TEXTRON: Weakly Supervised Multilingual Text Detection through Data Programming

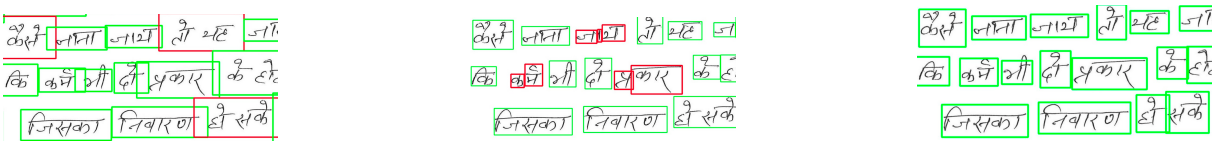
Dhruv Kudale, Badri Vishal Kasuba, Venkatapathy Subramanian, Parag Chaudhuri, Ganesh Ramakrishnan,
{dhruvk, badrivishalk, venkatapathy, paragc, ganesh}@cse.iitb.ac.in
Department of Computer Science and Engineering
IIT Bombay, India

Abstract

Several recent deep learning (DL) based techniques perform considerably well on image-based multilingual text detection. However, their performance relies heavily on the availability and quality of training data. There are numerous types of page-level document images consisting of information in several modalities, languages, fonts, and layouts. This makes text detection a challenging problem in the field of computer vision (CV), especially for low-resource or handwritten languages. Furthermore, there is a scarcity of word-level labeled data for text detection, especially for multilingual settings and Indian scripts that incorporate both printed and handwritten text. Conventionally, Indian script text detection requires training a DL model on plenty of labeled data, but to the best of our knowledge, no relevant datasets are available. Manual annotation of such data requires a lot of time, effort, and expertise. In order to solve this problem, we propose TEXTRON*, a* Data Programming-based approach*, where users can plug various text detection methods into a weak supervision-based learning framework. One can view this approach to multilingual text detection as an ensemble of different CV-based techniques and DL approaches.* TEXTRON *can leverage the predictions of DL models pre-trained on a significant amount of language data in conjunction with CV-based methods to improve text detection in other languages. We demonstrate that* TEXTRON *can improve the detection performance for documents written in Indian languages, despite the absence of corresponding labeled data. Further, through extensive experimentation, we show improvement brought about by our approach over the current State-of-the-art (SOTA) models, especially for handwritten Devanagari text. Code and dataset has been made available at* https://github.com/IITB-LEAP-OCR/TEXTRON

1. Introduction

Text detection aims to localize the textual information present in images. Text detection from documents is an essential and one of the first steps in Optical Character Recognition (OCR). With Deep Learning(DL) methods replacing conventional approaches in several fields, text detection also benefits from such DL techniques including various convolutional neural networks (CNNs) [7], region-based convolutional neural networks (R-CNNs), and fully convolutional networks (FCNs) [34], offering their robust solutions. A major limitation of deep learning methods is the need for ample training data, which can be challenging to obtain, especially in resource-limited situations like identifying new languages without readily available datasets. Further, as demonstrated in Figure 1a, it is seen that pre-trained DL-based methods (like DBNet [18]) are not able to capture different styles of handwriting or word level demarcation for multilingual scripts (such as those of Indian languages). As opposed to this, certain image processing-based techniques like contour-based techniques [20] which work on the pixel level, can work well to capture these different writing styles by engulfing concerned characters and nuances as illustrated in Figure 1b. A lot of CV-based techniques that work on the principle of edge detection, such as Sobel Filter Usage [14] or Canny Filter Application [4] have prevailed for a few decades. Since these CV-based methods that work on pixel-level classification are more adaptive to different styles and fonts of text, they can be considered insightful in contributing to the effective detection of diverse text. However, unlike DL-based methods, CV-based methods are sensitive to noise and may not be able to unify the complete word in one bounding box as can be seen in Figure 1b. This can drastically affect the performance of visually rich or scanned documents. Consequently, we propose integrating the robustness and noise immunity of DL-based methods with the diversity-capturing ability of CV-based techniques, leading to the introduction of TEXTRON. As shown in Figure 1c, TEXTRON can help accurately identify word-level boxes by combining the strengths



(a) DL-Based Method (b) CV-based method (c) Using TEXTRON

Figure 1. A comparative overview of different text detection approaches. The green boxes in subsequent figures highlight true positives while the red ones represent incorrect detections that either engulf more than two words in a single bounding box or fragment a single word into multiple boxes.

of both the aforementioned kinds of approaches (CV-based and DL-based). TEXTRON can help in the effective aggregation of different text detection methods (weak labels) and can also handle scenarios with scarcity or absence of labeled data. This aggregation of different text detection methods is different from an ensemble approach, because in the latter, there is a need to use two or more models to be fitted on the same labeled data, and then the predictions of each model are combined. Contrary to this, TEXTRON tries to eliminate dependency on labeled data. In an attempt to incorporate different CV-based techniques, TEXTRON also works on the document images on a pixel-based classification level that helps in capturing the subtlety of text. Thus, through TEXTRON , we can bring about effective multilingual text detection which is not very sensitive to noise and at the same time helps capture a rich diversity of text written in different languages and modalities (such as handwritten *vs.* printed). In addition to this, it also reduces and often eliminates the need to train new DL models and does not rely on labeled data. We highlight recent work in text detection and the role of weak supervision associated with it in Section 2. We also describe our methodology and associated unsupervised experimentation in Sections 3 and 4 respectively. We further demonstrate the usage of TEXTRON and report the relevant results in Section 5. Finally, we conclude with opening doors to more innovative contributions bringing about seamless multilingual text detection.

2. Related Works

Deep learning-based text detection has garnered recent attention. Such DL-based text detection in images can be either bounding box-based or pixel-based. Bounding box-based text detection takes inspiration from Object Detection approaches such as YOLO [27], SSD [19], and Faster RCNN [28]. On the other hand, pixel-based text detection methods include the approaches inspired by Mask RCNN [9] and FCN [21] which include CRAFT [3], PSENet [17], etc. Differential Binarization or DB-Net [18] is one of the recently introduced approaches that work at the pixel level followed by a label generation process to retrieve word-level bounding boxes. DBNet is a segmentation-based detection network with a RESNET [10] backbone making it faster, more accurate, and lightweight. Other segmentation-based text detection methods include LinkNet [6] which attempts to exploit parameters utilization of neural networks efficiently. Due to the optimization in parameter usage, LinkNet helps to decrease the processing time required for text detection. Unsupervised segmentation approaches have also been employed generically to classify regions of an input image and assign labels to them. One such unsupervised approach [13] leverages a pre-trained CNN model to assign labels to pixels. However, the number of unique labels should be sufficiently large to leverage this unsupervised segmentation technique. Computer Vision (CV) based text detection includes a variety of algorithms such as edge detection [14], connected component analysis [11, 15], Stroke Width Transform (SWT) [25], and contour-based methods [20]. Edge detection methods include the usage of the Sobel filter [14] method to find the corresponding edges of the textual information present in a binarized map of the input document image. Further, the canny [4] algorithm to detect regions based on edges can also help perform text detection. Recently, to address the scarcity of labeled data, there has been an emergence of weak-supervision and semi-supervision-based text detection in documents. Text-as-Lines [32] proposes a scene text detector based on weakly supervised learning that helps in the annotation process. This method uses coarse line-level pixel-based masks to denote text lines in documents. The usage of coarse masks as opposed to full pixel-level masks makes the process robust and faster to train. We-Text [31] exploits the paradigm of both weak supervision and semi-supervision to eventually train models to perform scene text detection. It follows a character-based approach to handle multilingual as well as multi-oriented text. Various semi-supervised and unsupervised data programming approaches [2, 22, 26, 33] have proven to be useful in creating labeled data. In our work, we leverage the data programming paradigm of weakly supervised learning as in CAGE [5] that also provides domain-based quality guides on the labeling functions [23].

3. Our Methodology

We begin by explaining the notion of labeling functions that get used for data programming.

3.1. Labeling Functions

Labeling Functions (LFs) [33] are conventionally weak supervision functions that generate noisy labels. In our context, LFs mark regions of document image as TEXT or NON-TEXT regions. In our setting, the LFs operate at the pixel level by assigning corresponding labels to each pixel. Following are some of the LFs used in our framework to identify TEXT regions. The output predictions from each of the LF would be a binarized pixel map. Each LF is associated with one of the two classes: TEXT for labeling a textual region in the document and NON-TEXT otherwise. While some LFs are based on conventional Computer Vision (CV) techniques, others make use of DL-based models:

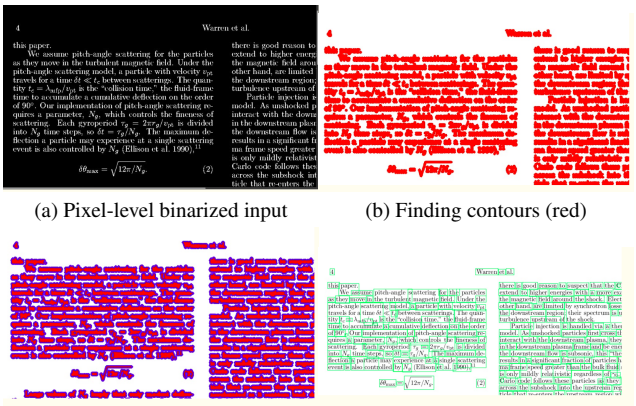
1. **Labeling Functions from Pretrained DL Models**

   (a) **DBNet [18]** is pre-trained for English[1] and uses neural network segmentation-based method for text detection. Given a document image, DBNet internally generates a binarized pixel map which is then processed to generate the word-level bounding boxes. Based on these bounding boxes, this LF gives a binarized pixel map as its output as shown in Fig 3b

2. **CV-based Labeling Functions**

   (a) **Contour-based LF**: Contour-based methods can determine word-level regions on a binarized pixel map by demarcating pixels having the same value. Given an input image (shown in Fig 3a), we binarize the image and find contours twice with an intermediate process of drawing contours[2]. This is finally used to get the pixel-level binary map. The process to obtain bounding boxes is pictorially depicted in Figure 2. The binarized output pixel map is shown in Fig 3c.

   (b) **Canny Filter-based LF**: This function detects edges along the words by using the canny [4] filter over the input image. Later the output of detected edges is fed into the contour-based post-processing with certain edge thickness [3] as shown in Fig 2 to retrieve bounding boxes and obtain the final binarized output as shown in Figure 3d.



(a) Pixel-level binarized input (b) Finding contours (red)

(c) Drawing contours over the top of red contours (d) Get bounding boxes from blue contours

Figure 2. Workflow for Contour-based processing to get word level bounding boxes

   (c) **Tesseract [30]**: The image-to-data method provided by Tesseract[4] can also determine word-level bounding boxes for text detection which is used to create a binarized output map. The output for the same is shown in Fig 3e

   (d) **Image Edges based LF**: As shown in Figure 3f, this function uses the Sobel filter [14] method to find the corresponding edges of the textual information present in a binarized input document image. The binary map is again fed to contour-based processing [5] to obtain bounding boxes.

Similar to the five LFs described above, we also define and use the corresponding five complementary LFs which are associated with the NON-TEXT class. Thus, we have designed ten LFs out of which, the complementary LFs use the same algorithm proposed in the above descriptions. However, the only difference is that they help to label the concerned set of pixels with NON-TEXT class. The reason to introduce the complementary LFs lies in the fact that currently, our method limits one labeling function to label entities for only a single class. The outputs for such complementary LFs will be the complement of what is shown in Figure 3 and will label the non-textual pixels of the input page image. The workflow of TEXTRON is presented in Figure 4 which highlights the application of LFs followed by aggregation of LF outputs to produce TEXTRON output, which is also a binarized pixel map (Y). Before proceeding to further details, we introduce two standard processes working in the periphery for the creation of word-level bounding boxes from the retrieved pixel-based TEXTRON output.

[1] https://mindee.github.io/doctr/latest/modules/models.html#doctr.models.detection.db_resnet50

[2] with thickness 4, which is a hyperparameter for this LF

[3] used as a hyperparameter for this LF

[4] https://github.com/tesseract-ocr/tessdoc

[5] with thickness 2, a hyperparameter for this LF

(a) Input Image (b) DBNet based LF (c) Contour based LF (d) Canny Filter LF (e) Tesseract based LF (f) Image Edges LF

Figure 3. Binary Image Outputs for different Labeling Functions

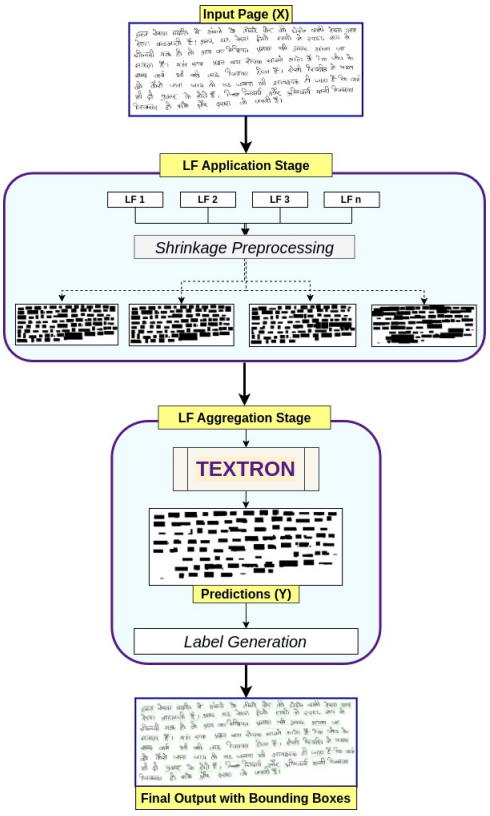


Figure 4. A step-by-step description of the TEXTRON workflow

3.1.1 Shrinkage Factor Preprocessing

The final pixel-level prediction (Y) obtained in Figure 4 is in the form of a binary map. Often, in such an output, we observe cases with overlapping bounding boxes as depicted in Figure 5a. The word level demarcation gets lost in these overlapping bounding boxes. Since it is preferable to get the output in bounding boxes format, we need to tweak the LFs in such a way that we are able to retrieve the word level demarcation again through the output. To achieve this, we shrink the width and height of the bounding boxes provided by each LF by a shrinkage factor (set to be a hyperparameter) to provide a binary map containing

(a) Original Result without shrinking dimesnions (b) With 20% Shrinkage of both height and width (c) With 40% Shrinkage of both height and width

Figure 5. Demonstration of Shrinkage Factor Preprocessing

disjoint non-overlapping bounding boxes. This encourages our TEXTRON model to fit and learn on shrunk bounding boxes. This process, in turn, provides a similar output map as seen in Figures 5b and 5c respectively. These figures show the effect on the overall results produced by shrinking heights and widths of bounding boxes with different values of shrinkage factors. The benefit of performing shrinkage becomes evident while recovering the original and independent word-level bounding boxes during the label generation step and this is explained in the following Section 3.1.2.

3.1.2 The Label Generation Logic

The generated binarized pixel map (Y from Figure 4) is subjected to a simple post-processing step to convert the pixel-level data into word-level bounding boxes. To achieve this, we use the contour-based[6] demarcation method. After retrieving the corresponding bounding boxes, their widths and heights are recovered by enlarging them to the same extent by which they were shrunk in the LF application stage. In Figure 6, we present the stepwise workflow of label generation in order to retrieve the final predictions. Through this step, TEXTRON is able to provide a sequence of bounding boxes as its output which can be further used for feedback provision or evaluation.

3.2. Our Framework

As mentioned in Section 1, our framework TEXTRON tries to combine different text detection methods together using a data programming paradigm. The objective is to

[6]Cv2 contours https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

(a) Textron Predictions (b) Contour-based processing (c) Final Predictions

Figure 6. Contour-based post-processing step for label generation.

train a model that can learn the parameters associated with each LF so as to generate a binarized pixel map that aligns with the consensus of LF outputs. In our framework, the data programmer can guide the training process by providing quality guides to each LF [5], which are then incorporated into a generative model. This is important for our scenario because while designing LFs, a programmer is aware of the cases in which the LF (text detection method giving weak labels) might work efficiently or fail. Let $\mathcal{X} = \{\mathbb{R}\}^{H \times W}$ where $\mathbb{X}$ represents binarized input image and $\mathbb{R}$ represents the intensity or color value of the pixel, H and W represent the height and width of the image respectively. $\mathcal{Y} = \{c1, c2\}$ denotes the binary label space indicating whether or not a pixel represents a text region. $P(\mathcal{X}, \mathcal{Y})$ denotes the joint distribution of pixels and labels. Our goal is to model the relation between a label y with a pixel $x \in \mathcal{X}$. We have *m=H*W* pixels for a given image instance. We have *n* LFs $\lambda_1, \lambda_2, .., \lambda_n$ and each LF λ_i is associated with a label k_i which is one of the labels *c1* (for TEXT) or *c2* (for NON-TEXT). λ_i provides a discrete label $\tau_{ij} = k_j$ when triggered and $\tau_{ij} = 0$ when not triggered. Each LF corresponds to a CV or DL-based algorithm to detect textual and non-textual regions as long as it returns values in the above format. More details on the different LFs used are presented in Section 3.1. Given the different LFs, our goal is to learn to create consensus among the outputs of the LFs. Thus, the model imposes a joint distribution between the true label y and the value τ_{ij} returned by each LF λ_i on any pixel x_j. Since the graphical model (parameterized by θ) is to be trained on unlabeled data, the model should ideally fit only on τ_{ij} without relying on the true label y. Hence, the joint distribution is defined as,

$$P_\theta(y, \tau_i) = \frac{1}{Z_\theta} \prod_{j=1}^{n} \psi_\theta(\tau_{ij}, y) \tag{1}$$

where θ denotes the parameters used in defining the potentials ψ_θ,

$$\psi_\theta(\tau_{ij}, y) = \begin{cases} \exp(\theta_{jy}) & if \tau_{ij} \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

and the normalizer Z_θ will be

$$Z_\theta = \sum_{y \in \mathcal{Y}} \prod_{j} (1 + \exp(\theta_{jy})) \tag{3}$$

Given the above model, the training objective is defined as,

$$\max_\theta LL(\theta \mid D) + R\left(\theta \mid \left\{q_j^t\right\}\right) \tag{4}$$

The first part maximizes the likelihood of the observed τ_i of the pixels in the training samples $D = x_1, ., x_m$ after marginalizing out the true y. It can be expressed as:

$$\begin{aligned} LL(\theta \mid D) &= \sum_{i=1}^{m} \log \sum_{y \in \mathcal{Y}} P_\theta(\tau_i, y) \\ &= \sum_{i=1}^{m} \log \sum_{y \in \mathcal{Y}} \prod_{j=1}^{n} \psi_j(\tau_{ij}, y) - m \log Z_\theta \end{aligned} \tag{5}$$

R is a regularizer that guides the parameters with the programmer's expectation of the quality of each LF. The q_j^t guide (a value between 0 and 1) is the user's belief or confidence in the fraction of the cases where y and t_j agree with $\tau_j \neq 0$. Given q_j^t we seek to minimize the KL divergence between the user-provided q_j^t and the model-calculated precision $P_\theta(y = k_j | \tau = k_j)$ which turns out to be:

$$\begin{aligned} R\left(\theta \mid \left\{q_j^t\right\}\right) = \textstyle\sum_j q_j^t \log P_\theta(y = k_j \mid \tau_j = k_j) \\ + \left(1 - q_j^t\right) \log\left(1 - P_\theta(y = k_j \mid \tau_j = k_j)\right) \end{aligned} \tag{6}$$

In a similar manner, we proceed with the training by loading another unlabeled image, keeping the θ unchanged, and optimizing the same training objective mentioned in equation (4). It is during the provision of these unlabeled images that we can try to make up for the scarcity or absence of labeled data such as by providing unlabeled document page-level images having contents written in low-resource languages or handwritten Indian languages. After an acceptable number of image iterations, we save the model configuration which includes the LFs and their corresponding θ parameters. During the inference stage, we provide this model with quality guides and LF hyperparameters to perform text detection on the input image.

For detailed motivation for the above equations, we refer the reader to the work of Chatterjee *et. al.* [5]. Intuitively, given an image with m pixels and n LFs (and its corresponding q_j^t quality guides), we train a graphical model by maximizing the log-likelihood training objective specified in equation (4).

4. Experiments

To demonstrate the efficacy of the proposed framework, we ran extensive experiments. We begin by describing the datasets used and then present our experimental setup.

4.1. Datasets

To compare the results in both the supervised and unsupervised settings, we selected two types of datasets. Publicly available datasets for the English language and text detection datasets for a few Indian languages for which there are no pre-trained models available, to the best of our knowledge.

4.1.1 English Datasets

Docbank [16] is a publicly available dataset on which we benchmark our experimental results. The dataset contains around 500,000 page images each of which has corresponding annotations in the form of bounding boxes. Each bounding box is associated with a class. For our purpose, out of the 12 Docbank classes, we excluded the figure class (*i.e.*, the 'figure' class will be considered as any other non-textual region) and considered the remaining 11 classes as the target, namely Abstract, Author, Caption, Equation, Footer, List, Paragraph, Reference, Section, Table, and Title. All regions belonging to the aforementioned classes are considered to be TEXT regions. Out of the 500,000 images, we used around 22,000 images for training, validation, and unsupervised assessment of LFs. We also use a sample test set of 100 images having 55,223 bounding boxes representing text class on which we report our results. We also use 199 images from Funsd [12] and 200 random images from the CTDAR dataset [8] for training and validation purposes.

4.1.2 Indian Language Text Datasets

We have created a benchmark annotated dataset of printed document images in three Indian languages which we use as a test set. This includes 200 pages of Malayalam (48,358 words), 225 pages of Tamil (48,239 words), and 323 pages of Gujarati (82,475 words) documents. The Gujarati document pages also contain a considerable amount of Sanskrit text. We have collected the required pages from textbooks [24] and other books [1]. Word-level bounding boxes are marked for every page-level image by a group of qualified annotators. Apart from that, we also split and used around 300 unlabeled Sanskrit images from a set of books [1] for training and validation respectively. The PHD Indic 11 dataset [29] is composed of handwritten document images of 11 official Indian language scripts. We have used 220 page-level images of handwritten Devanagari documents as our test set for which bounding boxes are marked by a group of verified annotators. We also use unlabeled document images of Telugu text (85 pages) and Kannada text (46 pages) from the PHD Indic dataset [29] for training, validation, and unsupervised analysis.

4.2. Unsupervised Quantitative Assessment of LFs

While choosing the best-performing TEXTRON LF set, we performed extensive experiments using different combinations of CV-based LFs as well as the LFs derived from pre-trained DL models. We analyzed the unsupervised performance measures of the ten LFs designed as mentioned in Section 3.1. These measures include coverage, overlaps, and conflicts for each LF. Recall from Section 3.1 that each LF can either get triggered or abstain from labeling a pixel. The fraction of pixels (with respect to the complete set of pixels of the image) an LF can label is what we refer to as the *coverage* of each LF. Similarly, the *overlap* of an LF is defined as the fraction of pixels out of all covered (labeled) pixels assigned the same label by at least one more LF. As opposed to that, we define the *conflict* of an LF as the fraction of pixels with respect to the overlapped pixels that do not match the label assigned by any other LFs. Ideally, a good LF will have significant coverage, high overlap, and fewer conflicts. While experimenting we saw that the Image Edge-Based LF had an extremely low coverage for TEXT pixels. Also, the corresponding complementary LF had high conflicts. So we dropped this LF and its corresponding complementary LF from the best-performing set and analyzed further for the remaining eight LFs. These eight LFs included four fundamental LFs namely pre-trained DBNet, Tesseract-based LF, Contour-based LF, and Canny Filter-based LF along with their complementary LFs. Figure 7 presents the three unsupervised performance measures for these LFs on a random image of the Docbank dataset. The fundamental LFs that labeled the text pixels had less (yet significant) coverage which was a favorable scenario as the amount of TEXT pixels in any generic document page is much less as compared to the NON-TEXT pixels. Similarly, all our chosen complementary LFs have a high percentage of coverage and overlap, both of which are above 80% for almost every image. We have performed a similar kind of analysis with various unlabeled images with handwritten and printed Indian language text training sets as well. All kinds of analysis and visual inspection of outputs (presented in the supplementary) have pointed in favor of usage of TEXTRON_{8LF}. With this newly defined TEXTRON_{8LF} configuration, we train our graphical model (parameterized by θ) for the training objective described in equation (4).

4.3. Training and Validation

After unsupervised experimentation with various combinations of LFs on the training sets described in Section 3.1, we trained the TEXTRON_{8LF} configuration. As mentioned in Section 3, we try to optimize the training objective for 50 epochs by keeping a constant learning rate of 0.01 for each image in the train set. We further experimented on our various validation sets described in Section 4.1. This included tweaking the LF quality guides and their hyperparameters

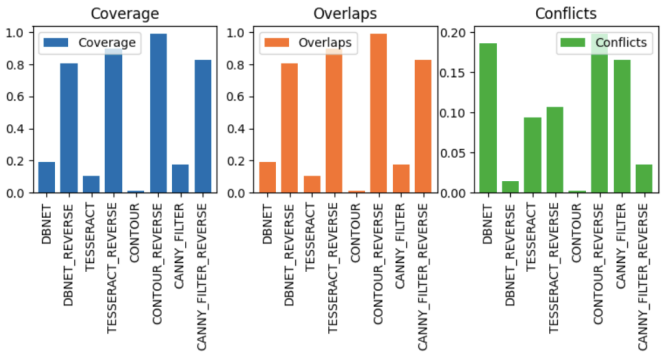
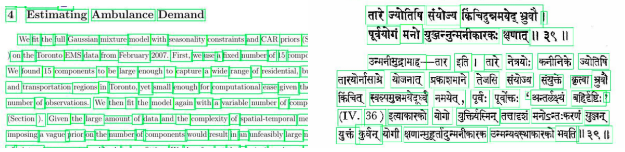


Figure 7. Unsupervised quantitative assessment on the eight LFs included in the experimentation displaying coverage, overlaps, and conflicts for each LF



(a) Docbank Image (b) Printed Sanskrit

Figure 8. Predictions during TEXTRON$_{8LF}$ Learning Phase

for an already trained TEXTRON$_{8LF}$ model. Hyperparameter tuning was performed on curated validation sets of 125 images (English), 85 images (Printed Sanskrit), and 90 images (Kannada and Telugu handwritten) to determine the best thickness and shrinkage parameters. Figure 8 shows our final word-level bounding boxes output on random images from the validation set produced by TEXTRON$_{8LF}$. Once the model was able to perform well through hyperparameter tuning for a sufficiently large number of images, we eventually used this saved TEXTRON$_{8LF}$ model and tuned hyperparameters for inference on unseen test sets.

5. Results and Discussions

In this section, we present the performance of our TEXTRON$_{8LF}$ model using various combinations of quality guides and hyperparameters on the various test sets described in Section 4.1.

5.1. Baselines

We use the current state-of-the-art DBNet [18] model for text detection as our baseline. Additionally, we also use Majority Based Voting (MBV) as a baseline to compare our performance. The MBV baseline performs pixel-based voting through exactly the same combination of LFs (taking into account a pair of both fundamental LF and its corresponding complementary LF) used by TEXTRON$_{8LF}$. Each LF will cast a vote for a pixel either as TEXT or NON-TEXT. Finally, if the number of TEXT votes for a pixel exceeds the NON-TEXT votes, then the pixel is assigned TEXT class. In this manner, MBV produces a binarized image that undergoes contour-based post-processing as described in Section 3.1.2 to obtain the bounding boxes. We evaluate TEXTRON$_{8LF}$ and the baselines using an IOU-based approach to determine the overall precision, recall, and F1 scores.

Dataset	Height Shrunk By	Contour Thickness	Edge Thickness
Docbank	20%	4	2
Malayalam	20%	5	4
Tamil	20%	4	3
Gujarati	20%	4	3
Devanagari	30%	5	5

Table 1. Hyperparameters of TEXTRON$_{8LF}$ for different datasets

Approach	P	R	F
DBNet	**90.49**	80.00	84.92
Tesseract	40.49	74.03	52.35
MBV	89.53	80.02	84.51
TEXTRON$_{8LF}$	90.17	**80.75**	**85.21**

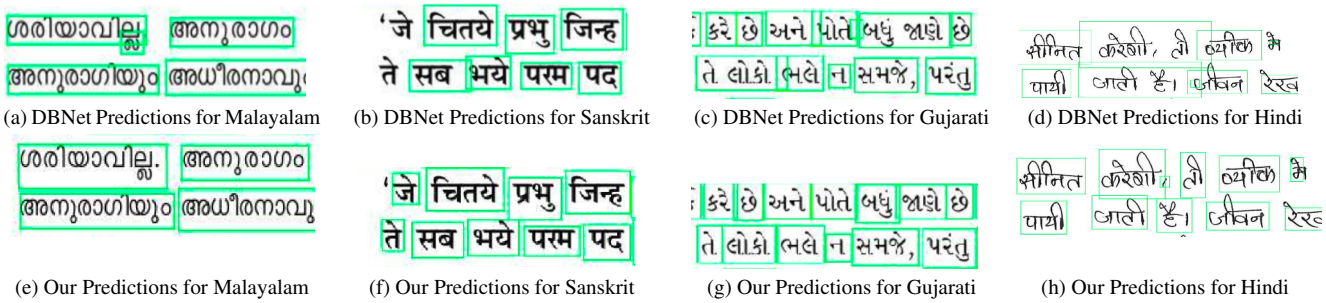
Table 2. Results with IOU 0.5 on Docbank 100 Test Samples

5.2. Hyperparameters for Inference

In this section, we mention the best-performing set of hyperparameters for TEXTRON$_{8LF}$ that have been tuned on validation sets. The fundamental four LFs used namely DBNet-based LF, Contour-based LF, Tesseract-based LF, and Canny Filter-based LF have quality guides set to 0.9, 0.85, 0.75, and 0.85 respectively. Quality guides of the complementary LFs are set to be 0.95 each. For this configuration, we apply 10% width shrinkage as the preprocessing during the LF application stage. Additionally, the hyperparameters of *height shrinkage*, *contour thickness*, and *edge thickness* were determined by tuning on different Indian language validation sets (described in Section 4.1). They are mentioned in Table 1. Using these TEXTRON$_{8LF}$ configurations, we infer and report the following results on unseen test sets.

5.3. Performance on Different Datasets

In this section, we evaluate and present the performance of TEXTRON$_{8LF}$ on different test sets. Table 2 presents a comparison of TEXTRON$_{8LF}$ along with other baselines for an IOU threshold of 0.5 on the Docbank test set described in Section 4.1.1. We also highlight the performance of the annotated Indian language text detection datasets described in Sections 4.1.2. We have used TEXTRON$_{8LF}$ configuration for carrying out the inference on these pages. We report our performance on all four language test sets along



(a) DBNet Predictions for Malayalam (b) DBNet Predictions for Sanskrit (c) DBNet Predictions for Gujarati (d) DBNet Predictions for Hindi

(e) Our Predictions for Malayalam (f) Our Predictions for Sanskrit (g) Our Predictions for Gujarati (h) Our Predictions for Hindi

Figure 9. Comparative Overview of DBNet results and our TEXTRON outputs for different kinds of test images

Language	Printed Malayalam			Printed Tamil			Printed Gujarati			Handwritten Devanagari		
Approach	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
DBNet	97.97	99.38	98.67	98.90	99.59	99.24	98.40	96.84	97.62	**72.84**	62.33	67.17
Tesseract	81.39	90.96	85.91	95.23	95.34	95.28	86.72	84.06	85.37	60.29	65.89	62.97
MBV	**99.71**	99.35	**99.53**	99.29	99.25	99.27	95.39	**98.38**	96.86	57.05	68.76	62.36
TEXTRON$_{8LF}$	99.03	**99.44**	99.23	**99.32**	**99.64**	**99.48**	**98.57**	97.65	**98.11**	69.24	**74.51**	**71.78**

Table 3. Results with IOU 0.5 on Indian Language Text Detection Test Sets

with other baselines in Table 3. We can see that our approach has the highest recall for Malayalam text detection and is at par with the best-performing baseline. On the other hand, TEXTRON$_{8LF}$ gives the best overall performance on both Tamil and Gujarati text detection. The improvement brought about by TEXTRON$_{8LF}$ is visible in Figure 9. Further, we also present our handwritten Devanagari text detection performance in Table 3. The improvement in case of TEXTRON$_{8LF}$ for this handwritten text detection is substantially high. This is also visually depicted in Figures 9d and 9h respectively. Figure 10 shows that TEXTRON$_{8LF}$ is able to maintain the best performance for Devanagari text detection even for higher IOU thresholds. We also highlight insights on the performances for other IOU scenarios and different classes of text (presented in the supplementary). This is beneficial for several downstream applications like handwritten text recognition.

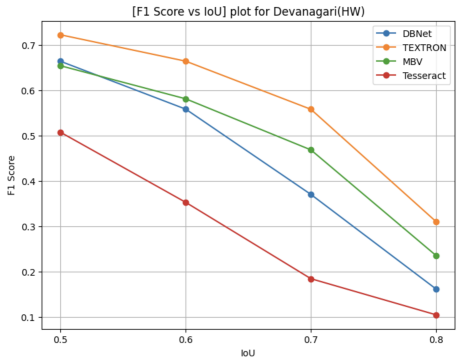


Figure 10. TEXTRON performance against other baselines with IOU thresholds on Handwritten Devanagari Text Detection

6. Conclusion and Future Work

We present TEXTRON, a weak supervision-based approach that provides efficient results for text detection in multilingual documents. We demonstrate the flexibility of TEXTRON by enabling the use of customized LFs to enhance text detection. The proposed approach is better off visually in addressing handwritten text and also near to state-of-the-art in other printed modalities. Our results also highlight that TEXTRON can effectively be used to perform text detection in Indian language printed and handwritten documents. This also indicates its versatility by performing well even in the absence or paucity of labeled data. Our work paves the way in the direction of improving script-specific or font-specific text detection in documents. Plug and play with a set of customized LFs, setting intuitive quality guides, and tweaking hyperparameters can bring about enhanced text detection in different types of documents. Further work includes incorporating the detection of document classes other than text such as figures, tables, and equations. It will also be beneficial to expand the scope of our framework to detect text ranging from different kinds of handwritten documents to natural scenes, by designing suitable labeling functions.

Acknowledgements

We acknowledge the support of a grant from IRCC, IIT Bombay, and MEITY, Government of India, through the National Language Translation Mission-Bhashini project.

References

[1] *Tamil and Malayalam Versions*. Yatharth Geeta Indian Languages, 2018. 6

[2] Guttu Abhishek, Harshad Ingole, Parth Laturia, Vineeth Dorna, Ayush Maheshwari, Ganesh Ramakrishnan, and Rishabh Iyer. Spear: Semi-supervised data programming in python. In *Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–127, 2022. 2

[3] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. *CoRR*, abs/1904.01941, 2019. 2

[4] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. 1, 2, 3

[5] Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. Robust data programming with precision-guided labeling functions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3397–3404, Apr. 2020. 2, 5

[6] Abhishek Chaurasia and Eugenio Culurciello. LinkNet: Exploiting encoder representations for efficient semantic segmentation. In *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, dec 2017. 2

[7] Manolis Delakis and Christophe Garcia. text detection with convolutional neural networks. In *VISAPP (2)*, pages 290–294, 2008. 1

[8] Liangcai Gao, Yilun Huang, Hervé Déjean, Jean-Luc Meunier, Qinqin Yan, Yu Fang, Florian Kleber, and Eva Lang. Icdar 2019 competition on table detection and recognition (ctdar). In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1510–1515. IEEE, 2019. 6

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[11] Pengfei Hu, Yang Chen, Yusheng Hao, Yiqun Wang, and Weilan Wang. Text line segmentation based on local baselines and connected component centroids for tibetan historical documents. In *Journal of Physics: Conference Series*, volume 1656, page 012034. IOP Publishing, 2020. 2

[12] Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. Funsd: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 1–6, 2019. 6

[13] Asako Kanezaki. Unsupervised image segmentation by backpropagation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018. 2

[14] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2):358–367, 1988. 1, 2, 3

[15] Hyung Il Koo and Duck Hoon Kim. Scene text detection via connected component clustering and nontext filtering. *IEEE transactions on image processing*, 22(6):2296–2305, 2013. 2

[16] Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. DocBank: A benchmark dataset for document layout analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 949–960, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics. 6

[17] Xiang Li, Wenhai Wang, Wenbo Hou, Ruo-Ze Liu, Tong Lu, and Jian Yang. Shape robust text detection with progressive scale expansion network. *CoRR*, abs/1806.02559, 2018. 2

[18] Minghui Liao, Yan Wan, Cong Yao, Kai Chen, and Xiang Bai. Real-time scene text detection with differentiable binarization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:11474–11481, 04 2020. 1, 2, 3, 7

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. 2

[20] Yangxing LIU, Satoshi Goto, and Takeshi Ikenaga. A contour-based robust algorithm for text detection in color images. *IEICE Transactions on Information and Systems*, E89D, 03 2006. 1, 2

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. 2

[22] Ayush Maheshwari, Oishik Chatterjee, KrishnaTeja Killamsetty, Rishabh K. Iyer, and Ganesh Ramakrishnan. Data programming using semi-supervision and subset selection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021. 2

[23] Ayush Maheshwari, Krishnateja Killamsetty, Ganesh Ramakrishnan, Rishabh Iyer, Marina Danilevsky, and Lucian Popa. Learning to robustly aggregate labeling functions for semi-supervised data programming. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1188–1202, 2022. 2

[24] School of Distance Education. *Malayalam: Language, Culture and Literature*. University Of Kerala, 2017. 6

[25] Il-Seok Oh and Jin-Seon Lee. Smooth stroke width transform for text detection. In Christo Dichev and Gennady Agre, editors, *Artificial Intelligence: Methodology, Systems, and Applications*, pages 183–191, Cham, 2016. Springer International Publishing. 2

[26] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré . Snorkel. *Proceedings of the VLDB Endowment*, 11(3):269–282, nov 2017. 2

[27] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 2

[28] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 2

[29] Obaidullah Sk, Chayan Halder, KC Santosh, Nibaran Das, and Kaushik Roy. *PHDIndic 11, page-level handwritten doc-*

ument image dataset of 11 official Indic scripts. IEEE Dataport, 2021. 6

[30] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007. 3

[31] Shangxuan Tian, Shijian Lu, and Chongshou Li. Wetext: Scene text detection under weak supervision. *CoRR*, abs/1710.04826, 2017. 2

[32] Weijia Wu, Jici Xing, Cheng Yang, Yuxing Wang, and Hong Zhou. Texts as lines: Text detection with weak supervision. *Mathematical Problems in Engineering*, 2020:1–12, 06 2020. 2

[33] Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. A survey on programmatic weak supervision, 2022. 2, 3

[34] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. Multi-oriented text detection with fully convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4159–4167, 2016. 1