contains 1, 555 images with an average of 7.37 annotations per image, compared to 2.73 for COCO-Text. In contrast to COCO-Text, the annotated text in the Total-Text-Dataset can have multiple orientations and is sometimes curved.

4.2 Pre-Processing

Two of the largest challenges of text localization, as illustrated in Figure 4, are to not classify graphical elements like circles or triangles as text elements and to detect text that is using a white font on a colorful background (cf. [2]). Regarding the first challenge, there are no explicit methods to detect geometric figures in order to filter them out. Thus, the idea is that the neural network is able to learn to separate such "noise" in form of graphical elements from the actual text. We apply two pre-processing heuristics on the scientific figures to prepare the images as input for the neural networks. First, we convert the training and testing images to grayscale, which is the most common pre-processing step for text extraction [2]. Based on findings from Kanan and Conttrell [29], we calculate the gray value by taking the mean of the color channels. Second, for each grayscaled image, we create a new, color-reversed image and add it to the training set. The second step is particularly addressing the issue of extracting text in white font.

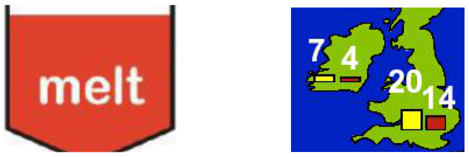

Figure 4: Examples of text elements that are challenging to localize due to the background color (left) and overlap with graphical elements (right).

4.3 Procedure

For the optimization of our text localization methods, we select from a random distribution two images per gradient descent step. All images are resized to their maximum possible size, which is restricted by GPU memory. When scaling to $x \in \mathbb{N}$ pixels, we resize the shortest side of the image to be x pixel long. We keep the aspect ratio of the image and cap the longest side at $\sim 2 \cdot x$ pixel. Thus, the shortest side may actually be shorter than x pixel. Please note, that the GPU memory restrictions are different for the different neural network model. Thus, the maximal input image resolutions differ as well. We sample 512 Regions of Interest (RoI) per image with a ratio of 1:3 (positive to negative) [14]. A RoI is positive, when it contains text and negative when it contains only background.

To ensure a fair test environment, we use the same training length and learning rate for all neural networks during all our experiments. For our text localization network, we chose a learning rate of 0.0075 and used 512 RoI proposals. We trained our network over 250, 000 iterations, which is equivalent to 158 epochs with respect to the fully augmented dataset.

4.4 Measurements

Precision, Recall, and F1-Measure. We are evaluating our text localization methods with the same measures used in Böschen et al. [2]. Thus, we compute precision, recall, and F1-measure for all bounding box matches with an Intersection-over-Union (IoU) of 0.1 or more. Precision calculates the proportion of true positives to all positives, i. e., precision denotes which percentage of the predictions are correct. Recall compares the true positives to all relevant elements, i. e., true positives and false negatives, and gives a percentage of how many of the ground truth bounding boxes have been found. F1-measure is as usual the harmonic combination of Precision and Recall to assign a single measure to a method.

Average Precision. In addition, we decided to use the Average Precision (AP) metric for evaluating our text localization method since it has been used in similar object detection tasks and challenges [12, 49]. We use AP, AP50, and AP75. AP50 counts a detected instance as correct if it has at least 50% IoU with the ground truth mask it has been matched with [11]. AP75 does the same, but with 75%. AP is a summary metric that combines ten equally spaced IoU thresholds from 50% to 95% (0.5, 0.55, 0.60,...,0.90,0.95). Figure 5 depicts some examples for IoU values of 0.5, 0.7, 0.9. While an IoU of 0.5 is sufficient in most cases to capture the text, an IoU of ≥ 0.7 ensures that the whole text is captured.

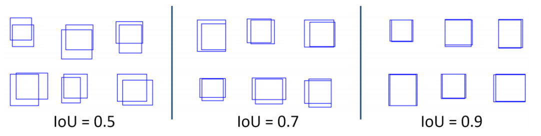

Figure 5: Examples for different Intersection over Union (IoU) values. Figure taken from [55].

Levenshtein Distance. Furthermore, we use Levenshtein distance to evaluate the single-character-edit-distance between the extracted text and the gold standard, with single-character-edits being either insertions, deletions, or replacements. Two variants have been used in Böschen et al. [2], a so-called local Levenshtein distance (LSD) and a so-called global Levenshtein distance (LSG). The LSD calculates the distance between the text string of two matching bounding boxes (predicted and gold standard) and then averages over all matches in a figure. The LSG combines all predicted text from all predicted bounding boxes into one string and sorts the characters in the string alphabetically. The same method is applied on the gold standard and the Levenshtein distance is computed between these two strings.

Gestalt Pattern Matching. Since the Levenshtein distance only provides the minimal edit distance from one string to another, it does not necessarily represent a subjective impression of similarity. Thus, besides the Levenshtein distance, we use the Gestalt Pattern Matching (GPM) to evaluate our extraction results. GPM measures the correctness of the word in relation to the total length of the prediction and gold standard, which gives a better insight into how correct a prediction is. It finds the longest continuous substring that is contained in both strings and repeats this process recursively to the left and right of the matching string until the full string is processed. As a rule of thumb, GPM values ≥ 0.6 mean that the strings are close matches.

5 RESULTS

We run a series of four experiments. First, we compare different architectures of our neural networks and the influence of choosing different hyperparameter values. Second, we conduct pretraining of the networks with natural images and apply dataset augmentation techniques to address the issue of the limited amount of training data. Third, we assess to which extent the best performing neural-network-based text extraction pipelines generalizes regarding train/testing over different datasets. This provides further insights into the pipeline's robustness as well as the impact certain characteristics of scientific figures from the different datasets have. Finally, the best configuration of our pipeline is compared with the best configuration from Böschen et al. [2].

5.1 Architectures and Hyperparameters

We use He et al.'s ResNet [22] as our base model and experiment with different network sizes. The most common configurations are 50 and 101 layers. While it is possible to create ResNets with as low as 18 or as high as 1, 000 layers, both seem to be unfitting for our specific problem. Decreasing the network's depth to less than 50 layers provides only a marginal performance boost, while our very limited training data seems to be insufficient to effectively train a network with more than 100 layers. In addition, we use the advanced and newer ResNeXt [46] architecture with the same depth configurations. For all networks, we use a cardinality of 32 and 64 and a bottleneck width of 8d and 4d, respectively, following Xie et al. [46].

Please note, we individually tailored the maximum possible input image resolutions for the different network architectures so that the images would still fit in our GPU memory (we used a Nvidia Titan-X 12 GB). To accommodate this, we choose to test the network variants with the same resolution restrictions and with their particular maximum resolution. While the test with restricted resolutions might not be crucial for our experiments, it might be interesting to see the performance differences between the networks with the same resolution restrictions.

The results of the networks with the input images restricted to 600 pixels are shown in Table 1. The best results are produced by the original ResNet50/101 variants. With lower image resolutions, the 50 layer version slightly outperforms ResNet101 in the more difficult AP and AP75 metrics, while ResNet101 is slightly better when using AP50. The ResNeXt based architecture performed worse in all experiments.

The results for networks with unrestricted image resolutions, i. e., maximal possible resolution on our GPU can be found in Table 2. Overall, the classic ResNet variants are still providing the best results. But ResNet101 leads in the harder AP and AP75 metrics while ResNet50 delivered the best AP50 results. The most important improvement is in the AP75 metric, where ResNet101 gained 6% and has proven itself as the superior network architecture for this task. It is also noteworthy that ResNet101 has to be restricted to 700-pixel images, while ResNet50's input can be increased to 800 pixel. However, ResNet101 does gain more accuracy by increasing the resolution by 100 pixel than ResNet50 by an 200 pixel increase.

	AP	AP50	AP75
ResNet50	**52.49%**	89.81%	**54.11%**
ResNet101	51.46%	**90.19%**	52.58%
ResNeXt50 32x8d	49.21%	88.11%	48.02%
ResNeXt50 64x4d	49.42%	87.86%	49.16%
ResNeXt101 32x8d	49.70%	88.60%	50.41%
ResNeXt101 64x4d	50.73%	88.84%	51.05%

Table 1: Performance of ResNet and ResNeXt on input images fixed to be 600px wide. Tested on CHIME-R, CHIME-S, DeGruyter, and EconBiz. We use AP, AP50, and AP75 measures to compare the results.

	AP	AP50	AP75
ResNet50 (800px)	53.15%	**91.79%**	54.65%
ResNet101 (700px)	**53.94%**	91.58%	**58.51%**
ResNeXt50 32x8d (750px)	51.26%	89.44%	53.14%
ResNeXt50 64x4d (750px)	51.50%	89.38%	53.66%
ResNeXt101 32x8d (600px)	49.70%	88.60%	50.41%
ResNeXt101 64x4d (600px)	50.73%	88.84%	51.05%

Table 2: Performance of RestNet and ResNeXt on input images with unrestricted, i. e., maximal possible resolution.

5.2 Pretraining and Dataset Augmentation

The five datasets of scientific figures contain together only 633 images, which we had to split into training and testing datasets. This left us with 507 images for training and 126 images for testing the models. In order to improve our training results on these rather small datasets, we first report the result of pre-training on natural images. Subsequently, we present the results of dataset augmentation.

Pre-Training the Networks. For pre-training our networks, we use COCO-Text [44] and the Total-Text-Dataset (TTD) [8] (see Section 4.1 for details). In general, we assume that pre-training the network could increase the performance results. However, in the natural image there are only a low average of text elements per image, compared to the scientific figures datasets. Furthermore, due to the nature of the images in COCO-Text and TTD showing real-world scenes, the image content and the text detection tasks are very different, i. e., the structures learned in pre-training might not transfer to the actualy task. Our test results are shown in Table 3 and demonstrate that the pre-training is effective. Both TTD and COCO-Text improve the test results, with COCO-Text providing the best results by improving AP, AP50, and AP75 by about 5%, 9% and 2%, respectively.

Dataset Augmentation. We apply several steps to increase our training dataset size. First, we rotate the images by 90°, 180°, and 270°. Second, we scale the input images at random to 400, 600, or 700 pixels. Furthermore, we add horizontally or vertically flipped images, translate the images using two different translation matrices, and add Gaussian or salt and pepper noise to our data in order to increase general training results and robustness.

Pretraining	*none*	on TTD	on COCO-Text
AP	58.37%	65.73%	**65.98%**
AP50	91.35%	94.49%	**95.21%**
AP75	63.49%	75.51%	**76.33%**

Table 3: Comparison of AP, AP50, AP75 for ResNet101 with and without pre-training on COCO-Text and TTD.

We tested the effect of using augmented datasets on all different networks. Our initial tests show that ResNet50 surpasses ResNet101 in those experiments, if the training duration is relatively short, i. e., up to $\sim 100,000$ iterations, which might be interesting when training speed is important. The results, after increasing the training to $250,000$ iterations, are shown in Table 4. While the differences are relatively small, ResNet101 provides the best results in all three metrics, which is why we chose to use it in all our following experiments.

	AP	AP50	AP75
Best results w/o augmentation	*53.94%*	*91.79%*	*58.51%*
ResNet50	64.35%	93.95%	74.29%
ResNet101	**64.89%**	**94.98%**	**74.82%**
ResNeXt50 32x8	62.70%	93.56%	71.40%
ResNeXt50 64x4	61.22%	93.49%	70.92%
ResNeXt101 32x8	61.93%	94.49%	70.08%
ResNeXt101 64x4	61.01%	94.85%	69.73%

Table 4: Comparing the effect of our augmentation methods with different neural network architectures. All networks have been trained for $250,000$ iterations. Tested on CHIME-R, CHIME-S, DeGruyter and EconBiz. We compared the results using AP, AP50, and AP75 measures.

5.3 Different Trainings over the Datasets

We train five different networks on each individual dataset, i. e., CHIME-R, CHIME-S, DeGruyter, EconBiz, and DeTEXT. In addition, we train the network on all five datasets combined. Subsequently, we compare the test results individually per dataset and on the combined dataset to determine the effect of the different datasets on our training process. Our assumption is that networks trained on a dataset that are similar to the test dataset will perform better than dissimilar training/test sets. This gives an indication of the effect of adding additional datasets to our training set.

The results can be seen in Table 5. The first observation is that there seems to be a big difference between the CHIME datasets and both the DeGruyter and EconBiz datasets. While training on one CHIME set yields acceptable detections on the other CHIME set, the results for EconBiz and DeGruyter are very poor. This is expected because CHIME-S consists of synthetic data which is generated to be similar to CHIME-R. Interestingly, a similar relation

Train x Test	Results for AP75 tested on ...					
Trained on ...	CHIME-R	CHIME-S	DeGruyter	EconBiz	DeTEXT	Combined
CHIME-R	92.77%	61.19%	12.20%	2.62%	33.83%	40.52%
CHIME-S	81.23%	**88.72%**	7.27%	1.56%	32.20%	42.20%
DeGruyter	16.68%	1.05%	66.78%	56.31%	11.32%	30.43%
EconBiz	1.11%	0.96%	57.94%	**98.89%**	8.77%	33.53%
DeTEXT	40.10%	33.56%	18.11%	5.00%	**67.15%**	32.78%
All Combined	**96.48%**	64.63%	**70.17%**	94.77%	55.09%	**80.02%**

Table 5: Results for training on single datasets for 100, 000 **iterations on ResNet101. The experiments show similarities between the different datasets. We used the AP75 measure to compare the results.**

holds true for EconBiz and DeGruyter. Training on one of these datasets gives acceptable detection results for the other, but very poor results for both CHIME datasets. DeTEXT does not show these big discrepancies and the test results are similar to each other. The combination of all datasets improves the results on CHIME-R, CHIME-S, and DeTEXT, and only performs slightly worse on DeGruyter and EconBiz. As expected, the results, when testing on all five datasets, improve by a factor of two, when the network was also trained on all datasets. In summary, our network architecture is quite robust with respect to the datasets used for training and test. For DeGruyter and EconBiz, only 1%-2% accuracy is lost, but up to 10% precision is gained on the other datasets.

Regarding the generalization capabilities of our pipeline, we perform five tests. For each test, we choose one of our datasets for testing and train our pipeline on the remaining four. The results are shown in Table 6. The generalization results on CHIME-R, CHIME-S, DeGruyter, and EconBiz look promising. The AP50 measure of $\geq 80\%$ is enough in most cases to extract the text even though the bounding box precision is much worse compared to our previous tests. The results of DeTEXT are different and only reach an AP50 value of 68.67%. The difference between DeTEXT and the other datasets is that figures from DeTEXT contain many complex graphical elements and real-life medical images, which are very rare in the other datasets.

Tested on	AP	AP50	AP75
CHIME-R	45.82%	80.45%	45.19%
CHIME-S	41.12%	87.73%	30.59%
DeGruyter	43.06%	86.63%	35.88%
EconBiz	34.03%	84.61%	15.88%
DeTEXT	33.21%	70.32%	23.41%

Table 6: Testing the generalization capabilities of our localization network. We used pre-training on COCO- Text and ResNet101 as backbone. The network is trained for 200, 000 **iterations on four of the datasets and tested on the fifth. We compare the results using AP, AP50, and AP75.**

5.4 Comparison with Unsupervised Approach

We evaluate the best performing ResNet101 architecture, which was pretrained with COCO Text, with the best unsupervised approach from the study by Böschen et al. [2]. We use the best performing hyperparameters determined in the previous experiments, the data set augmentations, as well as the preprocessing of the input for the text recognition step. The results can be found in Table 7. For the text localization an AP of 67% is achieved and an F1 score of 0.87. Our pipeline achieves an average GPM of 0.85, which is far above the default threshold of 0.60 for a good match (see Section 4.4), and a Levenshtein distance of 3.44 when evaluating the text recognition results. A direct comparison with the top-performing pipeline BS-4OS-O from the study by Böschen et al. [2] shows that our pipeline improves the results significantly in all measured categories. Precision and recall increase by 19% and 20%, respectively, and the F1 measure improves accordingly. The average local Levenshtein Distance (LSD) is reduced by 1.26 and the global LSD is halved, when using our supervised pipeline compared to the unsupervised BS-4OS-O pipeline.

6 DISCUSSION

Comparing the different configurations of our localization network leads to the conclusion that the added layers in ResNet101 improve the results compared to ResNet50. Interestingly, the added complexity of ResNeXt in all tested configurations did not result in the same positive effect. Both observations are made for training on the original scientific figures datasets as well as when increasing the training set size by data augmentation techniques. The implemented data augmentation methods emerge as being very effective at increasing our test results. Despite using comparably simple methods like rotations, adding noise, resizing or flipping the image, the effect of these methods is very convincing.

Comparing our results to the results of the best performing configuration BS-4OS-O from the survey of Böschen et al. [2] shows that our results outperform the baseline on all measures by a large margin. It is important to note that there are fundamental differences between the approaches presented in the survey and our pipeline. Namely, we use a neural network that needs training data and limits our ability to test our methods on all the data contained in the datasets. But at the same time, this difference could also be seen as an advantage, because the effectiveness and generalizability of our networks mostly depend on the available training data. As a result, the advantage of our pipeline compared to the BS-4OS-O pipeline could be further improved by adding additional datasets to our training data. Regarding TencentAiLab, the best performing approach in the ICDAR 2017 DeTEXT challenge, we can only compare the AP50 values since there is no paper or further insights

	Precision	Recall	F1	LSD_avg	LSG	AP	AP 50	AP75	GPM_avg
BS-4OS-O pipeline	0.67	0.63	0.67	4.71	95.49	n/a	n/a	n/a	n/a
Our pipline	**0.86**	**0.83**	**0.87**	**3.44**	**39.11**	67.16%	94.71%	78.48%	0.85
Difference (or n/a)	**+0.19**	**+0.20**	**+0.20**	**-1.26**	**-56.38**	n/a	n/a	n/a	n/a

Table 7: Best neural network (our pipeline) versus best pipeline from Böschen et al. [2]. We use all measures that both papers have in common to compare the experiments. In addition, we show the values for the new measures applied in this paper.

about this method available (see discussion in Section 2). The AP50 value for TencentAiLab in the ICDAR 2017 DeTEXT challenge was 93.15%, while our neural-network-based pipeline achieves 93.49%. The results are not perfectly comparable since the exact dataset used in 2017 could not be reproduced as only about 40% of the images are still available. Nevertheless, even with less images available for training, validation, and test, our pipeline based on Faster R-CNN is on a par with the closed-source TencentAiLab solution. The similarity of the AP50 score on the DeTEXT dataset further suggests that perhaps a similar network architecture was used in TencentAiLab.

While the generalization capabilities of our pipeline from one dataset to another are good, there is as expected a problem with generalization when the scientific figures are of different chart types. Using our trained pipeline on scientific figures that contain charts or other graphical elements that were not present in our training set might be problematic. An indictation to this problem can be seen in Table 5. When training the model on CHIME-R or CHIME-S (both contain only bar, pie and line charts), it produces very poor results when tested on the other, more diverse datasets.

It is also apparent that the pipeline performs better on scientific figures that only contain (simple) graph-like structures. The results on CHIME-R, CHIME-S, and DeGruyter are much better than the results for EconBiz and DeTEXT. The former datasets contain mostly simple graphs, while EconBiz has many more elaborate graphical elements like maps and DeTEXT's figures include many medical images. The quantity (AP50) of localized text when comparing these two subsets differs only by $\sim 5\%$, but the difference in the quality of the detections (AP75) is as high as $\sim 40\%$. While the results are precise enough to extract the correct text in most cases, this implicates a problem with complex structures, which might be solved by adding more complex training data. The generalization test shows that our pipeline performs 10-19% worse when testing on DeTEXT compared to the other four datasets. Interestingly, the pipeline performs on EconBiz as well as on CHIME-R, CHIME-S, and DeGruyter.

The biggest problem for the generalization of our pipeline seems to be the complex medical images contained in DeTEXT, which might indicate a more general problem with real life images. The images in DeTEXT are structurally quite different. Furthermore, samples have multiple images in one or contain images taken from X-rays or magnetic resonance imaging. In contrast, other datasets are more homogenous and consist to a large extend only of digitally created scientific figures. It remains future work to further explore the generalization of our approach.

7 CONCLUSION

Our experiments show that the use of neural networks for text extraction from scientific figures is very effective, despite the rather small training datasets available. We have shown that our neural-network-based approach outperforms the previously best, unsupervised approach [2] by a large margin. At the same time, they underline one of the main problems of neural networks, the requirement for a large training data set. The experimental results on the non-augmented datasets are already convincing regarding the AP50 results, but they generally lacked in prediction precision, which is indicated by AP75. We successfully tackled this issue with our data augmentation methods and show that our pipeline scales well with the added training data. This indicates that the results could even improve further by adding even more data or more complex data augmentation methods. In the future, it would be interesting to see the effect of more complex augmentation methods like creating additional training data with GANs [17] or straightforward synthetic data generation methods [18, 25, 28].

The generalization results of our pipeline had rather lower AP75 values but retained in general convincing results regarding the AP50 measure. Only figures with very different content such as medical images from real patients with some annotation were problematic. However, given the success of the methods investigated over the already high variety of scientific figures in our experiments, we assume that in the future the results can be well improved when adding more medical images to the training datasets.

REFERENCES

[1] Fedor Borisyuk, Albert Gordo, and Viswanath Sivakumar. 2018. Rosetta: Large Scale System for Text Detection and Recognition in Images. In *International Conference on Knowledge Discovery & Data Mining*. ACM, New York, NY, USA, 71–79.

[2] Falk Böschen, Tilman Beck, and Ansgar Scherp. 2018. Survey and empirical comparison of different approaches for text extraction from scholarly figures. *Multimedia Tools Appl.* 77, 22 (2018), 29475–29505.

[3] Sandra Carberry, Stephanie Elzer, and Seniz Demir. 2006. Information graphics: an untapped resource for digital libraries. In *Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 581–588.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. 2018. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 4 (2018), 834–848.

[5] Xiangrong Chen and Alan L. Yuille. 2004. Detecting and Reading Text in Natural Scenes. In *Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 366–373.

[6] Daniel Chester and Stephanie Elzer. 2005. Getting Computers to See Information Graphics So Users Do Not Have to. In *International Symposium on Foundations of Intelligent Systems*, Vol. 3488. Springer, Berlin, Heidelberg, 660–668.

[7] Yao-Yi Chiang and Craig A. Knoblock. 2015. Recognizing text in raster maps. *GeoInformatica* 19, 1 (2015), 1–27.

[8] Chee Kheng Chng and Chee Seng Chan. 2017. Total-Text: A Comprehensive Dataset for Scene Text Detection and Recognition. In *International Conference on Document Analysis and Recognition*. IEEE, 935–942.

[9] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. 2016. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in neural information*

processing systems. 379–387.
[10] Marc Pierrot Deseilligny, Hervé Le Men, and Georges Stamon. 1995. Character string recognition on maps, a rotation-invariant recognition method. *Pattern Recognition Letters* 16, 12 (1995), 1297–1310.
[11] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* 88, 2 (2010), 303–338.
[12] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. 2018. Detectron. https://github.com/facebookresearch/detectron.
[13] Ross B. Girshick. 2015. Fast R-CNN. In *International Conference on Computer Vision, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 1440–1448.
[14] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 580–587.
[15] Julinda Gllavata and Bernd Freisleben. 2005. Adaptive fuzzy text segmentation in images with complex backgrounds using color and texture. In *International Conference on Computer Analysis of Images and Patterns*. Springer, 756–765.
[16] Lluis Gomez and Dimosthenis Karatzas. 2016. A fast hierarchical method for multi-script and arbitrary oriented scene text extraction. *International Journal on Document Analysis and Recognition* 19, 4 (2016), 335–349.
[17] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Annual Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets
[18] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. 2016. Synthetic Data for Text Localisation in Natural Images. In *Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2315–2324.
[19] Kaiming He et al. 2016. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*. 770–778.
[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *International Conference on Computer Vision*. IEEE, 2980–2988.
[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. 2017. Mask R-CNN. In *International Conference on Computer Vision*. IEEE Computer Society, 2980–2988.
[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 770–778.
[23] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely Connected Convolutional Networks. In *Conference on Computer Vision and Pattern Recognition*, Vol. 1. 3.
[24] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. 2016. Deep networks with stochastic depth. In *European Conference on Computer Vision*. Springer, 646–661.
[25] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. 2014. Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. *arXiv preprint* abs/1406.2227 (2014).
[26] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep Features for Text Spotting. In *European Conference on Computer Vision*, Vol. 8692. Springer International Publishing, Cham, 512–528.
[27] Chandrika Jayant, Matthew Renzelmann, Dana Wen, Satria Krisnandi, Richard E. Ladner, and Dan Comden. 2007. Automated tactile graphics translation: in the field. In *International ACM Conference on Computers and Accessibility*. ACM, New York, NY, USA, 75–82.
[28] Zhao Jiuzhou. 2006. *Creation of Synthetic Chart Image Database with Ground Truth*. Honors Year Project Report. National University of Singapore. https://www.comp.nus.edu.sg/~tancl/ChartImageDatabase/Report_Zhaojiuzhou.pdf.
[29] Christopher Kanan and Garrison W. Cottrell. 2012. Color-to-Grayscale: Does the Method Matter in Image Recognition? *PLOS ONE* 7, 1 (01 2012), 1–7.
[30] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, Vol. 8693. Springer International Publishing, Cham, 740–755.
[31] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. *arXiv preprint* abs/1708.02002 (2017).
[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.
[33] Jorge Poco and Jeffrey Heer. 2017. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Comput. Graph. Forum* 36, 3 (2017), 353–363.
[34] Joseph Redmon. 2013–2016. Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/.
[35] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Conference on computer vision and pattern recognition*. 779–788.
[36] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. *arXiv preprint* abs/1612.08242 (2017).
[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.
[38] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Annual Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, 91–99.
[39] Hanan Samet and Markku Tamminen. 1988. Efficient Component Labeling of Images of Arbitrary Dimension Represented by Linear Bintrees. *IEEE Trans. Pattern Anal. Mach. Intell.* 10, 4 (1988), 579–586.
[40] Jerzy Sas and Andrzej Zolnierek. 2013. Three-Stage Method of Text Region Extraction from Diagram Raster Images. In *International Conference on Computer Recognition Systems*, Vol. 226. Springer International Publishing, Heidelberg, 527–538.
[41] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint* abs/1409.1556 (2014).
[42] Nitish Srivastava et al. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[43] Jasper R. R. Uijlings, Koen E. A. van de Sande, Theo Gevers, and Arnold W. M. Smeulders. 2013. Selective Search for Object Recognition. *International Journal of Computer Vision* 104, 2 (2013), 154–171.
[44] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge J. Belongie. 2016. COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images. *CoRR* abs/1601.07140 (2016).
[45] Andreas Veit, Michael J Wilber, and Serge Belongie. 2016. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*. 550–558.
[46] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated Residual Transformations for Deep Neural Networks. In *Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 5987–5995.
[47] Songhua Xu and Michael Krauthammer. 2010. A new pivoting and iterative text detection algorithm for biomedical images. *Journal of Biomedical Informatics* 43, 6 (2010), 924–931.
[48] Chenggang Yan, Hongtao Xie, Shun Liu, Jian Yin, Yongdong Zhang, and Qionghai Dai. 2018. Effective Uyghur language text detection in complex background images for traffic prompt identification. *IEEE transactions on intelligent transportation systems* 19, 1 (2018), 220–229.
[49] Chun Yang, Xu-Cheng Yin, Hong Yu, Dimosthenis Karatzas, and Yu Cao. 2017. ICDAR2017 Robust Reading Challenge on Text Extraction from Biomedical Literature Figures (DeTEXT). In *International Conference on Document Analysis and Recognition*. IEEE, 1444–1447.
[50] Li Yang, Weihua Huang, and Chew Lim Tan. 2006. Semi-automatic Ground Truth Generation for Chart Image Recognition. In *International Workshop on Document Analysis Systems*, Vol. 3872. Springer Berlin Heidelberg, Berlin, Heidelberg, 324–335.
[51] Xu-Cheng Yin, Chun Yang, Wei-Yi Pei, Haixia Man, Jun Zhang, Erik Learned-Miller, and Hong Yu. 2015. DeTEXT: A database for evaluating text extraction from biomedical literature figures. *PLoS One* 10, 5 (2015), e0126200.
[52] Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision*, Vol. 8689. Springer International Publishing, Cham, 818–833.
[53] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In *Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 6848–6856.
[54] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. 2016. Multi-oriented text detection with fully convolutional networks. In *Conference on Computer Vision and Pattern Recognition*. 4159–4167.
[55] C. Lawrence Zitnick and Piotr Dollár. 2014. Edge Boxes: Locating Object Proposals from Edges. In *European Conference on Computer Vision*, Vol. 8693. Springer International Publishing, Cham, 391–405.