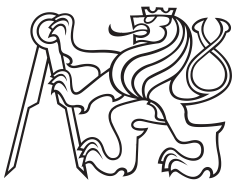
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Learning with Weak Annotations for Text in the Wild Detection and Recognition

Klára Janoušková

Supervisor: Prof. Ing. Jiří Matas, Ph.D.
Field of study: Open Informatics
Subfield: Computer and Information Science
May 2020

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Janoušková Klára** Personal ID number: **474386**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Learning with Weak Annotations for Text in the Wild Detection and Recognition

Bachelor's thesis title in Czech:

Učení s neúplnou informací pro detekci a rozpoznávání textu v obrazech

Guidelines:

1. Review possible sources of weakly annotated data usable for in the wild text detection and recognition, such as Google maps, product databases, ….. Download suitable datasets.
2. Develop a method for localising the weak annotations, typically in the form of a set of text strings, using a state of the art text detector and recogniser, e.g. TextSnake [1].
3. Using the matched detected text and the annotation, retrain the recognizer.
4. Propose a method for estimating location of the matched text for the case when it differs from the detected case.
5. Retrain the detector.
6. Evaluate the improvement of the text detection and recognition system achieved, using standard benchmarks [2].

Bibliography / sources:

[1] Long, S., Ruan, J., Zhang, W., He, X., Wu, W., & Yao, C. (2018). Textsnake: A flexible representation for detecting text of arbitrary shapes. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 20-36.
[2] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. Ramaseshan Chandrasekhar, S. Lu, F. Shafait, S. Uchida, E. Valveny, "ICDAR 2015 Competition on Robust Reading", In Proc. 13th International Conference on Document Analysis and Recognition (ICDAR 2015), IEEE, 2015, pp. 1156-1160.

Name and workplace of bachelor's thesis supervisor:

prof. Ing. Jiří Matas, Ph.D., Visual Recognition Group, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **03.01.2020** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

prof. Ing. Jiří Matas, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

First of all, I would like to thank my supervisor, prof. Ing. Jiří Matas, Ph.D., for patiently guiding me through the whole process.

The thesis builds on top of work done in collaboration with Dr. Dimosthenis Karatzas, Ing. Michal Bušta and Lluis Gomez, Ph.D., whom I would like to thank for all the insightful discussions and suggestions to improve the text. I am also grateful to all the other people from CMP for being a great inspiration and patiently answering all my technical questions.

Last but not least, I thank my friends (including you, Tom), family and my fluffy little friend Teddy for their unfailing support.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 22 May 2020

Abstract

In this work, we present a method for exploiting weakly annotated images to improve text extraction pipelines. The weak annotation of an image is a list of texts that are likely to appear in the image without any information about the location. An arbitrary existing end-to-end text recognition system is used to obtain text region proposals and their, possibly erroneous, transcriptions. A process that includes imprecise transcription to annotation matching and edit distance guided neighbourhood search produces nearly error-free, localised instances of scene text, which we treat as "pseudo ground truth" used for training.

We apply the method to two weakly-annotated datasets and use the obtained pseudo ground truth to re-train the end-to-end system. The process consistently improves the accuracy of a state of the art recognition model across different benchmark datasets (image domains) as well as providing a significant performance boost on the same dataset, further improving when applied iteratively.

Keywords: text detection and recognition, weakly-supervised learning

Supervisor: Prof. Ing. Jiří Matas, Ph.D.

Abstrakt

V této práci představujeme metodu využívající slabě anotované obrázky pro vylepšení systémů pro extrakci textu. Slabá antoace spočívá v seznamu textů, které se v daném obrázku mohou vyskytovat, ale nevíme kde. Metoda používá libovolný existující systém pro rozpoznávání textu k získání oblastí, kde se pravděpodobně vyskytuje text, spolu s ne nutně správným přepisem. Výsledkem procesu zahrnujícího párování nepřesných přepisů se slabými anotacemi a prohledávání okolí vedené Levenshtein vzdáleností jsou skoro bezchybně lokalizované texty, se kterými dále zacházíme jako s pseudo-anotacemi využívanými k učení.

Aplikování metody na dva slabě anotované datasety a doučení použitého systému pomocí získaných pseudo-anotací ukazuje, že námi navržený proces konzistentně zlepšuje přesnost rozpoznávání na různých datasetech (jiných doménách) běžně využívaných k testování a velmi výrazně zvyšuje přesnost na stejném datasetu. Metodu lze použít iterativně.

Klíčová slova: detetkce a rozpoznávání textu, učení s neúplnou informací

Překlad názvu: Učení s neúplnou informací pro detekci a rozpoznávání textu v obrazech

Contents

Figures

Tables

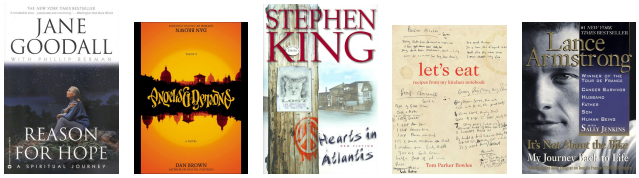
Chapter 1

Introduction

Written text is an important source of information for humans. It comes in many forms. Books and text documents are likely to come to mind as the first thing but it is also frequently a part of outdoor environments. The moment we leave our home, we are surrounded by text - a sign in front of the bakery tells us we can get a free croissant with a coffee, but we do not have the time because we can recognize the number of the bus we need to catch in the distance. In the bus, we subconsciously skim through the advertisments and the moment we get out, we encounter a sign telling us we have to take a different path then usually because of construction work in one of the streets. If we capture some of those texts by a camera, we refer to them as scene text or text in the wild images.

Knowing the text which appears in images can be beneficial to many applications. In order for the text to be usable by the computers, it needs to be converted to a machine-readable format. This task can be seen as a special case of object detection and recognition, text is actually a frequent class in the Common Objects in Context [38] dataset - a large-scale object detection, segmentation, and captioning dataset. Text detection and recognition are two closely related tasks, jointly referred to as end-to-end recognition, end-to-end text spotting or simply end-to-end reading. The goal of text detection is to localize all the text present in an image. The localization usually takes the form of enclosing polygons and can be done at different degrees of granularity - characters, words, lines or larger text regions. Text recognition follows text detection, its goal is to convert the image capturing a text into a sequence of characters. Scene text detection and recognition, unlike for example optical character recognition in documents, brings challenges in form of imperfect imaging conditions (perspective, blur), font variety, complex backgrounds and occlusion. It is a very active research field attracting the attention of both researchers and companies. It is an essential part of many applications ranging from translation systems and autonomous driving to image retrieval or visual question answering.

In the recent years, all state of the art methods have been based on deep neural networks. Deep neural networks require large-scale annotated data



(a) (b) (c) (d) (e)

Figure 1.1: An example source of weakly annotated data - images of book covers with the author and the title of the book as weak annotations. The ABC Dataset images, collected from Amazon books, are diverse, some have (a) both a simple layout and font, (b) a very artistic, almost illegible font and (c) a font that resembles handwriting. Others have (d) both dense background and hand-written text or (e) the background varies significantly even at the word level.

for training and in order to generalize well, such data should be rich in geometry, style and content. Ground truth data is typically defined at the granularity of words and consists of polygonal regions in the image along with the corresponding text transcriptions. The acquisition of such data requires substantial human effort and is very costly.

The lack of human-annotated data is usually mitigated in two different ways, either by generating synthetic data as in [4, 8, 40, 12, 21, 26] or with different forms of weakly, semi or unsupervised learning on real data as in [2, 36, 32].

There is also a large volume of weakly annotated data, i.e. images with a set of words likely to appear in them. This kind of weak annotations, obtained automatically, have not been exploited so far for text detection and recognition. An example source of such weakly annotated data are product databases where we can readily obtain the name of the product and other meta-data. Images from mapping services like Google Maps are another potential source where street names and numbers or business names are words very likely to appear in an image and easily obtainable through location-based search. For example, given images from the location where a restaurant is supposed to be, it is expected that the name of the restaurant will be visible in some of them. Illustration of one such source of weakly annotated data, book covers from Amazon Books used in our experiments, is shown in Figure 1.1.

This work presents a method that uses an existing end-to-end reading system (E2E), pretrained on fully-annotated data, to localize and recognize the text that potentially overlaps with text from the weak labels. The core idea behind our method is that the output of the recognition model can be used to identify the most probable text match from the weak annotations by finding the one with the lowest edit distance. The detections that produce an exact match with the weak label are assumed correct. Furthermore, the recognition output can be used to find the modifications of the detected region that minimize the edit distance to the matched text. For example, if we

have predicted the word 'car' and the best match was 'cartoon', running the recognition again on the detected region extended to the right may decrease the distance between the matched and predicted text, possibly leading to the prediction of the matched word 'cartoon'. The probability of the recognition model giving the same output as the weak label for a wrong text region is very low, thus it is safe to use such regions as ground truth for recognition training. Therefore, we will refer to it as pseudo ground truth (PGT). Furthermore, it may be possible to use the PGT to improve the detector as well. To our knowledge, previous methods for weakly supervised learning used weak labels which alleviate but do not eliminate human participation, such as annotating only the areas of interest or localizing words instead of characters.

In summary, given an image and a dictionary of words as an input, the method outputs a subset of the dictionary words with their corresponding text regions in the image. The method is independent of the underlying implementation of the models used and it is able to handle incomplete (not all the text in the image is in the dictionary) and noisy labels (the dictionary contains words that are not present in the image).

Possible applications of our method are improving the performance of an existing E2E system and domain adaptation, where the source domain has full ground truth data available whereas only weak labels are available for the target domain.

We apply the method to data from two different sources - a database of images of book covers downloaded from Amazon books, using the title and the author of the book as weak annotations, and the Uber-Text dataset [42], which is very similar to the kind of data that could be obtained using a mapping system. We train a recognition model with the PGT generated from both sources on various benchmarks, showing that it consistently improves the recognition accuracy across a wide range of datasets. While the images from the Uber-Text dataset are annotated and applying our method to them does not produce additional value in terms of generated PGT, working with a large-scale annotated dataset allows us to compare the performance of our method to fully supervised training, showing its applicability for domain adaptation. An estimate of the upper bound of the false positive rate of the method is estimated - less than 2 % of the generated PGT is incorrect.

The contributions of the work are:

- A new method for automatically generating pseudo ground truth data from images with weak annotations.
- We show that models trained with the PGT generated from two different sources perform well on a wide range of benchmarks datasets, consistently boosting the accuracy, even when the PGT data originates from a very different distribution.
- Training with PGT improves the recognition performance significantly in weakly-supervised domain adaptation.

- We show that it is possible to reach on-par performance with state of the art methods with no architecture changes and no human-annotated data for recognition training. The model can recognize very difficult texts which are, without context, challenging even for humans.
- The localization of the PGT texts in the Amazon Book Covers dataset, as well as the pretrained models, will be made public at `https://github.com/klarajanouskova/text-detection-recognition-PGT`.

Chapter 2

Related work

Following a short introduction of scene text detection and recognition methods and an overview of methods for generating synthetic data, we focus on weakly-supervised and semi-supervised learning, specially in the field of text detection and recognition.

2.1 Text Detection and Recognition

Before the deep learning era, methods based on SWT or MSERs were used for text detection, for example [7, 30]. These methods are not being applied anymore with the exception of scenarios with hardware constraints and scenarios where the methods perform well. Subsequent models were mostly based on region proposal approaches like [15]. Recently, methods have rather turned to segmentation-based approaches like [20] and focused on representing arbitrarily shaped text, for example [2, 25].

Recent approaches for text recognition also rely on deep learning. Most methods can be described by 4 stages - transformation, feature extraction, sequence modelling and prediction. In the transformation stage, a Spatial Transformer Network [16] is used to normalize the input image. For feature extraction, a CNN such as VGG [35] or ResNet [13] maps the input image to feature maps. In the sequence modelling stage, BiLSTMs [10] are used to provide contextual information to the feature maps. The last stage employs either CTC [11] or attention-based prediction [5] to convert the encoded features into a character sequence.

Some methods treat the two tasks jointly, sharing features for both detection and recognition, for example [24, 3, 31]. Such end-to-end models have shown superior performance to treating the tasks separately.

All recent state of the art methods have relied on deep learning approaches, both in detection and recognition. The method for generating PGT proposed in this work does not make any assumptions about the implementation of the models and thus, any of the methods could be used. The methods used

in our experiments are explained in detail in Chapter 6.

2.2 Synthetic Data for Text Detection and Recognition

The work of [12] and [15] had a great influence on the performance of text detection and recognition systems. Synthetic data have proven to be very effective for training generic text localisation systems. Still, the lack of realism (both in terms of positioning, and blending with the scene), diversity (in terms of text styles and scene backgrounds) and contextualisation of the text in the scene, have been limiting factors. More recent work aims to improve some of these aspects [4, 40, 21, 26] or exploit real scene text data to do augmentation [8] but in our experience, still does not replicate the quality of real-world data.

2.3 Weakly and Semi-Supervised Learning

Both weakly and semi-supervised learning tackle the lack of annotated data. Semi-supervised learning refers to learning with both annotated and unannotated data. Usually, there is only a small amount of annotated data available and training with unannotated data can significantly improve performance.

In weakly supervised learning, the annotations are known to have some limitations and are usually inexpensive, often obtained automatically or from another task. The limitations may be due to noise, lack of accuracy or precision. Our method combines both approaches, building on top of the pseudo-labelling technique first introduced in [19]. It is a simple strategy for semi-supervised learning where part of the data is fully labelled and *Pseudo-Labels* are created for unlabelled data as the class with the maximum predicted probability and further treated as true labels. This is equivalent to entropy regularization [9], favoring low-density separation between classes.

2.3.1 Weakly and Semi-Supervised Learning for Text Detection and Recognition

In [37], focused on Chinese street view images, weak annotations are used where only the text-of-interest region is annotated. They suggest an online proposal matching module incorporated in the whole model. The main difference from our method is that they do not do any modification of the proposed regions.

In [31], an existing OCR engine different from the one being trained is used to provide partial labels for one million unlabelled images. The partially labelled data is then used to train the recognition part of an E2E

model, improving the results significantly. The method relies on a confidence threshold to filter out noisy labels while our method relies on weak annotations to minimize the risk of incorrect labelling.

Focusing on text detection, [32] propose multiple approaches for unsupervised and weakly supervised learning. Their unsupervised approach simply relies on filtering out predictions with low confidence score. An improved approach requires weak annotations, where regions containing text are annotated and it is known that regions distant from the annotated ones do not contain any text, allowing for more accurate false positives filtering. Their last approach relies on rectangular bounding boxes as weak labels.

Chapter 3

Fully Annotated Datasets

In this section, we present the existing fully annotated datasets that were used in the experiments for PGT generation, training and evaluation.

MJSynth (MJ) contains almost 9M synthetically generated images of English words for text recognition. The text generation process performs the following steps: Font rendering, border/shadow rendering, coloring, projective distortion, natural data blending and noise introduction [15].

SynthText (ST) is a synthetic dataset designed for scene-text detection, widely used for recognition, too. It has over 7M text instances in 8,000 images [12].

Synthetic Multi-Language in Natural Scene Dataset (MLT) contains 245,000 images in total with text instances in multiple scripts: Arabic, Bangla, Chinese, Japanese, Korean and Latin. The dataset was published in [3] and the authors have adapted the framework of [12]. A non-latin dictionary was used and it contains special, non-alpha-numeric characters. We only use the Latin script subset of the dataset, which contains 288,917 text instances in total [3]. An example image shown in Figure 3.1.

IIIT 5K-word (IIIT) is a collection of 5,000 cropped words from Google image search using queries such as "billboards" or "movie posters", which are likely to contain text [29]. The training set consists of 2,000 images, the remaining 3,000 form the test set.

Street View Text (SVT) was collected from the Google Street View, providing annotators with a lexicon for each image, containing texts such as business names. Only the words from the lexicon were localised and provided with transcription, the rest of the text is ignored. There are 257 and 647 images of cropped words in the training and test sets [39].

Street View Text - Perspective (SVT-P) is a dataset of 645 images collected from Google Street View focused on perspective projections [33].

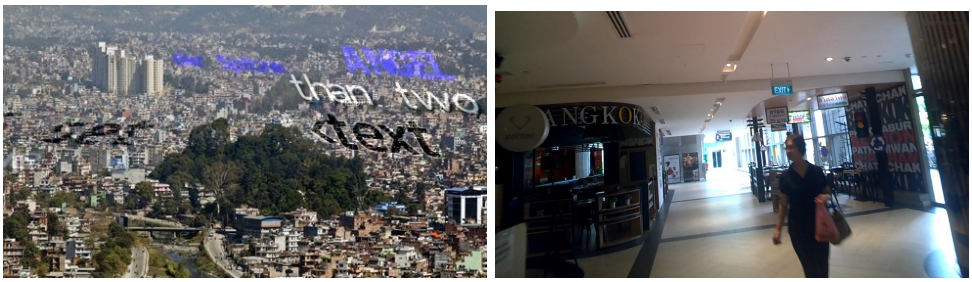
ICDAR2003 (IC03) has 258 training and 251 testing images with 1,156 and 1,110 annotated words respectively. It was collected for the ICDAR 2003 Robust Reading competitions [28].

ICDAR2013 (IC13) is a dataset with "focused text", the text being the main content of the image. It consists of a training set of 229 image with 848 words and a test set of 233 images with 1095 words. [18].

ICDAR2015 (IC15) , in contrast to IC13, focuses on incidental scene-text - the images were not taken with text in mind. The training set contains 1000 images (4,468 words) and the test contains 500 images (2,077 words) [17].
An example from the dataset is shown in Figure 3.1.

Total-Text (TT) is a dataset of 1,555 scene images with 9,330 annotated words. The images were collected with curved text in mind and the images often contain texts of different orientations [6].

CUTE80 (CT) contains 80 images with 288 words, focusing on curved text [34].



(a) : MLT-L [3] **(b) :** IC15 [17]

Figure 3.1: Synthetic (left) and real world (right) datasets for text detection.

Images of cropped words from the datasets are shown in Figure 3.2.

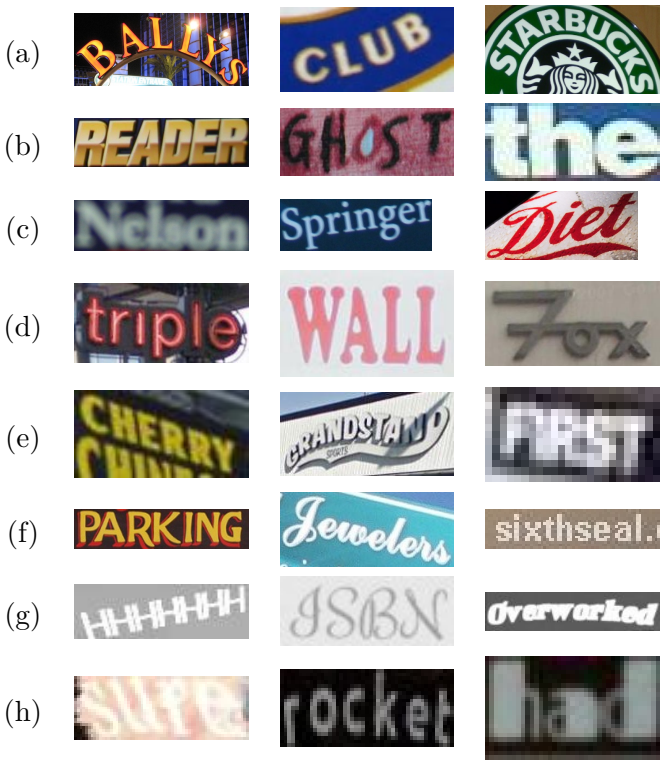


Figure 3.2: Examples from real-world datasets (a) CT [34] (b) IC03 [28] (c) IC13 [18] (d) SVT [39] (e) SVT-P [33] (f) IIIT [29] and from synthetic datasets (g) MJ [15] (h) ST [12]

Chapter 4

Weakly Annotated Datasets

This chapter discusses some of the potential sources of weakly annotated data and introduces the datasets that were used for PGT generation in our experiments.

The idea behind automatic acquisition of weakly annotated data is that for a lot of images, a collection of texts that are likely to appear in that image can be obtained easily and we believe such data can be exploited for text detection and recognition training, alleviating the work of human annotators. We mainly considered two different kind of sources: Product databases and mapping systems.

4.1 Weak Labels from Mapping Systems

One possible source of weakly annotated data are mapping systems. Given a picture and the GPS location where it was taken, one could retrieve a list of nearby businesses, street names, house numbers, restaurants and other texts. If some of it is actually present in the picture, it is possible that some of the text from the list is present in that image. This is illustrated in Figure 4.1. The data collected in this way has similar characteristics to many of the existing scene-text datasets.

A very simple way of obtaining such data would be making use of the Google Places API. GPS locations can be fed into the "Nearby Search", which outputs a list of nearby places and information about them, including URLs of pictures uploaded by users. For example, the images associated with a restaurant are likely to contain the name of the restaurant. Another possibility would be to make use of a Street View Crawler to collect the images. However, none of those options could be implemented due to policies that do not allow to store the collected data for academic or another purposes. Also, the API billing is per call which does not scale very well.

There are other mapping systems and it may be possible to use some of them, possibly together with images collected specifically for the purpose of

51.523784, -0.158314 (London, Baker Street)

Figure 4.1: Localising weak annotations from a mapping system. The "Nearby search" in the Google Maps Places API for the GPS location of a photo returns a list of businesses. The top (i.e. nearest) results might contain the following text: [The Sherlock Holmes Museum, 221b Baker St]. If the words highlighted in green are detected and recognised, possibly with errors, by an end-to-end text spotting system and reliably matched, the spatially localised text is treated as (pseudo) ground truth, PGT.

creating a weakly annotated datasets, however, none of those options was within the scope of this work.

To first test the viability of our approach, we decided to simulate the scenario of having scene-text like weakly annotated images on an existing dataset, the Uber-Text dataset [41].

Uber-Text dataset (UT) is one of the biggest datasets for text detection and recognition. It contains 117,969 images with 571,534 labelled text instances split into training, validation and test sets. Each set is divided into two subsets according to the image resolution - either 1K or 4K. The images were obtained through the Bing Maps Streetside program and come from 6 different cities in the US. The annotations are line-level. Most of the text regions form semantic units such as business names, street signs or street numbers. The datasets contains a lot of unannotated text, some text regions are not annotated at all, some readable text is labeled as unreadable. Images and annotations from the dataset are shown in Figure 4.2 [41].

Figure 4.2: The Uber-Text [41] training dataset. Annotation polygons: with complete transcription (blue), with unannotated characters (red). Some text regions are not annotated, some readable text is annotated as unreadable.

4.2 Weak Labels from Product Databases

Ranging from food to music, there are many product databases which have one thing in common - for each product, there is a photo of the product with corresponding metadata, which can be used as weak labels for the photo. With many of the images being born-digital documents, one may argue that they are very far from scene-text and may not be relevant. However, those images have one important thing in common - they often feature artistic texts written in uncommon fonts, possibly designed for single use only. Business owners want a unique style for their sign-boards, companies for their product logo, artists for their work. Furthermore, the same texts, like brand logos, are likely to appear in both scene-text images and product databases, using the same style. Figure 4.3 shows images from the Amazon CDs and Vinyl database that illustrate the variability of texts present in CD covers, which can contributed to different factors like font, distortion or complex background. It is very hard to cover all the possibilities when generating synthetic data. We can see that the name of the artist and the albums are present in the image, being the only text present, which makes the weak annotations less noisy then other possible sources.

For initial experiments, we chose book covers, which will be referred to as the Amazon Book Covers dataset.

Amazon Book Covers (ABC) is a dataset created from images of book covers downloaded from Amazon Books. We used a set of more than 200,000 images together with the author and the title for each book, which serve as

Figure 4.3: Weak annotations from the Amazon Music product database - the artist and the album name. The text style is very diverse and challenging for detection and recognition - the images contain curved and distorted text, complex background and difficult fonts.

Figure 4.4: Searching for 'Cafe London' photos with the 'Modifications allowed' license in the flickr application.

weak annotations. The same data were already used for genre prediction in [14]. Images from the dataset are shown in Figure 1.1.

4.3 Other sources

In future, we would like to explore the possibility of using the flickr online photo management and sharing application, which seems to have research friendly policies and free API, enabling develoeprs to filter images by the associated license. The API allows to search for photos by a tag. In this way, we could collect images of businesses and some of them would contain the texts from the image in their description.

Chapter 5

Method for Weakly Annotated Data Exploitation

This section describes the pseudo ground truth (PGT) generation algorithm (PGT-GEN). The algorithm uses weakly annotated images and an existing end-to-end reading system (E2E) performing text detection and recognition. All the steps are executed independently for each image, therefore, we define the algorithm for a single input image. First, we define the E2E output and the structure of the weak annotations. Then we describe the algorithm and its components in detail.

Given an image I, the output O of the end-to-end reading system,

$$O = \{(bb_1, tt_1), \dots (bb_t, tt_t)\}, \tag{5.1}$$

is a set of t text bounding box predictions and the corresponding text transcriptions. The transcriptions $T = (tt_1, \dots, tt_t)$ are strings (possibly containing spaces) and the bounding boxes are oriented rectangles characterized by their center coordinates, width, height and angle: $bb_i = (c_x, c_y, w, h, \alpha)$. It is also possible to obtain the recognition output from a bounding box bb separately:

$$\mathrm{REC}(I, bb) = tt. \tag{5.2}$$

Each image is associated with a list of texts $A = (t_1, t_2 \dots t_n)$ where each text $t_i = (w_1, w_2, \dots, w_m)$ is a non-empty ordered sequence of words. Words are strings that do not contain spaces. The set of weak labels $G = \bigcup_{i=1}^{n} g_i$ is obtained as a union of sets of k-grams, $k \in \{1, ..5\}$. Each set of k-grams g_i is formed by strings — consecutive words from t_i, sub-sequences of t_i of length k joined into a single string by the space character. In the simplest of cases, each text t_i only consists of a single word but because the texts are assumed to be extracted automatically as metadata accompanying the images, it may even be multiple words that form a semantic unit — a name of a product, its description, a business' name, contact information. These words are likely to appear in the image close to each other and get merged by the detector.

For example, the image from Figure 4.1 could be annotated with the texts "Sherlock Holmes, consulting detective" and "221B Baker Street". In that

case, A and G would be

$$A = (\text{“Sherlock”}, \text{“Holmes”}), (\text{“221B”}, \text{“Baker”}, \text{“Street”}))$$
$$G = \{\text{“Sherlock”}, \text{“Holmes”}, \text{“Sherlock Holmes”}, \text{“221B”},$$
$$\text{“Baker”}, \text{“Street”}, \text{“221B Baker”}, \text{“Baker Street”},$$
$$\text{“221B Baker Streeet”}\}.$$

Note that there is other text in the image, highlighted in blue, which is not present in A. On the other hand, A contains texts that do not appear in the image.

5.1 PGT-GEN algorithm

The PGT-GEN algorithm takes the image I, E2E output O and the set of weak labels represented as k-grams G as an input and outputs the PGT - a localized subset of G.

Algorithm 1: PGT-GEN

Input: I, O, G
Output: PGT
P := AssignWeak(O, G);
$PGT := \{\}$;
foreach $(bb, tt, g) \in P$ **do**
  (bb_f, tt_f) = FindOptimalBox(I, bb, tt, g);
  if IsPGT(tt_f, g) **then**
    $PGT = PGT \cup \{(bb_f, g)\}$;
  end
end
return PGT

AssignWeak - Weak annotation assignment. Each element from O is assigned at most one weak annotation from G. We construct a directed bipartite graph $B_G = (V, E)$ between O and G, thus $V = O \cup G$. For each proposal $o \in O$, $o = (bb, tt)$ and weak annotation $g \in G$ it holds that

$$(o, g) \in E \iff \text{dist}(tt, g) = \min_{i=1}^{|G|} \text{dist}(tt, g_i) \tag{5.3}$$

$$(g, o) \in E \iff \text{dist}(tt, g) = \min_{i=1}^{|T|} \text{dist}(tt_i, g) \tag{5.4}$$

where dist is the Levenshtein distance.

Finally, a set of proposals P is created:

$$P = \bigcup_{i=1}^{|O|} \text{Assign}(o_i, E) \tag{5.5}$$

$$\text{Assign}(o, E) = \begin{cases} \emptyset & \text{for } W = \emptyset \\ [W]_R & \text{otherwise} \end{cases} \tag{5.6}$$

$$W(o, E) = \{(o, g) : (o, g) \in E \wedge (g, o) \in E \wedge \text{match}(o, g)\}. \tag{5.7}$$

We define $\text{match}((bb, tt), g) = \frac{\text{dist}(tt,g)}{\max(\text{len}(tt),\text{len}(g))} \neq 1$ as a function to filter out proposals that do not make sense and $[.]_R$ selects a random element from a set. In most cases, $|W| = 1$.

At this point, we could apply some simple filtering to the set of proposals P instead of the edit distance guided neighbourhood search, for example, select

$$P' = \{p \in P, p = ((bb, tt), g) | dist(tt, g) = 0\} \tag{5.8}$$

and then output

$$\text{PGT} = \bigcup_{((bb,tt),g)\in P'} (bb, tt). \tag{5.9}$$

This would be equivalent to selecting such proposals where the predicted transcription was equivalent to the weak label text for PGT - we implement this version and compare it to the proposed one, showing the superiority of the proposed method.

FindOptimalBox - Edit distance guided neighbourhood search. For each proposal $(bb, tt, g) \in P$, we search for an optimal bounding box bb_f which minimizes the Levenshtein distance between the recognized text tt_f and g.

If $\text{dist}(tt, g) = 0$, we assume that bb is already optimal and assign $bb_f = bb$. If not, we predefine a set of new boxes in the neighbourhood of the original one and run the recognition on those in parallel, selecting one with minimal distance from g for bb_f. The generation of the set of predefined boxes in discussed in Subsection 5.1.1.

We compute

$$tt_f = REC(I, bb_f) \tag{5.10}$$

and the normalized edit distance between tt_f and g as

$$d = \frac{\text{dist}(tt_f, g)}{\max(\text{len}(tt_f), \text{len}(g))} \tag{5.11}$$

Finally, we find out whether (bb_f, tt_f) satisfies our requirements for being a PGT (**IsPGT**) as:

$$\text{IsPGT}(tt_f, g) = \begin{cases} True & \text{for } d = 0 \vee (d < \theta \wedge |tt_f| > \lambda \wedge tt_f^0 = g^0 \wedge tt_f^{-1} = g^{-1}) \\ False & \text{otherwise} \end{cases} \tag{5.12}$$

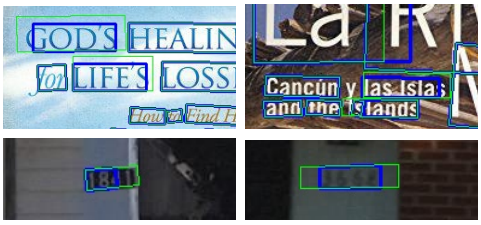


Figure 5.1: Localization of weak labels via neighbourhood search - the detected bounding boxes are shown in blue, the transformed ones in green. In some cases, the recognition model needs more context to recognize the text correctly. In others, the detection was imprecise.

where s^0 is the first character and s^{-1} is the last character of a string s. The thresholds θ, λ are set to $\theta = 0.35, \lambda = 4$. The intuition behind our choice of the IsPGT function is that even if the recognized text tt_f and the assigned text g are not identical, it is possible that there was simply an error in the recognition step. If the relative edit distance between two longer texts is low and the the first and the last characters are the same, it is likely that tt_f should actually be g. For example, if tt_f is "Boker" but g is "Baker", it is accepted as PGT. However, if tt_f was "Baked", it would not have been accepted.

Examples of how the neighbourhood search aids the PGT generation process can be seen in Figure 5.1.

5.1.1 Bounding box transformations

We define the following transformations to generate a set of bounding boxes in the neighbourhood of an input bounding box: Extending/shrinking the bounding box on the left/right/top. Angle modification and bottom extension/shrinkage were also considered but the benefits were insignificant. To keep the computational cost reasonable, we also assume that changes to the left side of the bounding box do not influence the recognition of the characters on the right side and vice versa, the optimization on each side is done independently. Horizontally, we extend/shrink the box with width w and height h in each direction by up to c characters, the character length being estimated as the average character length $ch_{avg} = \frac{w}{|text|}$. On the top, we extend by up to $\frac{h}{\beta}$ and shrink by up to $\frac{h}{\gamma}$.

Each of the transformed bounding boxes can be characterized by three integer parameters relative to the original bounding box - (t, l, r) - defining the extension/shrink (distinguished by the sign) by t, l, r units on top/left/right, where the horizontal unit is $\frac{ch_{avg}}{\delta}$ and the vertical unit is $\frac{h}{\kappa}$. Consequently, $l, r \in [-c \cdot \delta; c \cdot \delta]$ and $t \in [-\frac{\kappa}{\gamma}; 2\frac{\kappa}{\beta}]$. The transformed bounding boxes that exceed the image or do not overlap with the original one are immediately discarded.

To obtain the final bounding box bb_f characterized by (t_f, l_f, r_f), we find an optimal bounding box in both directions (left and right). For each direction, we find the set of boxes $B = \{(t_i, l_i, r_i)\}_{i=1\ldots n}$ that result in the lowest edit distance from g. The sets $T = \bigcup_{(t_i,l_i,r_i)\in B} t_i$, $L = \bigcup_{(t_i,l_i,r_i)\in B} l_i$ and $R = \bigcup_{(t_i,l_i,r_i)\in B} r_i$ are created.

From the boxes transformed in the left direction, we obtain

$$t_l = \min T \tag{5.13}$$

and

$$l_f = \frac{\min L + \min(\max L, o + \min L)}{2} \tag{5.14}$$

Analogously, for the right direction, we obtain

$$t_r = \min T \tag{5.15}$$

and

$$r_f = \frac{\min R + \min(\max R, o + \min R)}{2} \tag{5.16}$$

We set

$$t_f = \max(t_r, t_l) \tag{5.17}$$

In our experiments, the constants were assigned as follows: $c = 7$, $\beta = 2$, $\gamma = 4$, $\delta = 4$, $\kappa = 4$, $o = 8$. They were selected by observing qualitative results, attempting to minimize the amount of incorrect PGTs without loosing too many of the correct ones. The optimal numbers may vary according to the E2E system used and a more elaborate parameter search will likely improve the results.

Chapter 6

End-to-End Reading System

In this chapter, the end-to-end reading system (E2E) used in our experiments is described. Separate models for detection and recognition were used.

6.1 Detection

For text detection, we adopt TextSnake [25]. It is based on a fully convolutional network - U-net with VGG-16 [35] as stem network - which estimates the geometry attributes of text instances. A text instance is described as a sequence of ordered, overlapping disks centered at symmetric axes (center lines), each of which is associated with potentially variable radius and orientation. The architecture of the network is shown in Figure 6.1. and the text instance representation in Figure 6.2.

The following values are predicted for each pixel: tcl, tr, r and α, corresponding to the text center line, text region (which covers the text instance area), radius and angle. Those values are then post-processed to construct the text instances. In the original work, focused on arbitrary-shaped text, thresholds are applied to tcl and tr to obtain binary masks. Afterwards, the center lines are extracted as connected components of $tr \cdot tcl$ to account for tcl naturally being part of the tr. Each center line is associated with one text instance and the radius and angle predictions from the center line pixels are used to extract the sequence of disks that represent it. The whole pipeline is shown in Figure 6.2. For more details on the original text instance reconstruction, we refer the reader to the original paper as our work is focused on straight text and we will only describe the adapted post-processing steps used in our experiments.

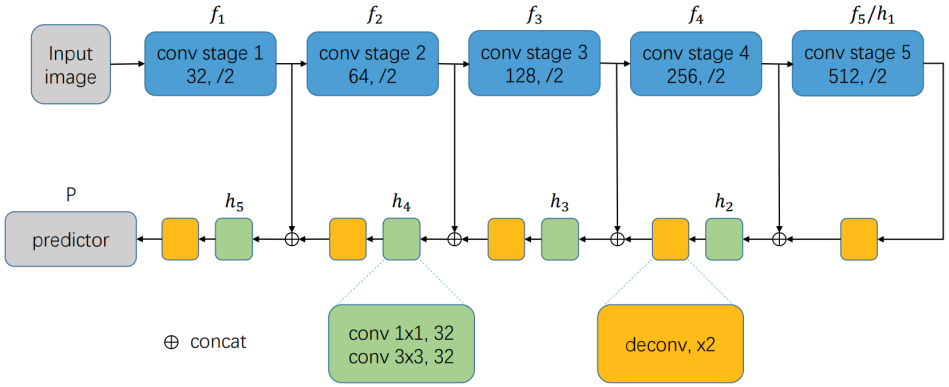
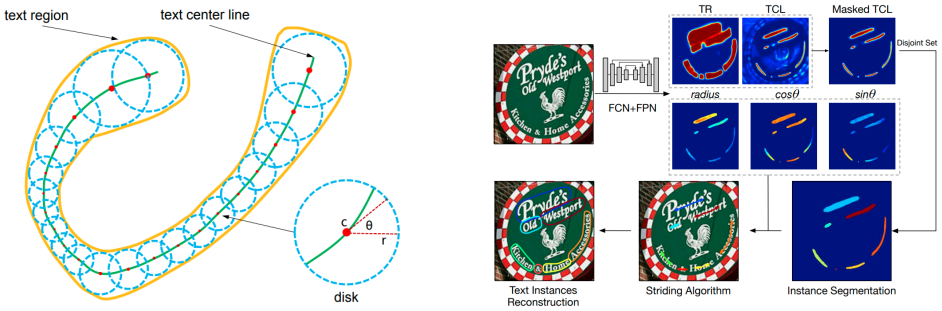


Figure 6.1: The architecture of TextSnake, reprinted from [25, Figure 4].



(a) : Text instance representation **(b) :** Detection pipeline

Figure 6.2: Text instance representation and the detection pipeline od TextSnake, reprinted from [25, Figure 2, 3]

6.1.1 Least squares fitting based text instance reconstruction

Taking into account the subsequent recognition, obtaining a rectangular bounding box for each text instance in an unavoidable step. We could use the original disk representation and convert it to a general polygon by simply merging all the instance disks and then obtain the bounding boxes as the minimum area rectangles of the polygons. This may result into sub-optimal angle prediction while the predicted *tcl* would have given a more precise estimate. Because the angle is essential to the PGT generation, we have adapted the post-processing method to skip the disk representation. We first convert *tr* and *tcl* into binary masks and obtain the connected components K from the masked center line prediction, $tr \cdot tcl$. This step is the same as in the original method. Then, for each component $k \in K$, where k are all the component points, we estimate the bounding box (c_x, c_y, w, h, α) (center coordinates, width, height and angle) directly.

The angle α is estimated by total least squares fitting (LSQ) of a line to the points of k shifted by the mean value of the coordinates $m = (m_x, m_y)$ to

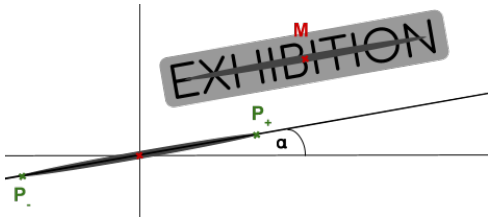


Figure 6.3: LSQ bounding box calculation - the *tr* prediction (light grey) and both the original *tcl* and the shifted *tcl* points (dark gray). M is the median point of the predicted *tcl* points.

the origin, using the slope of the best fitting line as the bounding box angle.

The height h is determined via the biggest radius predicted within the component: $h = \max(r[k]) \cdot 2$. To determine the width w, we project the shifted points onto the line with slope α passing through the origin and find the projected vectors with maximum norm in both directions, p_+ and p_-. The width is calculated as $w = |p_+ - p_-| + h$. The extra h is added to the width because during training, the *tcl* is shrank by $\frac{h}{2}$ (assuming the radius is constant, $\frac{h}{2}$, for straight text with rectangular ground truth) on each side. Afterwards, we shift p_+, p_- back by m and calculate the center of the bounding box as the middle point: $(c_x, c_y) = m + \frac{p_+ + p_-}{2}$.

The LSQ method is visualised in 6.3

6.2 Recognition

We adopt the best performing architecture from [1], which is very similar to the one of STAR-net [23] but with a different prediction mechanism. First, the input image is transformed into a rectified image using a variant of spatial transformer network (STN) [16]. The rectified image is then fed into a feature extraction module based on ResNet [13], which is followed by BiLSTM and an attention-based decoder. The nature of the decoder requires the set of characters to be extended by a special (EOS) (end of sentence) symbol.

The input are grayscale images and the input dimension is fixed to $H \times W$, $H = 32$ and $W = 150$ pixels. All images are first resized isotropically to height H. If the width of the resized image is less than W, the image is extended to the left and padded with zeros. If the width exceeds W, the image is horizontally shrunk to W - only in this case the aspect ratio of the input images is not preserved.

6.2.1 Transformation

The transformation module transforms an input image I into a rectified image $\tilde{I}$. It is an STN [16] which predicts the parameters of a thin-plate spline

Layers	Configurations
Input	grayscale image
Conv1	c: 64 k: 3×3
BN1	-
Pool1	k: 2×2 s: 2×2
Conv2	c: 128 k: 3×3
BN2	-
Pool2	k: 2×2 s: 2×2
Conv3	c: 256 k: 3×3
BN3	-
Pool3	k: 2×2 s: 2×2
Conv4	c: 512 k: 3×3
BN4	-
AdPool	$\rightarrow 512$
FC1	$\rightarrow 256$
FC2	$\rightarrow 2F$

Table 6.1: The architecture of the localization network from [1] - given a grayscale image as input, it outputs the (x, y) coordinates of the F fiducial points.

(TPS) transformation. The whole module consists of a localization network, a grid generator and a grid sampler.

A localization CNN predicts the coordinates of a set of k fiducial points C in the original image, $C = [c_1, \ldots, c_k] \in \mathbb{R}^{2\times k}$ where the i-th fiducial point $c_i = [x_i, y_i]^T$. A normalized coordinate system is used. No point annotations are needed to train the localization network, everything is trained in an end-to-end manner. The network architecture is summarized in Table 6.1.

The grid generator creates a sampler grid from the predicted parameters of the TPS transformation. A constant set of k base fiducial points, C', evenly distributed along the top and the bottom edge of the output rectified image, is generated.

The TPS transformation is represented by a matrix $T \in \mathbb{R}^{2\times(k+3)}$

$$T = \begin{pmatrix} \Delta_{C'}^{-1} & \begin{bmatrix} C^T \\ 0^{3x2} \end{bmatrix} \end{pmatrix} \tag{6.1}$$

where $\Delta_{C'}$ is a constant

$$\Delta_{C'} = \begin{bmatrix} 1^{k+1} & C'^T & R \\ 0 & 0 & 1^{1\times k} \\ 0 & 0 & C' \end{bmatrix} \tag{6.2}$$

and

$r_{ij} = d_{ij}^2 * \ln(d_{ij}^2)$, d_{ij} being the euclidean distance between c_i' and c_j'.

Figure 6.4: Transformation network output (left) and the original input image (right).

The grid of pixels on the rectified image I' is denoted by $P' = \{p_i\}_{i=1,\dots,N}$, $p' = [x_i, y_i]^T$ being the coordinates of the i-th pixel and N is the number of pixels. For each p'_i, the corresponding point p from the original image I is found:

$$r_{ij} = d_{ij}^2 * \ln(d_{ij}^2) \tag{6.3}$$

$$\hat{p}'_i = [1, x'_i, y'_i, r'_{i1}, \dots, r'_{ik}] \tag{6.4}$$

$$p = T\hat{p}'_i \tag{6.5}$$

where d_{ij} is the euclidean distance between p'_i and the k-th base fiducial point c'_j. Iterating over P', a grid $P = \{p_i\}_{i=1,\dots,N}$ is generated.

Finally, the value of the pixel at p'_i is bilinearly interpolated from the neighbourhood of p_i and by setting all the values, the rectified image I' is obtained

The rectified image has the same dimensions as the input image.

Transformed images are shown in Figure 6.4.

Layers	Configurations
Input	grayscale image
Conv1	c: 32 k: 3×3
Conv2	c: 64 k: 3×3
Pool1	k: 2×2 s: 2×2
Block1	$\begin{pmatrix} \text{c :128, k: } 3 \times 3 \\ \text{c :128, k: } 3 \times 3 \end{pmatrix} \times 1$
Conv3	c: 128 k: 3×3
Pool2	k: 2×2 s: 2×2
Block2	$\begin{pmatrix} \text{c :256, k: } 3 \times 3 \\ \text{c :256, k: } 3 \times 3 \end{pmatrix} \times 2$
Conv4	c: 256 k: 3×3
Pool3	k: 2×2 s: 2×2 p: 1×0
Block3	$\begin{pmatrix} \text{c :512, k: } 3 \times 3 \\ \text{c :512, k: } 3 \times 3 \end{pmatrix} \times 5$
Conv5	c: 512 k: 3×3
Block4	$\begin{pmatrix} \text{c :512, k: } 3 \times 3 \\ \text{c :512, k: } 3 \times 3 \end{pmatrix} \times 5$
Conv6	c: 512 k: 2×2 s: 1×2 p: 1×0
Conv7	c: 512 k: 2×2 s: 1×1 p: 0×0

Table 6.2: The architecture of the 32-layer ResNet based model from [1, 5] - batch normalization excluded for the sake of compactness.

6.2.2 Feature Extraction

Given the rectified image $\tilde{I}$, the feature extractor outputs a feature map

$$V = \text{CNN}(\tilde{I}) = \{v_i\}, i = 1, \dots K \tag{6.6}$$

where $K = 38$ is the number of columns in the output feature map.

The CNN is a 32-layer ResNet based model, the architecture details are in Table 6.2

6.2.3 Sequence modeling

BiLSTM creates contextual features from the visual features v_i and outputs $H = \text{Seq}(V)$. We use 2-layer BiLSTM. An i^{th} layer identifies two hidden states: Forward $h_i^{(t),f}$ and backward $h_i^{(t),b} \forall t$. A fully-connected layer between the two BiLSTM layers determines one hidden state, $\hat{h}_t^{(i)}$, from $h_i^{(t),f}$ and $h_i^{(t),b}$. The dimension of the hidden states and the FC layer is 256.

6.2.4 Prediction

Finally, a single layer LSTM attention decoder produces the output sequence of characters $Y = y_1, y_2, \dots y_n$

$$y_t = \text{softmax}(W_o s_t + b_o) \tag{6.7}$$

(W_o, b_0 are trainable parameters, s_t is the decoder LSTM hidden state at time t).

The LSTM hidden state s_t is computed as:

$$s_t = \text{LSTM}(y_{t-1}, c_t, s_{t-1}) \tag{6.8}$$

where c_t is a context vector which is computed as a weighted sum of H:

$$c_t = \sum_{i=1}^{I} \alpha_{ti} h_i. \tag{6.9}$$

The attention weight α_{ti} is obtained as

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{I} \exp(e_{tk})} \tag{6.10}$$

where

$$\exp(e_{ti}) = v^t \tanh(W s_{t-1} + V h_i + b) \tag{6.11}$$

(v, W, V, b trainable parameters).

The decoding stops when the (EOS) symbol is emitted.

The model is trained with the cross entropy loss function

$$L = -\sum_{t} ln P(\hat{y}_t | X) \tag{6.12}$$

where $\hat{y}_t$ is the ground truth of the t^{th} character and X is an image.

This recognition model is used in all of our experiments and will be referred to simply as OCR.

Chapter 7

Experiments

This chapter is dedicated to experimental results. The proposed pseudo ground truth (PGT) generation method was tested with two different sources of weakly annotated data, the Amazon book covers (ABC) and the Uber-Text (UT) training set where we ignored line-level localization information.

The method was applied iteratively, re-training the OCR model with the newly obtained data after each iteration. The recognition performance of the models was evaluated on the UT dataset, allowing for comparison with fully supervised training on the same dataset, as well as a number of different benchmarks. The generated data was also used to re-train the detection model and compared the number of generated PGT and performance on ICDAR2015 and UT.

7.1 Evaluation metrics

This section explains the evaluation metrics used in the experiments. When possible, conventional evaluation metrics were used, however, for the UT dataset, new metrics were created to account for different ground truth and detection granularity.

Two metrics are used for recognition evaluation - **accuracy** and **average normalized edit distance**. The accuracy of the model is computed as $\frac{|CRT|}{|T|}$ where $|CRT|$ is the number of correctly recognized texts and $|T|$ is the total number of texts. We use "texts" rather than "words" because in the UT dataset, the images with text to be recognized may contain multiple words. The normalized edit distance between a ground truth text g and a predicted text t is computed as $\frac{\text{dist}(t,g)}{\max(\text{len}(t),\text{len}(g))}$ where dist is the Levenshtein edit distance and $\text{len}(s)$ is the number of characters in a string s.

For detection evaluation, we compute the **recall** and **precision**. We adopt the evaluation protocol of [17], slightly modifying how `don't care` regions are treated. For a detection to be considered a true positive, it needs an overlap of at least 50 % with a ground truth bounding box. Detecting or missing

words marked as `don't care` does not have any influence on the results. In [17], any detections overlapping with a `don't care` bounding box by at least 50 % are ignored. We ignore any detections that have the intersection over detection ratio with some `don't care` bounding box of at least 30 % instead. This is to compensate for the inconsistencies in labelling `don't care` regions explained in Subsection 7.1.1. Ground truth regions marked as `don't care` are not further taken into account.

On the UT dataset, the detection recall and precision can not be computed directly because the ground truth is line-level and the detector output is not. We design two metrics that give us an insight into both the end-to-end performance (which reflects detection recall) of the model and the number of false positives.

E calculates the number of false positives as the number of detections that do not have the intersection over detection ratio with any of the ground truth regions of at least 50 %. If D_i are the detections predicted for the image i, R_i is the set of ground truth polygons for the image i and $\text{iod}(d, r)$ is the intersection over detection ratio of a ground truth polygon r and a detected polygon d, then

$$E = \frac{\sum_{i \in I} |F_i|}{\sum_{i \in I} D_i} \cdot 100 \tag{7.1}$$

where

$$F_i = \{d \in D_i | \max_{r \in R_i} \text{iod}(d, r) < 0.5\} \tag{7.2}$$

Q estimates the percentage of the ground truth text that was recognized correctly. While it only uses the recognition output, the quality of the recognition is conditioned by the detections and a better detector with the same recognition model should achieve higher values. It is computed as

$$Q = \frac{\sum_{i \in I} \sum_{g \in G(i)} \min_{d \in T(i)} \text{dist}(g, t)}{\sum_{i \in I} \sum_{g \in G(i)} \text{len}(g)} \cdot 100 \tag{7.3}$$

where I is the set of images, $G(i)$ is the set of ground truth words and $T(i)$ is the set of predicted region transcriptions split into separate words for image I and dist is again the Levenshtein distance and $\text{len}(s)$ is the number of characters in a string s.

7.1.1 Evaluation protocols drawbacks

Throughout the work, we have noticed several shortcomings of current evaluation protocols. While we mostly discuss the one of ICDAR2015 [17], the same or very similar protocols are used with most of the datasets.

The first problem stems from the inconsistent annotation of the `don't care` regions.. During evaluation, the detections that have the intersection over union of more than 50 % with a `don't care` region are discarded. This works well when separate words are annotated but in a lot of cases, the ground truth bounding boxes covers multiple words or even lines. This results into higher false positive rate for methods that detect this kind of texts. Also, even small but readable texts may be marked as `don't care`. Some very blurred text is not annotated at all. Jointly, these characteristics may favour detectors that ignore small and blurred texts altogether. Given the high precision numbers reached by recent methods, higher numbers may not necessarily correspond to better detectors anymore. The problem of imprecise annotations is illustrated in Figure 7.1.

Another question is how strongly should word-level detection be enforced. If the detector merges two words, the detection is usually matched with the longer one and the shorter one is considered undetected, which may not be the desired behaviour - currently, a detector that does not predict the shorter word at all obtains the same score as a detector that merged them. Often, in an end-to-end scenario, those merged words would be recognized as separate words by the recognition and it would not harm the end-to-end performance. The only case when the horizontal merge is really undesirable is when the words come from different semantic unit, like merging the text from two different street signs together.

So far, scene-text detection and recognition has only focused on separate words or lines detection. While that is a good starting point, the textual information is usually structured and with the emergence of new applications like visual question answering, it should be taken into account, possibly leading to a scene-text analogy of document layout analysis.

Overall, the evaluation protocols may benefit from a revision taking into account the advancement of the field, possibly leading to better end-to-end reading systems.

7.2 Pretraining E2E

The detection part of E2E (TextSnake) was trained on a mix of SynthText, ICDAR2015 and Total-Text datasets. Basic augmentation techniques like cropping and rotation are used. The post-processing thresholds of TextSnake were set to $tr = 0.4$ and $tcl = 0.7$, which lead to the best PGT generation performance on a small subset of UT and ABC images. We do not filter out the text marked as `unreadable` or `don't care` during training to maximise the use of available data. Detection recall is more important than precision for PGT generation – the more words detected, the more potential pseudo-labelled examples are available. On the other hand, false positives are very unlikely to be matched against weak annotations, thus they have minimal impact, besides slowing down the process.

The recognition part (OCR), which also serves as a baseline model (OCR_b) in our experiments, was trained on the ST (Synth-text), MJ (MjSynth) and MLT (Synthetic Multi-Language in Natural Scene) datasets. The OCR_b recognizes 70 characters – letters, not distinguishing lower and upper case, digits and frequent special characters like punctuation, brackets, the `(EOS)` symbol and the space.

Most recognition datasets provide word-level annotations, and thus `space` is never part of the transcription. We included `space` in the character set for three different reasons. First, if the model is capable of predicting spaces, it helps to guide the PGT generation process - a bounding box that is too wide leads to a space being predicted at the beginning or end of the transcription. Second, if the detector merges horizontally adjacent words into a single bounding box, the recognizer often splits the text by recognising a `space` between the merged words. Third, it allows exploiting annotations that contain multiple words.

Synthetic datasets used for training have word-level annotations and thus provide no training data for the `space` character. We therefore extended some of the bounding-boxes and included spaces at the beginning and end of the ground truth annotations. This produced a model with a limited ability to recognize the `space`. The ability was further improved during training on PGT, since it contains multi-word texts. During evaluation, we strip any leading/trailing spaces from the predictions.

The OCR processes images with fixed resolution of 32×150. Input images are first resized isotropically to height 32. If the width of the resized image is less than 150, the image is extended to the left and padded with zeros.u If the width exceeds 150, the image is horizontally shrunk to 150 - only in this case the aspect ratio of the input images is not preserved.

The implementation builds on top of
https://github.com/princewang1994/TextSnake.pytorch and
https://github.com/clovaai/deep-text-recognition-benchmark.

7.3 PGT from the Uber-Text Dataset

The experiment evaluates the PGT method as an adaptation technique to the Uber-Text dataset domain. The performance is also compared to fully supervised training in the UT domain.

A reference method, OCR_{UT_F}, is obtained by fully supervised training on UT_F, the set of 138,437 transcriptions and corresponding rectangular crops from the UT training set that contain no unreadable characters in transcription. The crops are the minimum area enclosing rectangles of the ground truth polygons. OCR_{UT_F}, as well as other OCRs described in this section, are validated on a set of 5,000 random transcriptions from the UT validation set is created.

Iteration #	Mined total	Mined w/o n. s.	Mined n. s.
1	92,909	72,990	19,919
2	105,126	86,295	18,831
3	109,557	90,480	19,077
4	111,663	92,660	19,003
5	113,046	93,994	19,052
6	113,810	94,890	18,920

Table 7.1: The results of six iterations of the PGT generation method on Uber-Text dataset with simulated weak labels. It shows how many images were generated in total and how many would have been obtained with/without the neighbourhood search.

For PGT generation, the whole UT training set is used. To facilitate GPU computations, we split large (about 4K) images into 16 blocks ensuring no text instance is split and discard those with no text. Such empty blocks are common, since text instances are sparse in many images. Each of the original ground truth transcriptions is a weak label in our experiment, the ground truth polygons are discarded. The weak labels are transformed into a set of n-grams, as explained in the PGT generation section. N-grams containing the $*$ symbol (unreadable or unknown characters) are discarded.

The PGT generation and OCR training progresses iteratively. First, the OCR_b results are processed, creating the UT_1 PGT dataset. Next, an updated model, $\mathrm{OCR}_{\mathrm{UT}_1}$, is trained with UT_1. In the k-th iteration, $k > 1$, while keeping the detections fixed, the OCR trained in the previous iteration, $\mathrm{OCR}_{\mathrm{UT_{k\text{-}1}}}$, generates the new PGT UT_k dataset.

PGT generation and OCR training. In the first iteration, 92,909 PGT text instances were obtained. The number of PGT texts increased in all iterations, reaching 113,810 texts after six iterations when the OCR performance stopped improving – a summary is shown in Table 7.1. The recognition rate, calculated on 20 000 randomly selected transcriptions from the UT test set, increased in each iteration from the baseline 41.6 % to 66.1 % in the sixth iteration. The accuracy of the fully supervised $\mathrm{OCR}_{\mathrm{UT_F}}$ is 78 %. The PGT has reduced the gap between the baseline model and the fully-supervised one by 67%. The performance of PGT training is limited by the detector which was not trained on the new domain. Improving the detector may help to reduce the gap further.

To test the contribution of the neighbourhood search and of allowing imperfect matches, a dataset, denoted UT'_1, is created. It contains only the detections that matched with 0 edit distance with some of the weak labels. The accuracy of the model trained with UT'_1 is 2.7 % lower, showing the importance of the additional retrieved PGT text. Table 7.2 summarizes the UT experiments.

PGT precision analysis was performed on a set of 5,000 images. For each image I, we denote the set of generated PGT as $\mathrm{PGT}(I) = \{(p_i, t_i)\}_{i=1,\ldots n}$

	Training datasets - example %						
	Full				Weak		
	MJ	ST	MLT	UT_F	UT (30)	Acc.	Norm. ED
[42]						56.4	
OCR_b	45	45	10	0	-	41.6	35.2
$OCR_{UT'_1}$	30	30	10	0	UT'_1	55.2	28.0
OCR_{UT_1}	30	30	10	0	UT_1	57.9	26.2
OCR_{UT_2}	30	30	10	0	UT_2	62.7	23.3
OCR_{UT_3}	30	30	10	0	UT_3	64.4	22.5
OCR_{UT_4}	30	30	10	0	UT_4	65.4	21.5
OCR_{UT_5}	30	30	10	0	UT_5	66.0	21.1
OCR_{UT_6}	30	30	10	0	UT_6	66.1	21.2
OCR_{UT_F}	30	30	10	30	-	78.0	10.0

Table 7.2: Recognition rates (acc.) and the normalized edit distance on the Uber-Text test set. The data obtained from the i^{th} iteration of the PGT generation is denoted as UT_i. UT_F is the fully annotated dataset and UT'_1 is a subset of UT_1 obtained without the neighbourhood search.

Iteration / Mined:	total	w/o neigh. s.	with neigh. s.
1	1,536,583	1,234,219	302,364
2	1,581,109	1,354,219	226,890
3	1,594,333	1,375,571	218,762

Table 7.3: Pseudo-ground truth (PGT) generation performance on the ABC dataset. The number of boxes with text generated: in total, without and with the neighbourhood search.

where p_i and t_i are the bounding box and the transcription of the i-th PGT element. and the ground truth set as $\mathrm{GT}(I) = \{(p_i, t_i)\}_{i=1,\dots m}$.

For each $(p, t) \in \mathrm{PGT}(I)$, we attempt to find $(p', t') \in \mathrm{GT}(I)$ such that $a(p \cap p')/a(p') > 0.3$ and t is a substring of t'; $a(.)$ is the area of the polygon. If no such GT element exists, the PGT element is considered a false positive. The estimated number of false positives in PGT was 1.6%. We consider this an upper bound on PGT precision since often the detected "false positives" are in fact texts that do appear in the image but were not annotated, while the same texts appear and were annotated in a different part of the image and thus were among the weak labels - an example would be the name of a restaurant written on a sign board and also on the entrance door. The majority of the real false positives are very common short words such as 'the', 'to', 'in' or 'on' that are being predicted for false positive detections.

Some of the PGT failure cases are shown in Fig. 7.2.

7.4 PGT from Book Covers

The PGT is generated from the ABC dataset. Over 200,000 images of book covers together with the author and the title of each book were used. The title often includes a subtitle - while the author and title are almost always present in the image, the subtitle is less common, or there may only be a part of it.

We performed three iterations, in the first, the OCR_b model generates the ABC_1 PGT dataset of 1,536,583 cropped images of texts. The model trained with this data is referred to as $\mathrm{OCR}_{\mathrm{ABC}_1}$. In the second iteration, using $\mathrm{OCR}_{\mathrm{ABC}_1}$ instead of OCR_b, we obtain the ABC_2 dataset with 1,581,109 images. Finally, the $\mathrm{OCR}_{\mathrm{ABC}_2}$ model is trained and used to generate ABC_3, a dataset of 1,594,333 images. A summary of the PGT generation results from all the iterations can be found in Table 7.3.

7.5 Recognition results on Benchmark Datasets

A recognition model trained with the PGT data from the previous experiments is evaluated on different domains using various commonly used recognition datasets - IIIT 5K-word (IIIT), Street View Text (SVT), ICDAR2003 (IC03), ICDAR2013 (IC13), ICDAR2015 (IC15), Street View Text - Perspective (SVT-P) and CUTE80 (CT). We evaluate on test set subsets commonly used by researches as identified in [1]. All the models were validated on a union of the training sets of all the previously mentioned datasets.

To evaluate models trained on word-level data only, a test set of 20,000 images where each image only contains a single word, referred to as UT_{W}, is created. While we evaluated the models on UT_{W}, it was not included in the validation set. We remove any spaces from all the predictions and when evaluating on datasets with images that contain punctuation but the ground truth does not, we filter any non-alphanumeric characters out.

Training with either UT_1 or ABC_1 generated in the first iterations of the PGT generation consistently improved the performance. For some datasets, UT boosted the performance more than ABC and vice versa and training with both leads to a superior performance on all evaluated datasets. Training with the data from the last iterations, UT_6 and ABC_3, further improved the accuracy with an average improvement of 3.7 % relative to OCR_b.

A model trained with the UT_1' and ABC_1' datasets was also evaluated. These datasets are subsets of the UT_1 and ABC_1 datasets that would have been obtained if no neighbourhood search or edit distance filtering was used. With the exception of IC03 dataset, this model's performance was always inferior to the model trained with all the data.

The results also show that while the baseline model trained on synthetic data

	Training datasets - % in batch				
	Full			Weak	
	MJ	ST	MLT	UT	ABC
OCR_b	45	45	10	0	0
OCR_{ABC_1}	30	30	10	0	30
OCR_{UT_1}	30	30	10	30	0
$OCR_{UT_1+ABC_1}$	20	20	10	20	30
$OCR_{UT_6+ABC_3}$	20	20	10	20	30
$OCR_{UT'_1+ABC'_1}$	20	20	10	20	30
OCR^{α}_b	50	50	0	0	0
$OCR^{\alpha}_{UT_6+ABC_3}$	25	25	0	20	30

Table 7.4: Recognition models and their training datasets, evaluated in Tables 7.5 and 7.6.

	Evaluation on							Summary				
	IIIT	SVT	IC03	IC13	IC15	SP	CT	Δ			UT_w	Δ
	3000	647	867	1015	2077	645	288	avg	min	max	20 000	
OCR_b	89.8	86.7	94.3	91.2	68.5	77.2	72.3	0.0	0.0	0.0	52.8	0.0
OCR_{ABC_1}	92.8	88.9	94.8	93.0	71.0	77.5	73.6	1.7	0.3	3.0	53.9	1.1
OCR_{UT_1}	92.2	88.6	95.0	94.1	70.6	79.2	76.4	2.3	0.7	4.1	61.6	8.8
$OCR_{UT_1+ABC_1}$	93.0	89.2	95.2	94.1	71.6	79.2	77.8	2.9	0.9	5.5	61.8	9.0
$OCR_{UT_6+ABC_3}$	93.5	90.7	95.5	94.0	74.6	80.1	77.8	3.7	1.2	6.1	67.8	15.0
$OCR_{UT'_1+ABC'_1}$	91.4	88.1	95.5	93.9	69.5	77.1	74.0	1.4	-0.1	2.7	59.1	6.3

Table 7.5: Recognition results on standard benchmarks, non-alphanumeric characters included. Validation was performed on the union of training sets, with the exception of the UT dataset. Average, min. and max. increments of each model relative to OCR_b (Δ).

only performed well over a wide range of different datasets, the performance on UT dataset was rather poor - only 52.8 % accuracy. This shows the challenging nature of the dataset, which is partially due to the presence of heavily blurred images and the high frequency of vertical/diagonal text direction. The summary of the experiments can be seen in Table 7.5.

For better comparison with other methods, we also trained and evaluated our best performing model on alpha-numeric characters only. The baseline model was pretrained with MJ and ST and fine-tuned with the the UT_6 and ABC_3. During evaluation, all images with unknown characters were filtered out. The boost in performance here was slightly lower, 3.3 % on average.

It should also be taken into account that the relatively poor performance on the IC15 dataset is partially due to the presence of highly rotated text, which we have not addressed at all.

The results confirm that training with automatically generated PGT from very different domains, such as born-digital documents, can significantly improve the performance of a recognition model over a wide range of scene-text datasets. Also, adding only a relatively small number of those images helps significantly, implying the variety of data is an important factor. Some common characteristics of the images where the PGT data has improved

	Evaluation on							Summary				
	IIIT	SVT	IC03	IC13	IC15	SP	CT	Δ			UT_W	Δ
	3000	647	867	1015	1922	645	287	avg	min	max	18 956	
OCR_b^α	87.6	88.6	94.5	91.9	75.2	78.9	73.2	0	0	0	54.9	0.0
$OCR_{UT_6+ABC_3}^\alpha$	91.7	91.8	95.7	94.2	80.0	82.5	77.4	3.3	1.2	4.8	68.9	14.0
Publ. SOTA	95.3	91.8	96.4	96.4	79.4	84.5	88.5					
References	[20]	[22]	[22]	[27],[20]	[22]	[22]	[20]					

Table 7.6: Recognition results on standard benchmarks excluding images containing non-alphanumeric characters. For comparison with other methods, the model was only pre-trained with MJ and ST datasets. Average, min. and max. increment relative to OCR_b (Δ).

the model's performance are blurred images, perspective distortions, artistic/handwritten fonts or occluded/cropped characters. Some of these are shown in Figure 7.3 while images where the performance has deteriorated are shown in Figure 7.4.

7.6 Detection training with PGT

Detection training with PGT is not as straightforward as recognition training where we could simply crop out the PGT regions to create a new training set. The situation is the following: For each image, we have a set of detections. Some of those (possibly modified) detections were confidently matched with a weak label and form a part of the PGT but we can not say much about the remaining ones. Those could be either false positives or true positives. Also, we do not know anything about the possible false negatives.

The problem of the "unconfirmed" detections could be addressed by using the recognition output confidence to obtain more information and discard detections below a certain threshold as false positives. During training, the regions corresponding to detections that did not make it to the PGT could be masked so that no loss is computed in those regions. However, that does not solve the false negatives problem and thus, training on the original images similarly to training on regular fully-supervised datasets does not seem like a feasible solution.

We propose training on positive PGT examples only. To use the GPU efficiently and given the image size is fixed to $H \times W$, we create new training images by concatenating the cropped PGT regions. For each row of the image, we randomly pick a height h_i from $[h_{low}, h_{high}]$ and fill it with cropped PGT regions resized so that the height of the cropped image is h_i, preserving the aspect ratio. If the next image does no fit in the row anymore, we fill any remaining columns in the row by copying the last column of the last cropped image. If $H - \sum_{j=1}^{i} h_j < h_{high}$, we set $h_{i+1} = H - \sum_{j=1}^{i} h_j$ instead of selecting it randomly and create the last $i+1$-th row. The limits on height are set to $h_{low} = 25$ and $h_{high} = 80$. One of the resulting images is shown in Figure 7.5, together with the corresponding text center line and text region

	Training dataset %							
	Full			Weak		Evalaution		
	IC15	TT	MLT-L	UT_1	ABC_1	Recall	Precision	CRT
A	70	20	10	0	0	77.2	70.4	1169
B	50	20	10	10	10	77.4	74.1	1156
C	60	20	10	10	0	77.0	74.0	1147
D	50	20	10	20	0	74.4	71.5	1130
E	45	15	10	15	15	75.5	73.5	1148
F	70	10	10	10	0	75.4	67.5	1139
G	70	10	10	5	5	73.0	73.5	1084

Table 7.7: Detection evaluated on the ICDAR2015 [17] test set. OCR_{UT_1} was used to obtain the number of correctly recognized texts (CRT).

training masks.

The results of TextSnake trained on different combinations of IC15, TT, MLT-L, UT_1 and ABC_1 datasets evaluated on the IC15 test set, using OCR_{UT_1} as recognizer, are reported in Table 7.7. Due to the lack of a validation set for IC15, the results are not conclusive and the small changes in performance could be attributed to oscillations in between training epochs. There may be a small increase in model precision at the cost of worse end-to-end performance in terms of the number of correctly recognized words.

We also evaluate the Q and E metrics on a subset of 5000 images of the UT test set. The results are summarized in Table 7.8. The results seem consistent with the observations made on the IC15 dataset - while the percentage of false positives, as well as the number of detections, decreases, the end-to-end performance deteriorates. We have also performed the second iteration of the PGT generation, this time using one of the retrained models as well as the retrained OCR_{UT_1}. The number of PGT generated was lower than the number of PGT generated while keeping the detector unchanged. This confirms our approach is not feasible.

	Training dataset %								
	Full			Weak		Evalaution			
	IC15	TT	MLT-L	UT_1	ABC_1	Q	E	Detections	Mined
A	70	20	10	0	0	36.5	24.9	16214	105126
B	50	20	10	10	10	38.6	17.8	14050	-
C	60	20	10	10	0	39.7	18.5	13891	-
D	50	20	10	20	0	38.8	19.5	15020	101929
E	45	15	10	15	15	39.0	15.7	13646	-
F	70	10	10	10	0	37.8	22.4	15584	-
G	70	10	10	5	5	40.3	19.4	13012	-

Table 7.8: Evaluation on a 5000 image subset of the Uber-Text [41] test set. OCR_{UT_1} was used to compute Q. The number of mined PGT regions is not evaluated for the majority of the models as the time and gpu demands are high and the results are unlikely to bring extra value.

(a) : Some `don't care` polygons span multiple lines - with the evaluation protocol of [17], this would result in multiple false positive predictions.

(b) : Some ground truth polygons are bigger than necessary. As a result, even detections that actually cover the whole word do not pass the 50 % intersection over union threshold.

Figure 7.1: Problematic ground truth polygons in [17]. The top images show the ground truth polygons, both regular (blue) and `don't care` (yellow). The bottom images show the predicted detections with - true positives (green), false positives (red) and ignored during evaluation (blue).

Figure 7.2: Failure cases of our PGT method (blue) - sometimes, very blurred texts, texts in unknown scripts or text-like patterns are recognized as short, common words such as 'on', 'the' or 'in'.

Figure 7.3: Images where the edit distance between the prediction and the ground truth decreased after training with PGT. OCR prediction (first line) and $\text{OCR}_{\text{UT}_6+\text{ABC}_3}$ prediction (second line) are shown below each image. Some common characteristics are blur, occlusion, perspective distortion or artistic fonts.

Figure 7.4: Images where the edit distance between the prediction and the ground truth increased after training with PGT. OCR prediction (first line) and $\text{OCR}_{\text{UT}_6+\text{ABC}_3}$ prediction (second line) are shown below each image.

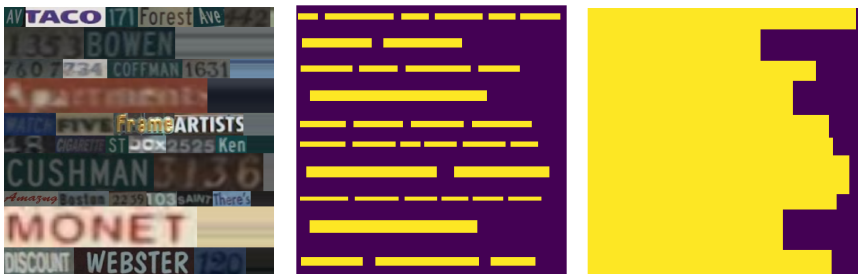


Figure 7.5: Detection training with PGT - the concatenated PGT image and the corresponding text center line and text region training masks.

Chapter 8

Conclusions and future work

In this work, we proposed and tested a method that exploits weakly annotated images. The weak annotations are texts that are being extracted automatically as metadata accompanying the images. An existing end-to-end reading system is used to localize the weak annotations, improving the results with an edit-distance guided neighbourhood search, alleviating human effort in annotating data for text detection and recognition. The method produces very little false positives and the output can be treated as regular ground truth, therefore we call it pseudo ground truth (PGT).

We applied the method to two different sources of weakly annotated data, the Uber-Text dataset and book covers from Amazon. The generated PGT was used to retrain both the detection and recognition models. It was shown that training with PGT consistently improves the accuracy of a state of the art recognizer both on images from the same domain and across different benchmark datasets (different domains). The proposed approach to detection training — training on positive samples only concatenated into a single image — showed small increase in precision but a decrease in recall and end-to-end performance on all evaluated datasets and does not seem feasible. Throughout the work, we have encountered several shortcomings of current evaluation protocols, which are discussed in Subsection 7.1.1.

The promising results open multiple directions of future work, which are discussed in the rest of the chapter.

8.1 Limitations and future work

We are aware of multiple limitations of our work which open the space for multiple directions of future work.

The first one stems from evaluating detection and recognition separately. The PGT images used to improve the recognition model were obtained through the detector, cropping out rotated bounding boxes, while the recognition test set often contained axis aligned bounding boxes created by human annotators.

Working with an end-to-end system and focusing on end-to-end performance may benefit from the PGT even more.

Our work was also limited by the fact that we only focused on straight-text — an adaptation of the PGT generation method to handle curved text is likely going to obtain more diverse and abundant data. Another way to improve the PGT data would be looking into more sources of weak annotations like flickr, as explained in Section 4.3.

Lastly, while our approach to detection training turned out as infeasible, it does not mean detectors can not benefit from the PGT. Even the msot recent synthetic datasets still lack the text style diversity of real world images and one possibility to improve that would be a combination of the PGT with style-transfer techniques, but those experiments were out of the scope of this work.

Appendix A

Bibliography

[1] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. *arXiv preprint arXiv:1904.01906*, 2019.

[2] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[3] M. Bušta, Y. Patel, and J. Matas. E2e-mlt-an unconstrained end-to-end method for multi-language scene text. In *Asian Conference on Computer Vision*, pages 127–143. Springer, 2018.

[4] D. Chen, L. Lu, Y. Lu, R. Yu, S. Wang, L. Zhang, and T. Liu. Cross-domain scene text detection via pixel and image-level adaptation. In *International Conference on Neural Information Processing*, pages 135–143. Springer, 2019.

[5] Z. Cheng, F. Bai, Y. Xu, G. Zheng, S. Pu, and S. Zhou. Focusing attention: Towards accurate text recognition in natural images. In *Proceedings of the IEEE international conference on computer vision*, pages 5076–5084, 2017.

[6] C. K. Ch'ng et al. Total-text: Towards orientation robustness in scene text detection. *International Journal on Document Analysis and Recognition (IJDAR)*, 23:31–52, 2020.

[7] 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. IEEE, 2010.

[8] R. Gomez, A. Furkan Biten, L. Gomez, J. Gibert, D. Karatzas, and M. Rusiñol. Selective style transfer for text. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 805–812, 2019.

[9] Y. Grandvalet and Y. Bengio. Entropy regularization. *Semi-supervised learning*, pages 151–168, 2006.

[10] Graves et al. Bidirectional lstm networks for improved phoneme classification and recognition. In *ICANN*, pages 799–804, 2005.

[11] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML '06*, 2006.

[12] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2315–2324, 2016.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] B. K. Iwana, S. T. R. Rizvi, S. Ahmed, A. Dengel, and S. Uchida. Judging a book by its cover. *arXiv preprint arXiv:1610.09204*, 2016.

[15] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20, 2016.

[16] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[17] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015.

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

[19] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 2, 2013.

[20] M. Liao, P. Lyu, M. He, C. Yao, W. Wu, and X. Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[21] M. Liao, B. Song, M. He, S. Long, C. Yao, and X. Bai. Synthtext3d: Synthesizing scene text images from 3d virtual worlds. *arXiv preprint arXiv:1907.06007*, 2019.

[22] R. Litman, O. Anschel, S. Tsiper, R. Litman, S. Mazor, and R. Manmatha. Scatter: Selective context attentional scene text recognizer. *arXiv preprint arXiv:2003.11288*, 2020.

[23] W. Liu, C. Chen, K.-Y. K. Wong, Z. Su, and J. Han. Star-net: A spatial attention residue network for scene text recognition. In *BMVC*, volume 2, page 7, 2016.

[24] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan. Fots: Fast oriented text spotting with a unified network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[25] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36, 2018.

[26] S. Long and C. Yao. Unrealtext: Synthesizing realistic scene text images from the unreal world. *arXiv preprint arXiv:2003.10608*, 2020.

[27] N. Lu, W. Yu, X. Qi, Y. Chen, P. Gong, and R. Xiao. Master: Multi-aspect non-local network for scene text recognition. *arXiv preprint arXiv:1910.02562*, 2019.

[28] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 682–687. Citeseer, 2003.

[29] A. Mishra, K. Alahari, and C. V. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2012.

[30] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Asian Conference on Computer Vision*, pages 770–783. Springer, 2010.

[31] S. Qin, A. Bissacco, M. Raptis, Y. Fujii, and Y. Xiao. Towards unconstrained end-to-end text spotting. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[32] X. Qin, Y. Zhou, D. Yang, and W. Wang. Curved text detection in natural scene images with semi-and weakly-supervised learning. *arXiv preprint arXiv:1908.09990*, 2019.

[33] T. Quy Phan, P. Shivakumara, S. Tian, and C. Lim Tan. Recognizing text with perspective distortion in natural scenes. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[34] A. Risnumawan, P. Shivakumara, C. S. Chan, and C. L. Tan. A robust arbitrary text detection system for natural scene images. *Expert Systems with Applications*, 41(18):8027–8048, 2014.

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[36] Y. Sun, J. Liu, W. Liu, J. Han, E. Ding, and J. Liu. Chinese street view text: Large-scale chinese text reading with partially supervised learning. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[37] Y. Sun, J. Liu, W. Liu, J. Han, E. Ding, and J. Liu. Chinese street view text: Large-scale chinese text reading with partially supervised learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9086–9095, 2019.

[38] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.

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

[40] F. Zhan, C. Xue, and S. Lu. Ga-dan: Geometry-aware domain adaptation network for scene text detection and recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9105–9115, 2019.

[41] Y. Zhang, L. Gueguen, I. Zharkov, P. Zhang, K. Seifert, and B. Kadlec. Uber-text: A large-scale dataset for optical character recognition from street-level imagery. In *SUNw: Scene Understanding Workshop - CVPR 2017*, Hawaii, U.S.A., 2017.

[42] Y. Zhang, L. Gueguen, I. Zharkov, P. Zhang, K. Seifert, and B. Kadlec. Uber-text: A large-scale dataset for optical character recognition from street-level imagery. In *SUNw: Scene Understanding Workshop-CVPR*, 2017.