Text Detection And Extraction Using OCR

(Optical Character Recognition)

-A CNN Approach in Street Navigation

IDS 705
Team Orange

Yi Feng
Jennie Sun
Ruoshui Li
Michael Tang
Shangwen Yan

Abstract

Previous approaches for text detection and information extraction with Optical Character Recognition (OCR) techniques have already achieved promising performances across various applications. However, open source text recognition softwares, such as Tesseract, tends to fall short when recognizing natural scene text-based images. In this work, we propose a pipeline that combines a tuned Convolutional Neural Network (CNN) model (LeNet) and an existing deep learning model (EAST text detector), which yields relatively accurate text detection and localization results for street sign images. This pipeline performs text localization, character segmentation, and character classification on an image, and outputs recognized text as the results. When further combined with a text-to-speech program, this pipeline can potentially provide people who are visually impared with the technology and ability to recognize and understand street signs.

Introduction

Extracting text information in images has a wide variety of applications. It can address general tasks like converting handwritten and printed texts in images to digital, machine readable text. Field-specific tasks like passport recognition, license plate recognition, electricity meter reading, and traffic sign recognition can also benefit from image text extraction. However, most of the current text recognition products focus merely on extracting text from scanned documents. While they have stellar performance on PDF files or scanned documents, existing services generally do not help tackle OCR tasks in the real world, such as recognizing text from street signs and camera-taken pictures. One example is Tesseract, an open source text recognition software sponsored by Google. When tested on a computer generated image (Figure 1), Tesseract demonstrates excellent performance in segmenting characters within words.

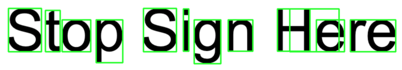

Figure 1. Computer-generated image

However, when applied to street sign images taken by cameras (Figure 2), Tesseract failed to detect or recognize any text.

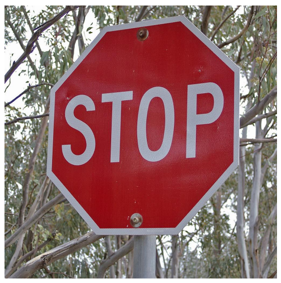

Figure 2. Street sign taken by camera

Therefore, we are interested in exploring the implementation of OCR that works in more general use cases, including both computer-generated images and camera-taken pictures. Successful implementation could potentially be applied to applications including licence plate recognition, automatic meter reading (AMR), and street sign reading. As a result, stakeholders can leverage it for recognition tasks that have been resource-demanding or susceptible to human errors.

Background

A common OCR pipeline consists of three stages: text localization, segmentation, and classification.

In text localization, the goal is to automatically compute the bounding boxes for every word region in an image, which will be processed by segmentation algorithms. In the fundamental stage, a detector's performance largely determines overall accuracy and processing speed of the whole recognition result (Laroca et al., 2019). As many new applications of computer vision techniques have been introduced with deep learning, traditional detectors relying on connected components analysis have been replaced by state-of-art CNN-based ones, with significantly improved accuracy of localization. State-of-the-art object detection algorithms include YOLO (You Only Look Once) (Redmon et al., 2016; Busta et al., 2017), Fast Region-CNN (Ma et al., 2018; Ren et al., 2017), and FCRN (Fully Convolutional Regression Network) (Gupta et al., 2016; Xie et al., 2016).

Text segmentation divides word regions into contiguous characters based on shape and structure. Generally, this not only involves segmenting line, word, character, but also isolating the text body from the background (Saha et al., 2010). Current segmentation algorithms include both CNN-based techniques (Jo et al., 2020) and traditional image processing methods.

The segmented texts are then fed into classification algorithms. When combining digit segmentation with classification, approaches such as CR-NET, Multi-Task, and CRNN, can all achieve promising results. When it comes to single-character classification, CNN-based models and traditional classifiers can be used.

Other than tackling text localization and segmenting characters separately, some literatures also proposed end-to-end frameworks that simultaneously detect and recognize text in images. These solutions can sometimes outperform the combination of state-of-the-art localization and recognition methods (Gupta et al., 2016; Busta et al., 2017), by avoiding over-segmentation or error amplification problems (Gómez ET AL., 2018).

Overall, the larger body of work tries to address challenges of translating images using various architectures. In our work, we aim to design an adaptive navigation tool for visually impaired individuals. This requires a combined effort from both the computer vision and speech processing communities. Here, we decided to focus on OCR and its classification component. Because navigation involves reading street signs, store banners (handwritten fonts) etc, which mostly consists of alphanumeric characters, we tackle this problem by devising a NN-based classifier trained with the EMNIST and the Chars74k dataset. This classifier along with mature localization and segmentation architectures can extract texts from street sign or traffic sign images.

Data

Our image data for street signs and storefronts pose a challenge that is three fold: firstly, how can we accurately identify textual information from irrelevant background information? Secondly, how do we segment localized text into classifier-compatible characters? Lastly, how do we construct a good classifier for recognizing characters?

After scrutinizing the problem, we have come up with the following strategies: use specialized text localization algorithms to help locate word regions on images; segment characters through contour detection; classify characters by training a NN-based model.

Figure 3: An all-way stop sign

In our analysis, we used the EMNIST (Extended MNIST) and Chars74K datasets to train the classifier because they both encompass digits and English alphabets that are common in street signs. Additionally, the combination of both the handwritten and printed forms allow for better generalizability.

The EMNIST dataset follows the same conversion paradigm used to create the MNIST dataset. In this dataset, we used the EMNIST Balanced split, which includes 47 balanced classes, with 131,600 alphanumeric characters in total. The Chars74K English dataset consists of 64 classes (0-9, A-Z, a-z), with a total of 7705 characters obtained from natural images, 3410 hand-drawn characters using a tablet PC, and 62992 synthesised characters from computer fonts. This produces a total of over 74K samples.

One of the challenges we encountered with these 2 datasets was the difference in number of classes. Specifically, the EMNIST data merged 15 classes such as ‘c’, ‘i’, ‘j’, where their lower case forms pose similar structural features as their uppercase counterparts. We therefore applied the same rule to the Chars74K dataset. However, we still expect a relatively poorer performance in predicting similar looking characters. Furthermore, a lower accuracy in cross-domain validation is expected. In order to overcome these two foreseen challenges, we will combine the two datasets and test on the combined version to optimize performance and data representation.

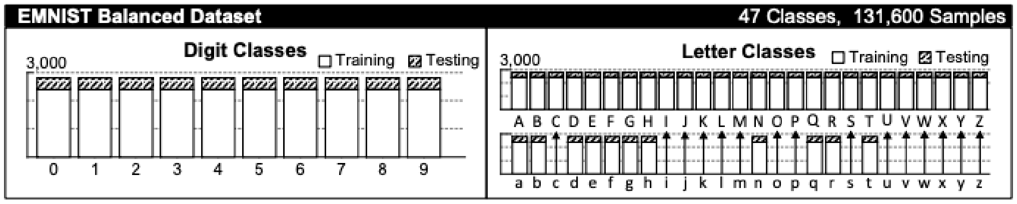

Figure 4. EMNIST Balanced Dataset Structure

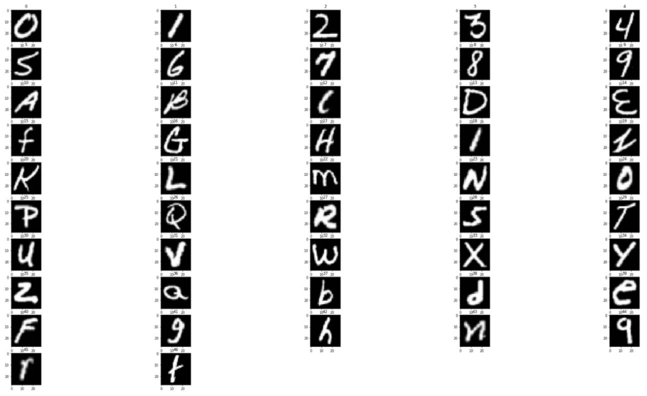

Figure 5. EMNIST Balanced Dataset Visualization - 47 Classes

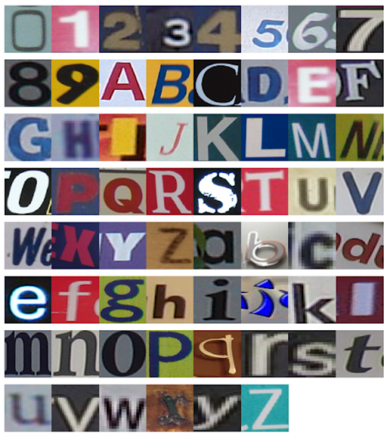

Figure 6. Chars74K Dataset Visualization - 64 Classes

Methods

Using the 3-stage paradigm of OCR, the Text Localization step accepts a 2-dimensional array as the input, and locates one or more word regions within that image. More generally, the problem of text localization falls under the umbrella of object detection. Traditional approaches involve CNN-based methods such as Fast R-CNN and YOLO. These implementations differ mainly in their speed, where YOLO is significantly faster and offers superior real-time detection. For the purpose of OCR, we decided to use EAST, a specialized text scenes detector. The architecture of EAST is shown in the figure below:

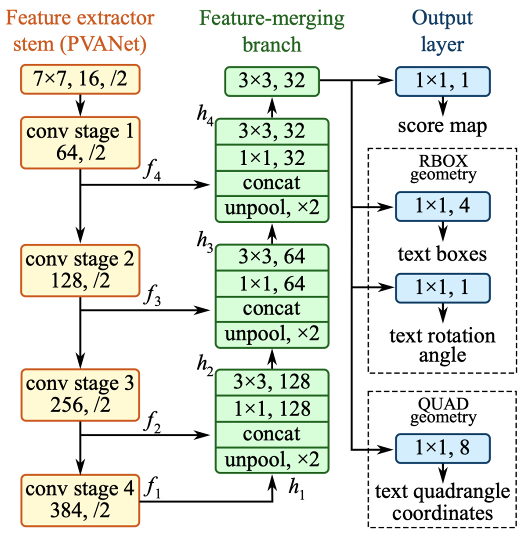

Figure 7. Architecture of EAST model

The feature extractor stage comprises 4 convolutional layers, which produces 4 feature maps that adequately extract features from word regions of different sizes. All 4 levels of features are gradually merged into a U-shaped pipeline to minimize the computational cost and to optimize the word detection at different spatial resolutions. The final output layer contains the confidence score and geometric information of each bounding box. A predefined threshold value and Non-max Suppression are used to suppress any weak or overlapping bounding boxes.

The character segmentation stage segregates word regions further into individual characters. We decided to pursue an non-deep learning based approach, because YOLO can struggle with low resolution images (Borel-Donohue & Young, 2019), which are often the cases for character segmentation. Concretely, we utilize image-processing techniques including blurring, thresholding, eroding on each word, as depicted in Figure 8. The ideal output of these processes is a binary image where each character is clearly defined by a contour. A bounding box for each character can then be produced with the contour approximation method.

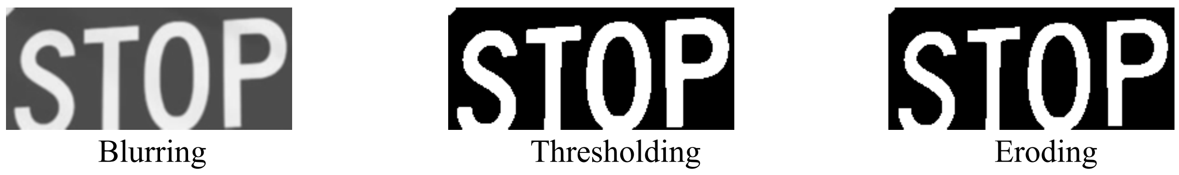

Figure 8. Image-processing Techniques

Once the bounding boxes for all characters are identified, they are fed into our classifier. In order to have the best-performing character classifier, we have adapted five different machine learning algorithms on both datasets to select the one with the highest performance. These classifiers are - SVM, Logistic Regression, MLP, LeNet and AlexNet. One thing to note is that the classifier is chosen independently from the OCR pipeline.

For classifiers evaluation, we trained and tested each classifier on EMNIST and Chars74K, respectively. We adopted a train-test split ratio of 80:20, and compared classifier performance based on test accuracy. After considering results across both datasets, the final result suggested LeNet was the best performing model for character recognition, and we decided to keep it as part of the final OCR pipeline. Details on performance comparison can be found in the next section.

For preprocessing, each character from the text segmentation was reshaped to (28, 28, 1) as proposed by the original literature. Padding is also added to center the character and match the characteristics of the training data. Furthermore, pixels are divided by 255 to normalize the image before performing classification.

Despite not having the full control of the first 2 stages of the pipeline, an optimized classifier can improve the overall OCR performance and contribute to the goal of providing a more assistive navigation tool for the visually impaired. The complete OCR pipeline and the corresponding output format are summarized in the figure below:

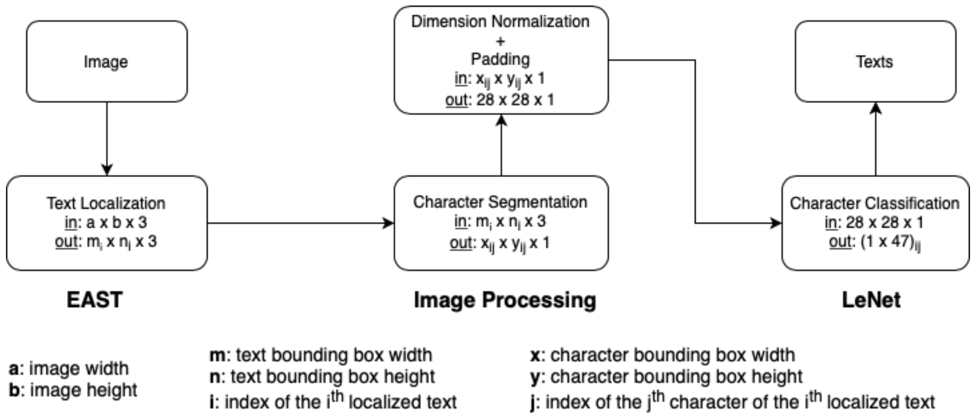
Image
Dimension Normalization + Padding
in: x_{ij} x y_{ij} x 1
out: 28 x 28 x 1
Texts
Text Localization
in: a x b x 3
out: m_i x n_i x 3
Character Segmentation
in: m_i x n_i x 3
out: x_{ij} x y_{ij} x 1
Character Classification
in: 28 x 28 x 1
out: $(1 \times 47)_{ij}$
EAST
Image Processing
LeNet
a: image width
b: image height
m: text bounding box width
n: text bounding box height
i: index of the ith localized text
x: character bounding box width
y: character bounding box height
j: index of the jth character of the ith localized text

Figure 9. Pipeline Flowchart

Results

For both EMNIST and Chars74K datasets, we used a train-test split ratio of 80:20. Model accuracy and loss were examined during the training process to prevent overfitting or underfitting before we apply the training result for prediction.

During the early-stage analysis, we trained 5 machine learning models on the EMNIST Balanced dataset and evaluated our model performance by comparing the accuracy scores across these models. Below summarizes accuracies from those models.

Table 1: Summary Table of Test Accuracy by Model

Model	**SVM**	**Logistic Regression**	**MLP**	**LeNet**	**AlexNet**
Accuracy	0.84	0.75	0.85	0.88	0.84

Since LeNet returned the highest accuracy, we decided to adopt this model and further tuned its hyperparameters before adding it into our OCR pipeline. And we used the result of AlexNet as our baseline for the analysis.

In order to evaluate how well our models performed, we also conducted cross domain validation, where the model was trained and tested on different datasets to optimize its generalizability.

Table 2: Summary Table of LeNet Within Domain and Cross Domain Test Accuracy by Dataset

Test / Train (Model: LeNet)	**Chars 74K**	**EMNIST**
Chars 74K	0.88	0.03
EMNIST	0.38	0.86
Chars 74K & EMNIST Combined	0.85	

From the evaluation matrix above, we can see that our LeNet model performed well in within-domain validation, with 0.88 on the Chars74K dataset and 0.86 on the EMNIST dataset. However, when it comes to cross-domain evaluation, the same model performed poorly, with 0.38 accuracy when trained on the Chars74K dataset and tested on the EMNIST dataset, and only 0.03 accuracy when trained on the EMNIST dataset and tested on the Chars74K dataset. It is also worth noting that the model trained on Chars74K still had a much higher accuracy when tested on the EMNIST dataset, compared to the other way around. This is likely due to the relatively higher complexity for the computer generated characters in the Chars74K dataset.

We also trained our model and tested it on the combined version of the two datasets, which resulted in an accuracy of 0.85 that is very close to the within domain accuracies in both scenarios mentioned above (0.88 and 0.38). This is expected since the training data now are much more representative of the testing data compared to the cross-domain scenario.

Furthermore, we examined other classification metrics at a more granular level. Specifically, we computed precision[1], recall[2], F-measure[3], and support[4] for each class in the combined dataset. For classes that are more likely to be indistinguishable, such as ‘F’ vs ‘f’ and ‘O’ vs ‘0’, we observe relatively low F1 scores (0.68 vs 0.59, 0.36 vs 0.64, respectively).

Additionally, we compared the computational speed across different combinations. It took the tuned LeNet model 1,844 second to train on the EMNIST dataset, and slightly longer for it to train 1,919 seconds on the combined dataset.

With the complete pipeline, we can see from Figure 9 that the OCR successfully detected the text in the image, shown by a green bounding box around the text ‘STOP’, and accurately recognized the text. However, the pipeline sometimes produced inaccurate results, as shown in Figure 10.

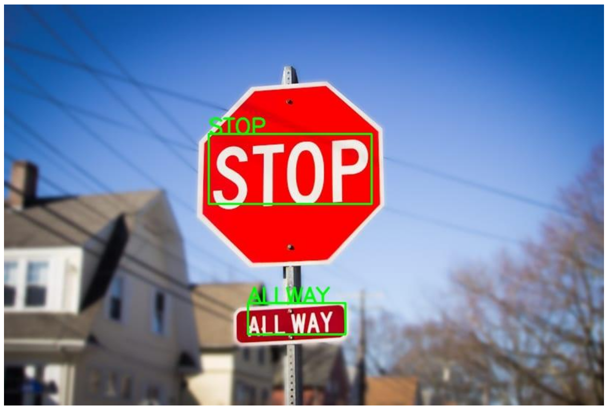

Figure 10: Words on the all-way stop sign image recognized by the OCR pipeline

Our OCR in both cases correctly identified the word regions, which suggested errors made in Figure 11 could have resulted from the segmentation stage, where certain features that make a character distinctive had been obscured/removed through processes such as blurring and erosion. Therefore, characters such as ‘B’ and ‘D’ could easily be misclassified if any parts of the character body are concealed. Other characters such as ‘O’, ‘S’ and ‘W’ are highly distinctive, making them less prone to classification errors. Additionally, the majority of our training data came from the EMNIST dataset. This means that our classifier does not generalize well to printed characters, which is the case for street signs. Future work that could make our classifier more robust is described in the next section.

[1] The precision is the ability of the classifier not to label as positive a sample that is negative.
[2] The recall is the ability of the classifier to find all the positive samples.
[3] The F1 score can be interpreted as a weighted average of the precision and recall.
[4] The support is the number of occurrences of each class in true labels.

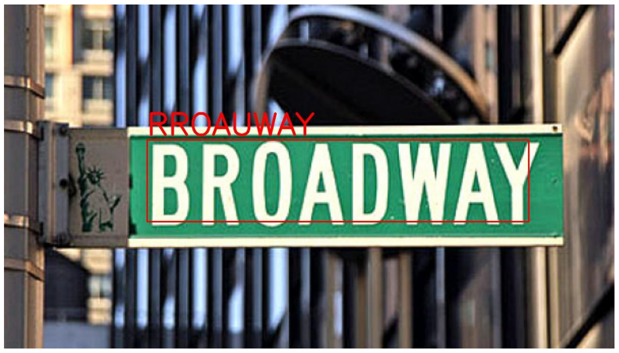

Figure 11: OCR pipeline recognizes BROADWAY as 'RROAUWAY'

Conclusions

Optical Character Recognition, based on our exploration and experimentation, is a powerful and promising field that has a variety of use cases. Our project focused on reviewing the research and application status-quo of OCR, building a pipeline imitating standard framework, as well as combining existing packages with our trained CNN model (LeNet) to recognize text-based street images.

It is important to note that since the LeNet model was only trained and tested on two datasets, we could potentially improve the by-class and overall accuracy by exploring more relevant datasets during our model training and tuning stage. This way, our model would be more representative of different types of text-based image data and more adaptable to various real-life applications. Another issue that needs to be addressed is the computational speed for model training. In order to develop a scalable machine learning pipeline that works more efficiently for OCR applications, we could further optimize our hyperparameter search to minimize loss and boost model performance.

Currently, our pipeline is only suitable for detecting text in static images. Going forward, we would also like to apply this technique to perform real-time text recognition. Once we have optimized the performance of our model to a greater extent, we can integrate our OCR pipeline with services that convert text into audible speech (e.g., AWS Text-to-Voice Solutions) for close-to real-time street navigation aid. Although there still exists a gap in terms of accuracy and computational speed between our pipeline and cutting-edge OCR techniques, we believe this project has made a meaningful attempt in envisioning a more tech-integrated environment where people with disabilities can lead a quality life just like other members of the society.

Roles

Ruoshui	Took the role of doing literature review and reading up similar work. Contributed to training and tuning of the multi-class logistic regression model during classification stage. Tried to implement YOLO object detection on electricity meter images.
Shangwen	Cleaned and preprocessed data; trained and tuned an multi-class SVM classifier on the data for the character classification part of our OCR pipeline; researched state-of-the-art OCR softwares and platforms such as Tesseract, Google Vision AI and Amazon Textract. Explored each state-of-the-art OCR's strength and weakness.
Michael	Trained and tuned the LeNet model on the character dataset. Researched the architecture and open source tools for constructing an OCR pipeline. Implemented EAST-based text detector and image processing-based character segmentation. Streamlined the process by integrating the pipeline with our final LeNet classifier. Tune and examined OCR performance.
Jennie	Trained an MLP model for character classification; explored, cleaned, and prepared the two major datasets to ensure the data are ready for experimental conditions and applications of the OCR pipeline; compiled, visualized, and compared performance metrics from the LeNet model; provided a coherent assessment across the results from different workstreams.
Yi	Took on the role of project manager and enacted weekly meeting agendas and kept track of meeting notes; contributed to the training/tuning of one of the CNN models - AlexNet on character data; collaborated with Jennie on within-sample and cross-sample training and validation of LeNet, as well as model metrics collection.

Timeline of Activity

Milestone	Date
Conducted group literature review and cutting-edge techniques on OCR	February, 27th
Confirmed OCR architecture and machine learning models for character recognition	March, 6th

Finished training different models on EMNIST and compared results	March, 13th
Finished training models on Chars74K, decided to use LeNet going forward	March, 20th
Applied state-of-the-art techniques to realize text localization	April, 4th
Designed workflow for character segmentation	April, 14th
Inserted tuned LeNet model into OCR pipeline; tested and collected metrics on OCR pipeline	April, 19th

References

Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). East: an efficient and accurate scene text detector. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 5551-5560).

Nguyen, Q. (2020, May 18). Detect and Recognize Vehicle’s License Plate with Machine Learning and Python — Part 2: Plate character segmentation with OpenCV. Medium. https://medium.com/@quangnhatnguyenle/detect-and-recognize-vehicles-license-plate-with-machine-learning-and-python-part-2-plate-de644de9849f

Laroca, R., Barroso, V., Diniz, M. A., Gonçalves, G. R., Schwartz, W. R., & Menotti, D. (2019). Convolutional neural networks for automatic meter reading. Journal of Electronic Imaging, 28(01), 1. https://doi.org/10.1117/1.jei.28.1.013023

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 27–30. https://doi.org/10.1109/cvpr.2016.91

Busta, M., Neumann, L., & Matas, J. (2017). Deep TextSpotter: An End-to-End Trainable Scene Text Localization and Recognition Framework. 2017 IEEE International Conference on Computer Vision (ICCV), 2204–2212. https://doi.org/10.1109/iccv.2017.242

Ma, J., Shao, W., Ye, H., Wang, L., Wang, H., Zheng, Y., & Xue, X. (2018). Arbitrary-Oriented Scene Text Detection via Rotation Proposals. IEEE Transactions on Multimedia, 20(11), 3111–3122. https://doi.org/10.1109/tmm.2018.2818020

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149. https://doi.org/10.1109/tpami.2016.2577031

Gupta, A., Vedaldi, A., & Zisserman, A. (2016). Synthetic Data for Text Localisation in Natural Images. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1. https://doi.org/10.1109/cvpr.2016.254

Xie, W., Noble, J. A., & Zisserman, A. (2016). Microscopy cell counting and detection with fully convolutional regression networks. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 6(3), 283–292. https://doi.org/10.1080/21681163.2016.1149104

Saha, S., Basu, S., Nasipuri, M., & Basu, D. K. (2010). A Hough transform based technique for text segmentation. arXiv preprint arXiv:1002.4048.

Jo, J., Koo, H. I., Soh, J. W., & Cho, N. I. (2020). Handwritten Text Segmentation via End-to-End Learning of Convolutional Neural Networks. Multimedia Tools and Applications, 79(43–44), 32137–32150. https://doi.org/10.1007/s11042-020-09624-9

Gupta, A., Vedaldi, A., & Zisserman, A. (2016). Synthetic data for text localisation in natural images. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2315-2324).

Gómez, L., Rusinol, M., & Karatzas, D. (2018, April). Cutting sayre's knot: Reading scene text without segmentation. application to utility meters. In 2018 13th IAPR International Workshop on Document Analysis Systems (DAS) (pp. 97-102). IEEE.

Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. Retrieved from http://arxiv.org/abs/1702.05373

T. E. de Campos, B. R. Babu and M. Varma. Character recognition in natural images. In Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP), Lisbon, Portugal, February 2009.

Borel-Donohue, C. C., & Young, S. S. (2019). Image quality and super resolution effects on object recognition using deep neural networks. Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, 54. https://doi.org/10.1117/12.2518524