Deep Neural Networks based Optical Character Recognition for Parking Signs

Muhammed Adil YATKIN MUHAMMED@UT.EE
Gholamreza Anbarjafari (Shahab) GHOLAMREZA.ANBARJAFARI@UT.EE
Till Quack TILL@MAPILLARY.COM

Editor: Muhammed Adil YATKIN

Abstract

Optical Character Recognition_(OCR) is a common computer recognition technology that is used to extract textual information from an image or document. OCR technologies often detect and extract the text contained in images and make it computer-readable. As shown in many previous studies, text written images do not always have good quality to be recognized by automatic OCR systems. This study compares the performance of automatic OCR systems such as Google and Amazon on the parking sign OCR dataset that we have prepared. In addition, it is proved that a system developed as a result of deep research on the latest technologies can give more efficient and high accuracy results.

Keywords: Optical Character Recognition, Text Detection, Text Recognition, Convolutional Neural Network, Recurrent Neural Network

1. Introduction

Extracting textual information from natural images is an important issue that has recently attracted attention in the computer vision field and many studies[1; 2; 3; 4] have adressed the topic. Although work in this area has achieved great success, this subject is still challenging due to the long text sentences, poor photo quality and complicated background.

Recently, Optic Character Recognition models based on the deep learning method[5; 6; 7; 8; 2] have shown promising performances. Most of these models handle and solve the problem as two separate tasks, text detection and text recognition[9; 4].

The aim of this paper to compare the performance of these models based on deep learning methods. To achieve this goal, a dataset of parking signs was prepared first and then the state-of-the-art technology was investigated in depth. A high precision system has been developed to recognize parking sign images.

This work offers the possibility to create a better custom OCR system based on final detection and independent recognition models.

2. Related Work

Text detection and recognition is an active topic in the fields of computer vision and document analysis. In this section, a close examination and analysis of some methods will be presented.

2.1. Text Detection

Distinguishing text from natural images is one of the hot topics that computer vision field has focused on in recent years. Many important papers have been published on this subject in recent years[10; 11; 9].

Usually the models proposed in these papers are compared to some official benchmark datasets such as ICDAR series, CocoText, SynthText etc.[12; 13; 14; 15; 16]

Text detection plays a critical role in the procedure of extracting and understanding all text information. The basis of text detection is the design of the features that distinguish the text from the backgrounds.

Many text detection approaches were difficult to perform in challenging scenarios before the field of text learning with deep learning was discovered. Traditional approaches use manually designed[17; 18; 16; 19] features, while deep learning methods learn effective features from training data. These traditional approaches are often multi-stage, and overall performance is slightly less. In this work, three state-of-the-art works will be analyzed for the purpose.: EAST[2], FOTS[5], CRAFT[20] .

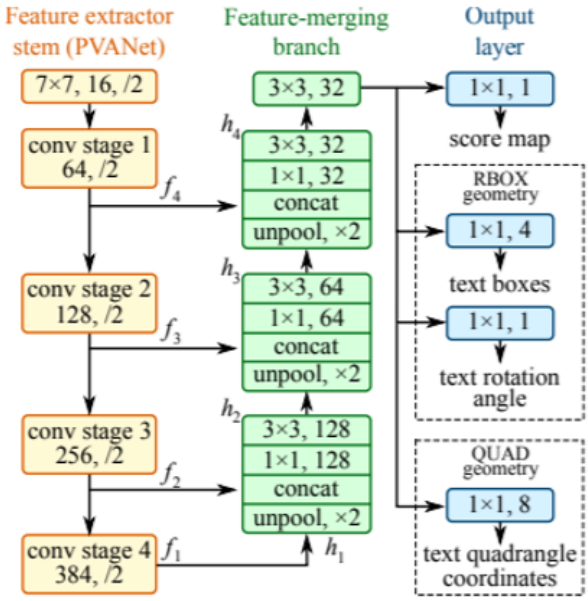

Figure 1: Structure of EAST text detection

2.1.1. EAST: An Efficient and Accurate Scene Text Detector

EAST is a deep learning-based algorithm that detects text with a single neural network with the elimination of multi-stage approaches. The EAST algorithm uses only one neural network to predict a line of text given within the image. It can detect text in arbitrary orientation with quadrilateral shapes. In 2017, this algorithm performed better than cutting-edge methods. This algorithm consists of a fully Convolutional network with a non max suppression (NMS)[21; 22] merging state. Inside the model the Convolutional Neural Network is used to localize text in the image,and this NMS stage is mainly used to combine many imprecise text boxes into one bounding box for each text region (word or line text)

EAST architecture was created taking into account different sizes of word regions. The idea is to identify small word regions that require low-level features from the first stages, while identifying large word regions that require features from the next stage of the neural network. To create this network, the authors used three branches joined together in a single neural network.

Feature Extractor Stem: This stage of the network is used to extract features from different layers of the network. This part can be a pre-trained convolution network on the ImageNet dataset[23] dataset. Authors of EAST architecture used PVANet[24] and VGG16[25] for experiment.

Feature Merging Branch: In this branch of the EAST network, it combines feature outputs from a different layer of the VGG16 network. The in put image is passed through the VGG16 model and outputs are obtained from four different VGG16 layers. Combining these feature maps can be computation ally expensive, because EAST uses a U-net architecture to progressively combine feature maps, shown in Fig. 1. First, the results from the pool5 layer are sampled upward using a deconvolution layer. Then the size of the properties after this layer will be equal to the output from the pool4 layer and both will be combined in a single layer. Conv 1 X 1 and Conv 3 X 3 are then applied to combine the information and produce the output of this merge step.

Output Layer: The output layer contains a score map and a geometry map, shown in Fig. 2. While the score map shows the probability of text in that region, the geometry map defines the boundary of the text box. This geometry

Geometry	channels	description
AABB	4	$\mathbf{G} = \mathbf{R} = \{d_i \mid i \in \{1,2,3,4\}\}$
RBOX	5	$\mathbf{G} - \{\mathbf{R}, \theta\}$
QUAD	8	$\mathbf{G} = \mathbf{Q} = \{(\Delta x_i, \Delta y_i) \mid i \in \{1,2,3,4\}\}$

Table 1. Output geometry design

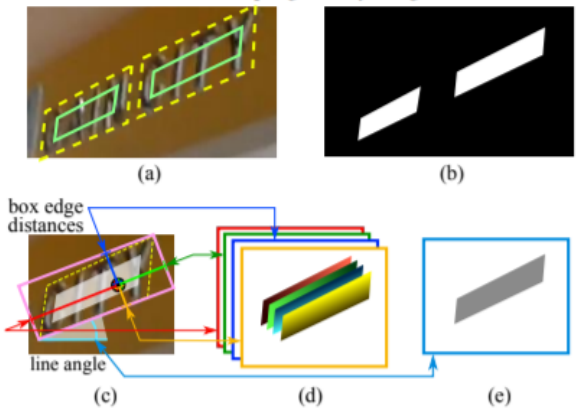

Figure 2: Score and Geometry Map Generation. (a) Text quadrangle (yellow dashed) and the shrunk quadrangle (green solid); (b) Text score map; (c) RBOX geometry map generation; (d) 4 channels of distances of each pixel to rectangle boundaries; (e) Rotation angle. This image is taken from[2].

map can be either a rotated box or quadrangle. A rotated box consists of the upper-left coordinate, width, height, and rotation angle for the text box. While quadrangle consists of all four coordinates of a rectangle.

Loss Function: The loss function in [2] is determined as follows:

$$L = L_s + \lambda L_g \tag{1}$$

where L_s and L_g represent the losses for the score map and the geometry, respectively, and λ weighs the importance between two losses. In the EAST paper, authors determined λ as 1.

Non-max Suppression Merging Stage: After the fully Convolutional network, the predicted geometries are passed through a threshold value. After this threshold, the remaining geometries are suppressed using a locality-aware NMS. Naive NMS runs on $O(n2)$. However, to run it in $O(n)$, the authors have adopted a method that uses row-by-row suppression. This row-by-row suppression also takes into account the iterative merging of the last merged item. This makes this algorithm fast in most cases, but the worst time complexity is still $O(n2)$.

2.1.2. FOTS: Fast Oriented Text Spotting with a Unified Network

As mentioned earlier, in many models, text detection and recognition are treated as two separate tasks. In the FOTS[5] paper, the authors propose a model capable of detecting and recognizing text simultaneously from the image which means that FOTS is a unified end to end trainable network that uses sharing computation and visual information among the two complementary tasks.

The authors suggest a strategy called shared Convolutional Neural networks, creating a network model that works much faster than two-stage methods. Although this model requires less computation, it learns more common features.

The key point in the FOTS[5] model is to establish the link between detection and recognition parts. The model achieves this connection with the **ROIRotate** technique.

FOTS Network Architecture: The architecture is presented in Fig. 4. Feature maps are firstly extracted with shared convolutions. The fully convolutional network based oriented text detection branch is built on top of the feature map to predict the detection bounding boxes. The RoIRotate operator extracts text proposal features corresponding to the detection results from the feature map. The text proposal features are then pass into Recurrent Neural Network (RNN) encoder and Connectionist Temporal Classification (CTC)[26] decoder for text recognition.

RoIRotate: RoIRotate applies transformations on oriented feature regions to obtain axis-aligned feature maps, as shown in Fig. 3. First, it calculates affine parameters with the help of ground truth and predicted coordinates of text propos-

Figure 3: RIO-rotate example. The proposed method uses the input image to illustrate text locations, but it is actually operated on feature maps in the network[5].

als. Then, affine transformations are applied to shared feature maps for each zone and canonical horizontal feature maps are derived from text regions.

Loss Function: In this model, the loss function is calculated by combining the losses from detection and recongition.

$$L = L_{detect} + \lambda L_{recog} \tag{2}$$

λ controls the trade off between recognition and detection loss which is taken 1 by the authors of the paper.

2.1.3. CRAFT: Character Region Awareness for Text Detection

Unlike EAST and FOTS, the CRAFT model uses character level annotation for text detection training. However, it is difficult to produce and find a dataset annotated on character level.

In the CRAFT paper[20], the authors first propose a solution to the limited character-level dataset problem. Then they come up with a new model for text detection.

Although there is no public character-level dataset available, there are large datasets synthetic as annotated such as SynthText[27]. Using this advantage, the authors propose a model that can make character-level annotation to world-level annotated datasets like ICDAR datasets[14; 12; 28].

Ground Truth Label Generation: The model in the paper trains synthetic and real images together and produces the **region score** and **affinity score** for word-level annotated boxes. In the paper, authors describe this method as **Weakly-Supervised Learning**. This training is important to extract character level annotation from real pictures. The region score gives the probability that each given pixel belongs to the center of the character and the affinity score represents the center probability of the space between adjacent characters, shown in Fig. 5.

Weakly Supervised Learning: When a real image with word-level annotations is provided, the learned interim model predicts the character region score of the cropped word images to generate character-level bounding boxes, as shown in Fig. 6. Also, the confidence map is calculated by dividing the number of specified characters and the actual character number to ensure the reliability of the model's prediction. This calculated confidence map is used for weights during training. This training process consists of three stages. First, the world-level annotated boxes are cropped from real images. Second, the model trains until region score prediction. Third, the watershed algorithm[29] is used to separate character compositions.

While training, the CRAFT model can predict characters more accurately. In early training stages, region score can be relatively low. This situation stems from unfamiliar text in natural images. The model learns appearances of new texts, such as irregular fonts, and synthesized texts that have a different data distribution against the SynthText dataset.

The CRAFT model was released in 2019 and has a superior performance compared to previously released models. Results related to the comparison can be learned from the paper[20].

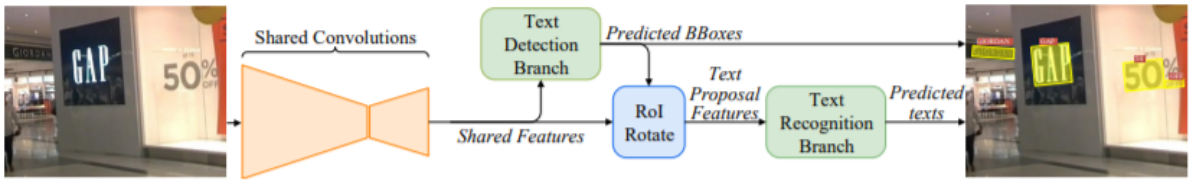

Figure 4: FOTS architecture. The network predicts both text regions and text labels in a single forward pass[5].

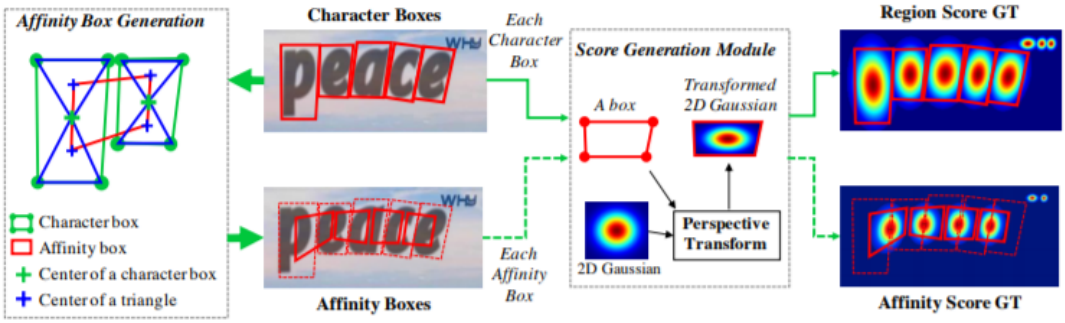

Figure 5: Illustration of ground truth generation procedure in the framework. The generated ground truth labels from a synthetic image that has character level annotations. Image taken from [20].

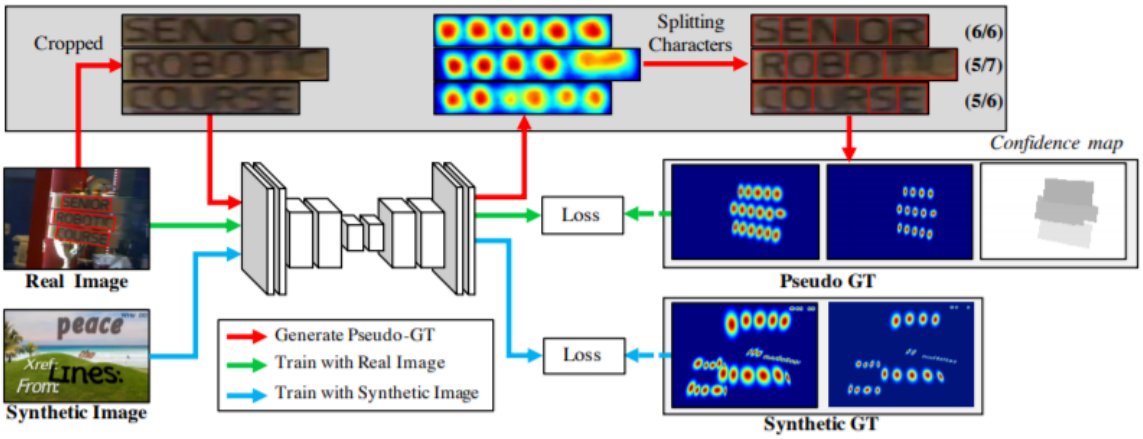

Figure 6: Illustration of the overall training stream for the proposed method. Training is carried out using both real and synthetic images in a weakly-supervised fashion [30].

2.2. Text Recognition

Text recognition and reading capability takes an important place in computer vision tasks. Many practical applications, i.e. hand written texts, tag reading and traffic sign reading requires text recognition. With the development of text detection models[31; 32; 33; 34], text recognition has started to take a very significant place in front of research subjects. Text recognition research is still open and hot topic in computer vision.

Recently, some text recognition methods[35; 36; 37; 38; 39], have achieved significant success. Reading text is naturally considered as multiclassification problem that contains squence-like objects[38]. These models and methods are usually based on the Convolutional Neural network[40; 9], but also require the integration of Recurrent neural network[41; 38; 32] and attention mechanism[42; 43; 44; 45].

In addition, the text recognition models are generally tested on benchmark datasets such as IIIT5K-Words (IIIT5K)[46], Street View Text (SVT)[47], ICDAR 2003 (IC03)[48], ICDAR 2013 (IC13)[13], SVT-Perspective (SVT-P)[49] and their performance is compared.7

Although these models have had great success, various shapes and distorted patterns of the irregular texts give this field a new challenge. Scene texts with irregular shapes, such as perspective and curved texts are still very difficult to recognise, shown in Fig. 7. In the paper "A Multi-Object Rectified Attention Network for Scene Text Recognition"[39] published in 2019 and the authors propose a method for solution to this problem.

2.2.1. MORAN : A Multi-Object Rectified Attention Network for Scene Text Recognition

MORAN[39] is a multi-object rectified attention network that can read curved text, various shapes and stretched characters from different background of images. This model includes two parts, multi-object rectification network (MORN) and an attention-based sequence recognition network (ASRN), shown in Fig. 8. While the MORN part is for image rectification, the ASRN part is for reading the text.

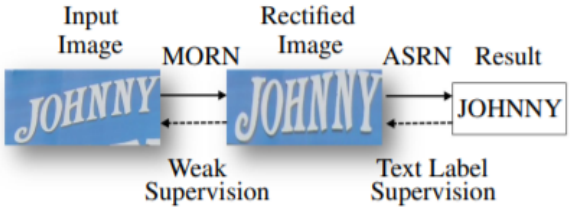

Figure 7: Example for scence text with irregular shape. [39].

Multi-object Rectification Network(MORN): MORN is trained under weak supervision to understand the offset of each part of the image. Rectified text image is produced by predicting the offset. MORN does not predict character categories when predicting position offsets.

The advantages of the MORN method can be summarized as follows:

- MORN gives more readable corrected text resulting in a normal form of the text image with less noise.
- Usually affine transformation methods are used to rectify images, however, these methods are geometrically limited. MORN is more flexible than affine transformation.It has no geometric boundaries and arranges images using complex transformations.
- While the MORN method does not require extra labeling for character positions, it can be trained on existing training data through weak-supervison.

Attention-based Sequence Recognition Network(ASRN): ASRN architecture has attention mechanism with CNN-RNN, shown in Fig. 8. In the model BLSTM stands for bidirectional LSTM[50]. Bidirectional LSTM is a network that consists of two independent RNNs and uses two different inputs at each stage, one from the

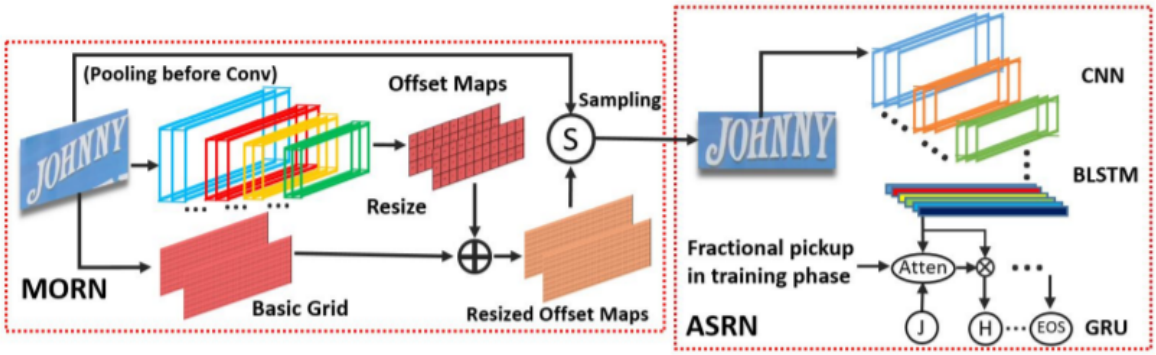

Figure 8: MORAN architecture [39].

past and the future. In this way the network preserves information from both past and the other for the future. In the last part of the ASRN, the GRU[51; 52] makes word predictions and the model selects the word with the highest probability

Basically, ASRN decoder learns the relationship between tags and target characters. It is a data-based process and the ability to select the right regions is enhanced by the feedback of correct alignment.

2.2.2. Image-based Sequence Recognition and Its Application to Scene Text Recognition

Text recognition is directly related to image-based sequence recognition. In[38], a neural network architecture is proposed that integrates feature extraction, sequence modeling and transcription into a unified framework. The advantages of the model proposed in this paper can be listed as follows:

- The network can be trained end to end, unlike most existing algorithms whose components are individually trained and tuned.
- It processes sequences naturally from random lengths, without characters segmentation or horizontal scale normalization.
- The model is effective and small, which is more practical for real application scenarios.

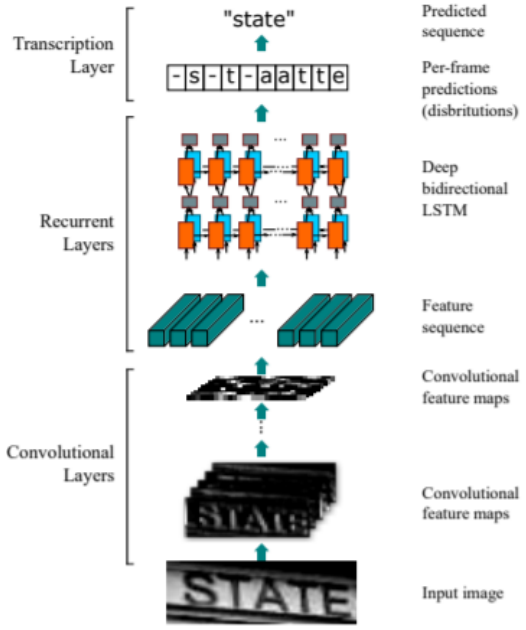

Figure 9: The network architecture[38].

The CRNN model proposed in the article consists of three parts, shown in Fig. 9. These are Convolutional layers, the Recurrent layers and transcription layer parts.While the Convolutional layer part extracts features from each given image, the Recurrent Neural Network part makes predictions for each frame. The transaction layer in the last part converts the prediction made for each frame into a label sequence. As a result, CRNN[38] is formed by training CNN and RNN together with a single loss function.

Feature Sequence Extraction: In this part, a sequence of feature vectors is extracted from fea-

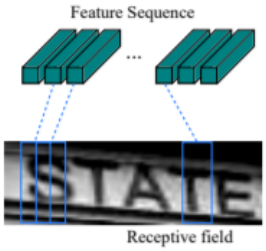

Figure 10: The receptive field. Each vector in the extracted feature sequence is associated with a receptive field on the input image, and can be considered as the feature vector of that field [38].

ture maps produced by the Convolutional layers component. Each feature vector array is created by generating a feature, from left to right, on feature maps. Each column on the feature map is associated with a rectangle on the original image, and the rectangular regions(receptive field) are in the same row from left to right, shown in Fig. 10.

Sequence Labeling: This part is created by combining bidirectional Recurrent neural Networks, thus prediciting label distribution for each frame in the feature sequence.

Transcription: Transformation of RNN predictions into label-sequences for each frame is a process of finding a transcription. In other words transcription is a process of finding the highest probability label sequence which is conditioned on the per-frame predictions. Connectionist Temporal Classification (CTC)[26] is used to calculate the loss function based on that conditional probability. Experiments on benchmark datasets have been observed that the CRNN model proposed in the article gives better results than other CNN and RNN based models[53; 9].

3. Methodology

To develop an Optical Character Recognition system that can read traffic signs with high performance, firstly the analyzed models should be tested and compared with the proposed model. Therefore, in this research, a special dataset consisting of real traffic signs was prepared and tested with past models and the proposed model.

Mapillary is the street-level imagery platform that scales and automates mapping using collaboration, cameras, and computer vision. Contributors of the Mapillary want to represent the whole world with photos. Mapillary connects to the OpenStreetMap platform that everyone can contribute and upload images from all around the world.

Traffic sign recognition is one of the crucial subjects that matters for autonomous driving and smart cities. By using the advantage of computer vision algorithm that detects traffic signs and classify automatically from street level images, Mapillary extracted traffic signs[54] and introduced the worlds largest and most diverse traffic sign dataset.

Therefore, when preparing data sets, The OpenStreetMap platform offered by Mapillary was used.

3.1. Datasets

First, 5 different regions were selected from United: Los Angeles, Austin, Washington, Hayward and Detroit. As mentioned earlier, Mapillary has already computer vision algorithm that detects traffic signs and classify from street view with its map features.

The traffic signs belonging to the classes containing text from the traffic sign objects in the regions were selected by using Mapillary API. Then the images were downloaded in those areas where these objects were visible. Then, using the keys of the images that were downloaded, the traffic sign coordinates were determined in each image. Finally, those coordinates were cropped from each image, shown in Fig. 11.

In this way, different numbers of traffic signs containing text for each region were obtained.

At this point we have labelled them manually according to the quality of the photos and the text they contain.

Figure 11: Example from the street view of traffic signs.

For the quality of traffic sign we determined three classes as high, ok and low, shown in Fig. 12.

Many OCR systems as described in the relative work section were divided into two parts. One is detection of the text spot and the other one is text recognition. Therefore, two different training data sets are needed, one for detection and the other for recognition.

3.1.1. Text Spotting Train Dataset

To be able to train with OCR models, bounding box coordinates containing text in images provided as input must be known, and what is written in it. But in real life it is hard to annotate thousands of images. However, this problem can be solved by preparing a synthetic data set.

Two types of data sets were prepared for text perception training.

The initial training data includes a total of 600 actual traffic signs, and these traffic signs were taken from the Chicago area, shown in Fig. 13.

Second, an annotation was added to around 200 traffic sign templates to create a synthetic dataset. It is essential to simulate real traffic signs to create a synthetic dataset.

Each image was rotated in 9 different angles [5,10, ..., 45] both right and left.

After the traffic signs were rotated, the changing corners of the boxes were calculated. Boxes were recalculated using the formula below:

$$X = x \times \cos\theta - y \times \sin\theta \tag{3}$$

$$Y = x \times \sin\theta + y \times \cos\theta \tag{4}$$

Additionally, cropped images from real street images were added to the background of rotated synthetic signs and also random blur was added, shown in Fig. 14

In this way, 7000 synthetic datasets were created for text detection training.

3.1.2. Text Recognition Train Dataset

For text recognition training data, only need small images with short text groups. Therefore, half of the text recognition model's training data was taken from actual traffic signs that were annotated manually. Bounding box coordinates containing already annotated text from actual traffic signs, shown in Fig. 15 are truncated.

The other half of the recognition data set was synthetically created by using TextRecognitionDataGenerator[55] tool.

Region	Low quality	Ok quality	High quality	Unique Signs	Total detections
Los Angeles	1132	575	531	583	2240
Austin	689	456	612	692	1842
Detroit	753	546	518	756	1821
Hayward	2086	753	1213	993	4055
Washington	1535	536	780	884	2851

Figure 12: Parking sign benchmark dataset statistics.


```
34,17,3,21,7,51,38,47,NO
77,43,7,56,12,83,82,70,STOPPING
43,80,18,84,21,104,47,99,ANY
73,73,47,77,50,97,76,94,TIME
```

Figure 13: Example for annotation of real traffic sign crop.

Figure 16: Examples from synthetic Text Recognition Dataset

Figure 14: Synthetic dataset generation

Figure 15: Real Text Recognition Datasetn

In U.S.A. traffic signs have usually different versions of Highway Gothic fonts. In addition, a dictionary of words that can be included in traffic signs has been prepared. Synthetic recognition data was generated using this dictionary and 5 different Highway-Gothic fonts[56] in that tool[55], shown in Fig. 16

3.2. Proposed method

Text Detection Part: We found the FOTS[5] model suitable for the detection part because the FOTS model has better performance than the EAST[2] model and also it only requires word-level bounding box labelled data. In fact, when looking at the reviews in the papers, it will be seen that, the CRAFT[20] model works better than FOTS, but the CRAFT model needs character-level annotation or needs to be trained weakly-supervised with SynthText[27] dataset which needs a lot of computational power. Also, there is no big difference between the results of the FOTS model and the CRAFT model. Papers can be examined for detailed results[5; 20].

This[57] application was used to train the tagged parking sign dataset built with the FOTS model. However, as can be expected, this implementation was made for ICDAR 2015[12] dataset, so some modifications had been made.

First the long portions of the pictures were fixed at 768 pixels and short pieces were enlarged in the same proportion. Then the pictures are rotated between [-5.5] at random. When rescaling the height of the images from 0.5 to 3.0 were left unchanged in width. After that, a random sample of 192x192 was taken from transformed images.

While the FOTS model was being trained, it was used as a bottleneck for ResNet-50 features for parking sign pictures.

While the data training was in progress, RIO tresholding was applied and the NMS method was applied to the predicted boxes. These predicted boxes were given to the recognition section to get the recognition results. For multi-scale testing, all results scales are combined and sent back to NMS to get the final results.

Text Recognition Part: We were inspired by the study in[38] paper for the text recognition part of the parking sign OCR system that we were trying to create. The reason we chose [38] was that it works very efficiently, and it is easy to implement. We created our model in three basic parts, as described in the article. Our model consists of a combination of Convolutional Neural network[58], Recurrent Neural network[44] and Connectionist Temporal Classification[26] loss.

$$MODEL = CNN + RNN + CTC_{loss} \quad (5)$$

As a result, our model consists of three different parts:

1. Convolutional Neural Network part helps to extract features from images.
2. Recurrent neural network part in order to predict sequential output per time step.
3. Connectionist Temporal Classification(CTC) loss function, the transcription layer used to predict the output for each time step.

While creating the architecture, we followed the paths below:

1. In the architecture created, the height of the input images was determined as 32 and the width as 128..
2. We used seven convolutional layers of which six are having kernel size (3,3) and the last one is of size (2,2), whereas the number of filters is increased from 64 to 512 layer by layer.
3. Two max-pooling layers are added with size (2,2) and then two max-pooling layers of size (2,1) are added to extract features with a larger width to predict long texts.
4. We also used the batch normalization layers after the fifth and sixth layers of convolution that speeded up the training process.
5. Next, we used a lambda function to squeeze the output from the Convolutional layer and make it compatible with the LSTM layer.
6. Then, two bidirectional LSTM layers with 128 units each were used. This RNN layer outputs the dimension (batch-size, 31, 63). 63 is the total number of output classes, including null characters.

Loss of Connectionist Temporal Classification(CTC)[26] is very helpful in text recognition. It helps us avoid adding notes at any time step and avoids the problem that a single character can span multiple times, which requires more processing if CTC is not used.

A CTC loss function requires four arguments to calculate the loss, predicted outputs, ground truth labels, the sequence length of the LSTM, and the ground truth label length. To achieve this, a special loss function must be created and added to this model. To combine it with the model, a model was created that takes these four inputs and subtracts the loss.

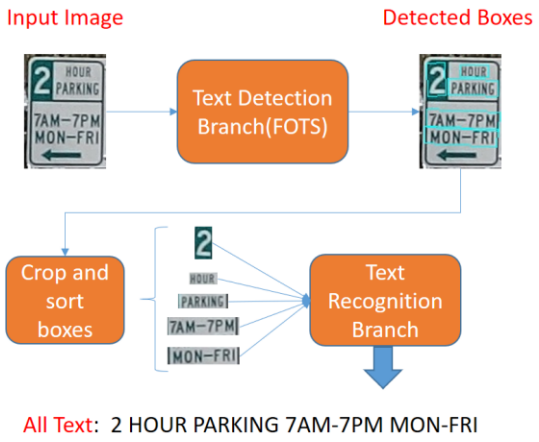

Figure 17: Shematic representation of parking sign OCR

The schematic visual of the system established is shown in Fig. 17.

4. Experimental results and discussions

4.1. Evaluation Methods and Evaluated Models

When evaluating the results, we determined two kinds of seperate terminologies. We defined Recall and Precision to evaluate OCR results:

C.T. (correct text): the number of signs that read as correct.

- **Recall:** The number of signs for which an OCR result was returned.
- **Precision:** $\frac{C.T.}{Recall}$

Recall was defined as the number of signs for which an OCR result was returned. In other words, If the OCR system returns text from a photo given to it, then the recall increased with one.

We defined the Precision as a rate that shows us the percentage of correct texts from the returned texts on the given OCR systems.

Levenshtein and Generalized Jaccard similarity used to determine if a returned text is correct or not.

The Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance[83, 84, 85, 86] between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. In evaluations Levenshtein similarity measure was used, representing the normalized levenshtein similarity score between two strings. For example, to convert the word "sample" to "examples", 3 adjustments are required and the last word as "example" contains 7 letters, then the levenshtein similarity is calculated as 1 - 3/7 = 0.57.

The jaccard similarity for two sets is the intersection of two sets divided by two sets. For instance the set ["data","science"] and ["data","management"] has 1 element in their intersection and 3 elements in the union therefore the jaccard similarity between two sets is 1/3=0.33333

- **Jaccard (X,Y) :** $\frac{|X \cap Y|}{|X \cup Y|}$

Generalized Jaccard similarity measure is a softened version of the Jaccard measure. The Jaccard measure is a promising candidate for tokens which exactly matches across the sets. However, in practice tokens are often misspelled, such as energy vs. The generalized Jaccard measure will enable matching in such cases. Jaro similarity and threshold used inside the Generalized Jaccard function.

The Jaro Similarity is calculated using the following formula:

- **Jaro Similarity**:

$$\begin{cases} 0, \text{ if } m = 0 \\ \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right), for: \end{cases}$$

where;

- **m** is the number of matching characters
- **t** is half the number of transpositions
- where $|s_1|$ and $|s_2|$ is the length of strings s_1 and s_2

Characters are matching;

- if they are the same,

and;

- the characters are not farther than,

$$\left\lfloor \frac{max(|s_1|, |s_2|)}{2} \right\rfloor$$

Therefore, the Generalized Similarity function has two parameters: one is the similarity function is determined as jaro and threshold value. If the similarity of a token pair exceeds the threshold, then the token pair is considered a match.

Therefore, in order to evaluate OCR results, these two measures of similarity were used as Generalized Jaccard and levenshtein.

Threshold values were determined for each similarity:

- Levensthein similarity treshold : **0.75**
- Generalized Jaccard similarity treshold : **0.80**

As a result, if the returned text from the OCR system passed these thresholds compared to the actual text, then we marked it as correctly labelled. In order to evaluate the results in more detail and to better understand and analyze the system, we developed, and evaluated each category(image quality as low, ok, high) separately.

In addition, we evaluated the photos with a resolution greater than 100 x 70 in a separate plot and the images where the traffic signs were detected at the best resolution.

Also, a resolution analysis was made for each region separately.

We used pretrained text detection and text recognition methods from different systems to evaluate the results.

We used the results of the following nine different systems:

- **Google**: The Google Cloud Vision API used directly to review the text results from Google's OCR system. One can get the JSON file that includes the entire extracted string, as well as individual words, and their bounding boxes from images by sending the images to the API as base64 encoded string. For more information, see the Google documentation.[59]
- **Amazon**: The Amazon Rekognition API used for this operation. It has similar logic with Google Cloud Vision API. To use this engine, it must be provided as base64-encoded byte-sequence photos or as an image stored in an Amazon S3. This engine returns JSON file that includes the following items:
  - The detected text (DetectedText).
  - The location of text on the image (Geometry).
  - The confidence Amazon Rekognition has in the accuracy of the detected text and bounding box (Confidence).
  - The type of the detected text (Type).

  For more information, see the documentation.[60].
- **Tesseract**: Tesseract is a text recognition engine offered by Google as a free software since 2006. It is working on different operating systems(Windows,Linux etc.). Tesseract supports a wide variety of languages. It is an open source software that can directly download and use for recognition. For more information, see the Tesseract documentation[61].
- **Tesseract-det**: Tesseract is normally for reading text inline, so it gives much better

results if used with a text detector. Because, in order for the tesseract to be feed, the text field should be determined from any image and then cut and give to the tesseract.

For this reason, a pre-trained model was taken with the EAST(An Efficient and Accurate Scene Text Detector)[62] model previously trained on ICDAR 2013 (training set) + ICDAR 2015 (training set), and the results were analyzed by combining it with tesseract.

- **Tesseract-craft**: The same method described and used above was used. However, this time, CRAFT-trained[20] model was used instead of EAST model.

- **Fots-tesseract** : The FOTS model was trained using previously created synthetic and real traffic sign data sets. Later, the detection part of the FOTS model was combined with the tesseract.

- **Fots-moran** : The detection part of the FOTS model was used again, but this time pre-trained MORAN(A Multi-Object Rectified Attention Network for Scene Text Recognition)[63] in the text recognition section was used.

- **Fots-our** : Here, the FOTS cite 8578693 detection model used above was used. However, this time proposed text recognition model was trained with the previously created text recognition data set. While creating the model, the article "End-to-End Trainable Neural Network and Scene Text Recognition Application for Image Based Sequence Recognition" was used.[38].

- **Fots-our-new** : This part is actually not much different from Fots, which is the model described above, only some problems in the system were analyzed and some improvements were added.

4.2. Training Process

We separated training for text detection and recognition parts. We used the FOTS[5] model for detection training. We used An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition modeling for recognition training.

For text detection training, we used the FOTS model implemented in PyTorch. We allocated 10% of the dataset prepared for detection for validation. We used Res-Net50 features as a backbone to our images. We used One Tesla P100 GPU was used for training and it took 20 hours for 40 epochs; also, we used batch size as 96 with learning rate 0.0001 and Adam optimizer. For each epoch, we saved the checkpoints, and then used them for evaluating our results.

Data augmentation is important for robustness of deep neural networks, especially when the number of real data is limited, as in this case.

Therefore, we tried to increase the dataset we prepared for recognition training with a certain logic. Firstly, we fixed the width of each recognition photo to 32 and at with the same rate, changed the height of each photo. We used four different methods to resize each photo: a nearest-neighbor interpolation, a bi-linear interpolation, resampling using pixel area relation, and bi-cubic interpolation over 4X4 pixel neighborhood. These methods were obtained from python library as OpenCV. Detailed information is available in the documentation[64]. Then, each photo was shrunk in 12 different proportions again using these four methods. The reason is that the size of the picture we get when cutting it after detecting the writing area in real traffic signs samples can be very different. So, we have followed such a way to simulate this situation.

Before starting Recognition training, preprocess has been done to input images and labels. Each image read and convert into a gray-scale image, and make each image of size (128,32) by using padding. Then image dimension was ex-

panded as (128,32,1) to make it compatible with the input shape of architecture. Finally, normalization was also made with the image pixel values by dividing it with 255. To preprocess the output, labels were encoded each character of a word into some numerical value by creating a function(as 'a':0, 'b':1 . . . . 'z':26 etc). Let say we are having the word 'abab' then the encoded label would be [0,1,0,1].

Two other lists were created as one is label length and the other for input length to RNN(Recurrent Neural Network). These two lists are important for CTC(Connectionist Temporal Classification) loss.

We allocated 10% of the dataset prepared for recognition for validation, which means that we trained on 240128 samples and validated on 26680 samples.

After all these preprocesses, we trained the text recognition model with 50 epochs and 256 batch size on one tesla P100 GPU.

We used Adam optimizer with the learning rate 0.001. After each epoch, the weights of the best model were saved on the basis of validation loss.

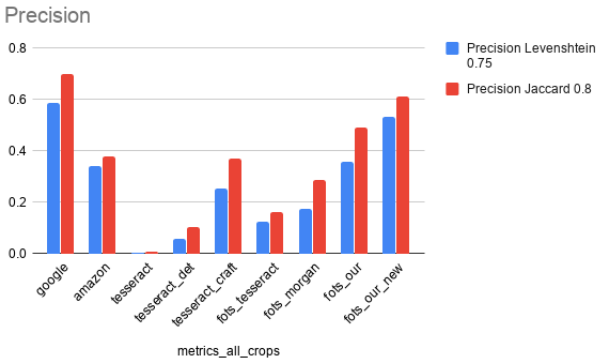

Figure 18: Precision Scores from all Hayward traffic sign detections

4.3. Results

Six different plots for five different regions were created. In this section, we present the results belonging to Hayward but other results are supplied in Appendix, A

First, all the traffic sign images belonging to the Hayward region were evaluated18.

The recall results were evaluated separately from all Hayward traffic sign detections, shown in Fig. 19

Based on the results, it was revealed that our system was able to find a detection box for almost every traffic sign. But it turned out that either the recognition part could not read the bounding boxes or some of the bounding boxes returned from our system were turning completely wrong.

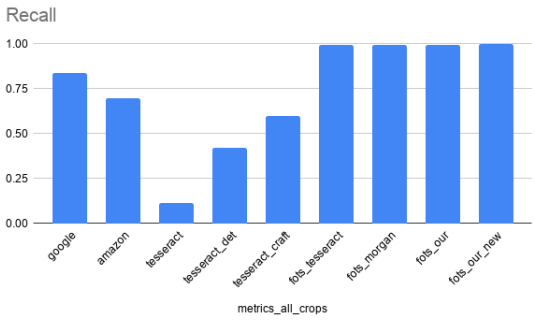

Figure 19: Recall scores from the Hayward Region

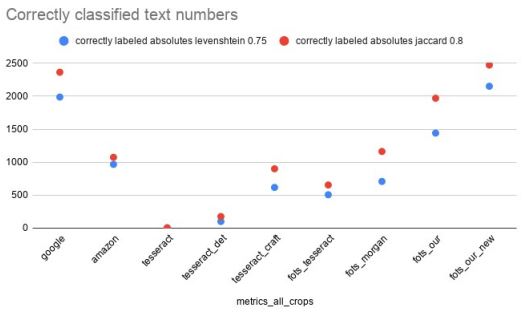

Figure 20: The number of the correctly classified texts

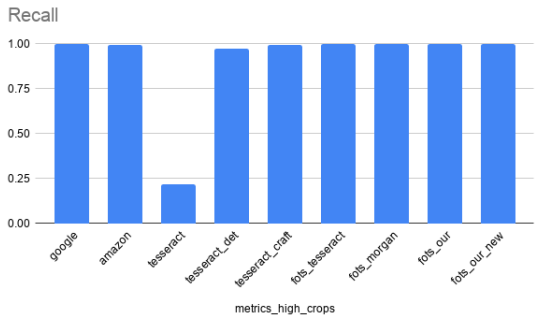

Figure 22: Recall of high-quality images of the Hayward region

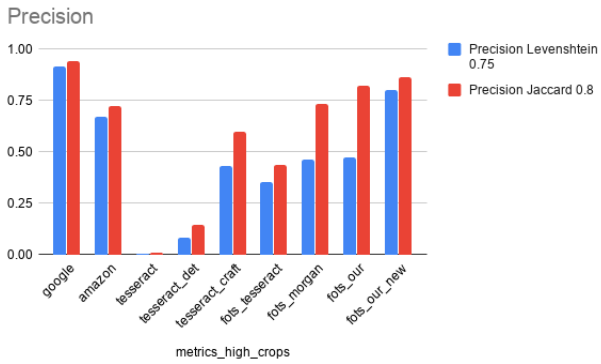

Figure 21: Results of high-quality photos of the Hayward region

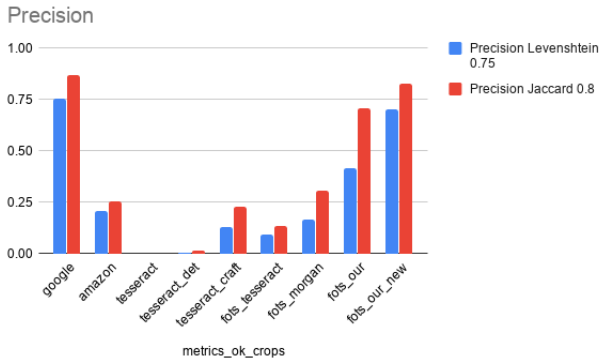

Figure 23: Results of ok-quality photos of the Hayward region

In order to better understand this situation, it is useful to examine the numbers that appear directly, shown in Fig. 20. When looking at the correct number of classified text numbers, it turns out that we have classified more text from Google correctly. However, because of the high recall, our precision appears lower.

If we take only the high quality images from the Hayward region, shown in Fig. 21, we are almost Google level precision; on the other hand, if we check the recall plot, shown in Fig. 22 it turns out that we are at the same level with Google, i.e., there is little difference between Google and our system but still Google is better than ours.

If we only take ok quality images and evaluate them, we will see that the gap between Google and us is less, shown in Fig. 23,24.

On the other hand, when only the low quality images are examined, the system we produced is working much better(according to Levenshtein metric), shown in Fig. 25,26. This special case can be explained as follows: normally, while Google is producing the OCR results, and the photo quality is very low, it does not return text because it is not sure about the box containing the text(Low Confidence Level). However, since our system is only trained on traffic signs, it tries to find boxes that contain text even in very low quality photographs and at least a certain part of it is correct. Besides, the recognition

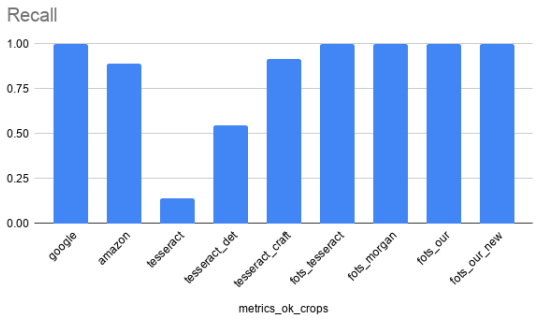

Figure 24: Recall of ok-quality photos of the Hayward region

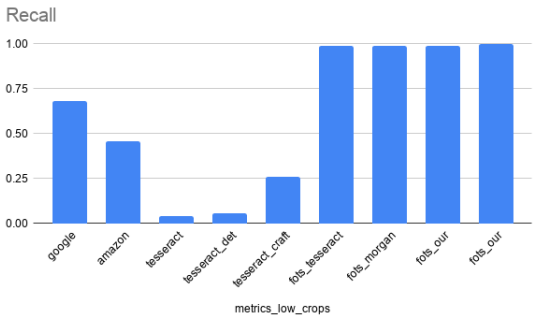

Figure 26: Recall of low-quality photos of the Hayward region

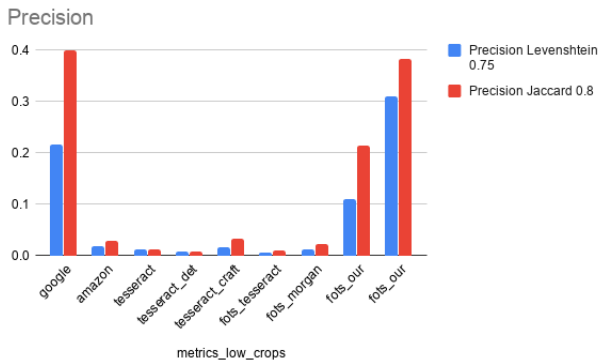

Figure 25: Results of low-quality photos of the Hayward region

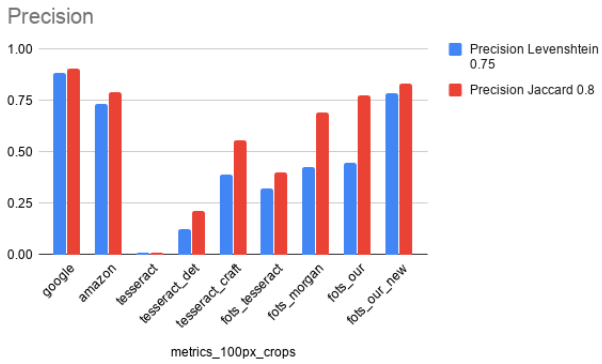

Figure 27: Precision scores for resolution greater than 100px

system we have trained can read even in difficult situations, especially since it sees even very low quality writing sets beforehand.

Finally, we only analyzed using traffic sign photos with a resolution greater than 100px, shown in Fig. 27,28.

As can be understood from the qualitative results, our system generally performs better than Google in low quality photos. Actually this is very important and good, and we still have space to improve our system, and in difficult cases, we can already do better than Google

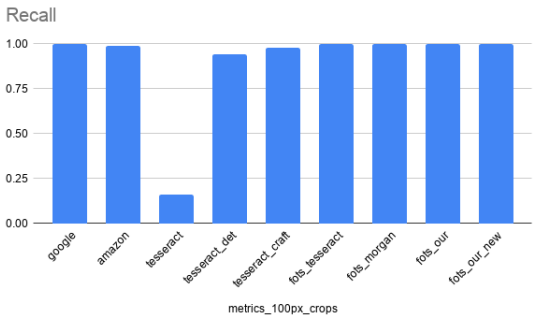

Figure 28: Recall scores for resolution greater than 100px

5. Conclusion

In the current study, we defined a benchmark dataset representative for understanding the park- ing signs, and made attempts to produce a special OCR for parking signs. We outperformed open source methodologies like tesseract. Our system showed an equal or better performance than commercial cloud based systems(Google, Amazon etc.) on this task. Our system worked particularly well on very low quality images, which is the more diffi- cult case for OCR systems. However, many aspects need to be improved. At the same time, this study demonstrates and proves that instead of general trained OCR systems, systems designed and developed for special use-cases can be cre- ated, and they can work much better than general systems.

As a future work, real data annotation can be increased and more realistic synthetic datasets can be developed. Because of our limited real dataset, the system could not understand some special cases very well and the synthetic dataset was not sufficient to simulate a real dataset. For example, by adding things like shadow, special noise, perspective transform in the synthetic dataset, a more realistic simulation can be created

In addition, by using a different and special data augmentation method, the Neural Network can acquire sufficient boost in detection and recognition abilities.

Acknowledgments

I would first like to thank my thesis supervisors Till Quack and Prof. Gholamreza Anbarjafari (Shahab), who inspired me to write this work and who helped me a lot with finishing it. Also, I would like to express my gratitude to my mates from Mapillary company for they took me in their awesome family and navigated me at all aspects of my internship and master thesis.

References

[1] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. *CoRR*, abs/1609.03605, 2016.

[2] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. EAST: an efficient and accurate scene text detector. *CoRR*, abs/1704.03155, 2017.

[3] Wenhao He, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Deep direct regression for multi-oriented scene text detection. *CoRR*, abs/1703.08289, 2017.

[4] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu. Textboxes: A fast text detector with a single deep neural network. *CoRR*, abs/1611.06779, 2016.

[5] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan. Fots: Fast oriented text spotting with a unified network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5676–5685, 2018.

[6] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. *CoRR*, abs/1807.01544, 2018.

[7] Pengyuan Lyu, Cong Yao, Wenhao Wu, Shuicheng Yan, and Xiang Bai. Multi-oriented scene text detection via corner localization and region segmentation. *CoRR*, abs/1802.08948, 2018.

[8] Minghui Liao, Baoguang Shi, and Xiang Bai. Textboxes++: A single-shot oriented scene text detector. *CoRR*, abs/1801.02765, 2018.

[9] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *CoRR*, abs/1412.1842, 2014.

[10] M. Buta, L. Neumann, and J. Matas. Fastext: Efficient unconstrained scene text detector. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1206–1214, 2015.

[11] Shangxuan Tian, Yifeng Pan, Chang Huang, Shijian Lu, Kai Yu, and Chew Lim Tan. Text flow: A unified text detection system in natural scene images. *CoRR*, abs/1604.06877, 2016.

[12] Xinyu Zhou, Shuchang Zhou, Cong Yao, Zhimin Cao, and Qi Yin. ICDAR 2015 text reading in the wild competition. *CoRR*, abs/1506.03184, 2015.

[13] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i. Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazàn, and L. P. de las Heras. Icdar 2013 robust reading competition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1484–1493, 2013.

[14] A. Shahab, F. Shafait, and A. Dengel. Icdar 2011 robust reading competition challenge 2: Reading text in scene images. In *2011 International Conference on Document Analysis and Recognition*, pages 1491–1496, 2011.

[15] O. Zayene, J. Hennebert, R. Ingold, and N. E. BenAmara. Icdar2017 competition on arabic text detection and recognition in multi-resolution video frames. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1460–1465, 2017.

[16] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1083–1090, 2012.

[17] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2963–2970, 2010.

[18] L. Neumann and J. Matas. Text localization in real-world images using efficiently pruned exhaustive search. In *2011 International Conference on Document Analysis and Recognition*, pages 687–691, 2011.

[19] W. Huang, Z. Lin, J. Yang, and J. Wang. Text localization in natural images using stroke feature transform and text covariance descriptors. In *2013 IEEE International Conference on Computer Vision*, pages 1241–1248, 2013.

[20] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. *CoRR*, abs/1904.01941, 2019.

[21] J. Hosang, R. Benenson, and B. Schiele. Learning non-maximum suppression. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6469–6477, 2017.

[22] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855, 2006.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[24] Kye-Hyeon Kim, Yeongjae Cheon, Sanghoon Hong, Byung-Seok Roh, and Minje Park. PVANET: deep but lightweight neural networks for real-time object detection. *CoRR*, abs/1608.08021, 2016.

[25] K. Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[26] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA, 2006. Association for Computing Machinery.

[27] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. *CoRR*, abs/1604.06646, 2016.

[28] C. Clausner, A. Antonacopoulos, and S. Pletschacher. Icdar2017 competition on recognition of documents with complex layouts - rdcl2017. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1404–1410, 2017.

[29] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[30] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9365–9374, 2019.

[31] Lluis Gomez-Bigorda and Dimosthenis Karatzas. Textproposals: a text-specific selective search algorithm for word spotting in the wild. *CoRR*, abs/1604.02619, 2016.

[32] L. Sun, Q. Huo, W. Jia, and K. Chen. Robust text detection in natural scene images by generalized color-enhanced contrasting extremal region and neural networks. In *2014 22nd International Conference on Pattern Recognition*, pages 2715–2720, 2014.

[33] Anna Zhu, Renwu Gao, and Seiichi Uchida. Could scene context be beneficial for scene text detection? *Pattern Recogn.*, 58(C):204–215, October 2016.

[34] Vijeta Khare, Palaiahnakote Shivakumara, Paramesran Raveendran, and Michael Blumenstein. A blind deconvolution model for scene text detection and recognition in video. *Pattern Recognition*, 54:128 – 148, 2016.

[35] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *2013 IEEE International Conference on Computer Vision*, pages 785–792, 2013.

[36] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3538–3545, 2012.

[37] Bolan Su and Shijian Lu. Accurate recognition of words in scenes without character segmentation using recurrent neural network. *Pattern Recognition*, 63:397 – 405, 2017.

[38] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *CoRR*, abs/1507.05717, 2015.

[39] Canjie Luo, Lianwen Jin, and Zenghui Sun. A multi-object rectified attention network for scene text recognition. *CoRR*, abs/1901.03003, 2019.

[40] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3304–3308, 2012.

[41] Pan He, Weilin Huang, Yu Qiao, Chen Change Loy, and Xiaoou Tang. Reading scene text in deep convolutional sequences. *CoRR*, abs/1506.04395, 2015.

[42] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. Focusing attention: Towards accurate text recognition in natural images. *CoRR*, abs/1709.02054, 2017.

[43] Zhanzhan Cheng, Xuyang Liu, Fan Bai, Yi Niu, Shiliang Pu, and Shuigeng Zhou. Arbitrarily-oriented text recognition. *CoRR*, abs/1711.04226, 2017.

[44] Chen-Yu Lee and Simon Osindero. Recursive recurrent nets with attention modeling for OCR in the wild. *CoRR*, abs/1603.03101, 2016.

[45] Xiao Yang, Dafang He, Zihan Zhou, Daniel Kifer, and C. Lee Giles. Learning to read irregular text with attention mechanisms. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 3280–3286. AAAI Press, 2017.

[46] Karteek Alahari and C. Jawahar. Scene text recognition using higher order language priors. 09 2012.

[47] Kai Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *2011 International Conference on Computer Vision*, pages 1457–1464, 2011.

[48] Proceedings seventh international conference on document analysis and recognition. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, 2003.

[49] T. Q. Phan, P. Shivakumara, S. Tian, and C. L. Tan. Recognizing text with perspective distortion in natural scenes. In *2013 IEEE International Conference on Computer Vision*, pages 569–576, 2013.

[50] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.

[51] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[52] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ArXiv*, 1409, 09 2014.

[53] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Deep features for text spotting. In *European Conference on Computer Vision*, 2014.

[54] Christian Ertler, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, Gerhard Neuhold, and Yubin Kuang. The mapillary traffic sign dataset for detection and classification on a global scale. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII*, volume 12368 of *Lecture Notes in Computer Science*, pages 68–84. Springer, 2020.

[55] Edouard Belval. Textrecognitiondatagenerator.

[56] DaFont. Fonts.

[57] Ning Lu. Fots.pytorch.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[59] Google. text recognition from image.

[60] Amazon. text recognition from image.

[61] Google. text recognition from image.

[62] Xinyu Zhou. A tensorflow implementation of east text detector.

[63] Canjie-Luo. Moran: A multi-object rectified attention network for scene text recognition.

[64] Open Source Computer Vision. Opencv.

Appendix A. Appendix