USTH
VIETNAM FRANCE UNIVERSITY

**Lecturer: Prof. Nguyen Duc Dung**
*Computer Vision*

# COMPUTER VISION FINAL PROJECT

## Total-Text Scene Text Recognition: A Two-Stage Approach with U-Net Segmentation and Transformer-Based Recognition


**MEMBERS:**

NGUYEN THE KHAI — 23BI14205
TRAN GIA KHANH — 23BI14218
NGO XUAN KIEN — 23BI14239
NGO MINH PHUOC — 23BI14361


2026

# CONTENTS

1 Introduction 5
1.1 Contributions . . . 5
1.2 Report Organisation . . . 6
2 Background and Related Work 6
2.1 Traditional Scene Text Detection . . . 6
2.2 Deep Learning for Text Detection . . . 6
2.3 U-Net for Segmentation . . . 6
2.4 Instance Segmentation with Mask R-CNN . . . 7
2.5 Scene Text Recognition . . . 7
2.6 End-to-End Systems . . . 7
3 Dataset and Evaluation 7
3.1 Total-Text Dataset . . . 7
3.2 Evaluation Metrics . . . 9
4 Preprocessing and Data Preparation (Detection Pipeline) 10
4.1 Ground-Truth Mask Generation . . . 10
4.2 Image Resizing . . . 10
4.3 Data Augmentation . . . 11
5 Stage 1: Text Region Segmentation 11
5.1 U-Net Architecture . . . 11
5.2 Training Details . . . 12
6 Stage 2: Polygon Extraction from Predicted Masks 13
6.1 Contour Detection . . . 13
6.2 Polygon Approximation . . . 13
6.3 Limitations of Contour-Based Polygon Extraction . . . 14
7 Stage 3: Orientation Classification 14
7.1 Problem Formulation . . . 14
7.2 Dataset Construction . . . 14
7.3 Classifier Architecture . . . 15
7.4 Training Details . . . 15
7.5 Full Inference Pipeline . . . 15
8 Text Recognition Pipeline: Mask R-CNN and TrOCR 15
8.1 Stage 1: Instance Segmentation with Mask R-CNN . . . 16
8.2 Stage 2: Text Recognition with TrOCR . . . 16
8.3 Evaluation Adaptation . . . 17
9 Results and Discussion 17
9.1 Detection Pipeline (U-Net) . . . 17
9.2 Recognition Pipeline (Mask R-CNN + TrOCR) . . . 19
9.3 Comparative Analysis . . . 19
9.4 Error Analysis . . . 19
10 Broader Discussion 20
10.1 The Divide-and-Conquer Trade-off . . . 20
10.2 Challenges Specific to Curved Text . . . 20
10.3 Data Scale and Pre-training . . . 20
10.4 Evaluation Protocol Considerations . . . 20
11 Conclusion 21

# LIST OF FIGURES

Figure 1 Total-Text Dataset: Annotation Types . . . 7
Figure 2 Total-Text Dataset: Label Distribution . . . 8
Figure 3 Total-Text Dataset: Annotation Types . . . 10

Figure 4 Total-Text Dataset: Data Augmentation . . . . . 11
Figure 5 Total-Text Dataset: Polygon Vertex Count Distribution . . . . . 14
Figure 6 Total-Text Dataset: Sample visualization . . . . . 18
Figure 7 Total-Text Dataset: Sample visualization with transcription . . . . . 19

## LIST OF TABLES

Table 1 Total-Text dataset statistics. . . . . 9
Table 2 U-Net architecture summary for Stage 1 segmentation. . . . . 12
Table 3 Stage 1 segmentation training results . . . . . 13
Table 4 Stage 3 orientation classifier training results. . . . . 15
Table 5 Detection pipeline performance on Total-Text test set under the DetEval protocol. 17
Table 6 Recognition pipeline performance on Total-Text test set. . . . . 19
Table 7 Summary comparison of both pipelines on the Total-Text test set. . . . . 19

## ABSTRACT

Scene text recognition in natural images is a challenging computer vision task, especially when text appears in arbitrary orientations and curved configurations, as is common in real-world photographs. This report presents our work on the Total-Text dataset, a benchmark specifically designed to evaluate detection and recognition of curved, multi-oriented, and irregular text.

Our approach divides the end-to-end recognition pipeline into two complementary contributions. The first contribution focuses on text region detection: we train a U-Net segmentation model from scratch to produce binary text region masks, extract variable-length polygons from those masks via contour approximation, and classify each polygon's orientation (curved, horizontal, or vertical) using a lightweight convolutional classifier. The second contribution addresses text transcription: a fine-tuned Mask R-CNN instance segmentation model is used to localise text instances at the pixel level, and a fine-tuned TrOCR model provides end-to-end transformer-based recognition of the cropped text regions.

Our detection-only pipeline achieves a DetEval F-score of 0.5461 (Precision 0.5129, Recall 0.5839) on the Total-Text test set. The full pipeline including transcription achieves an IoU-based F1 of 0.4361 and a DetEval H-mean of 0.3304. These results demonstrate the feasibility of the divide-and-conquer strategy while also highlighting the challenges posed by curved text and overlapping instances.


# 1 INTRODUCTION

Reading text from natural images — a task often referred to as *scene text recognition* — underpins many practical applications: autonomous vehicles reading road signs, assistive tools for the visually impaired, document digitisation, product label scanning, and image-based search. While text detection and recognition in scanned documents or constrained environments has been largely solved, the problem in unconstrained natural images remains an open research challenge.

The primary source of difficulty is the diversity of text appearance in the wild. Text may appear in arbitrary fonts, sizes, and colors, printed on surfaces with complex backgrounds, or subject to motion blur, perspective distortion, and partial occlusion. Most critically, text in real scenes is not restricted to horizontal lines: it may curve along a surface, spiral around an object, or be oriented vertically. Standard datasets such as ICDAR 2013 and ICDAR 2015 focus predominantly on horizontal or slightly tilted text, and methods trained on them degrade significantly when confronted with curved or irregular text.

The Total-Text dataset [1] was introduced specifically to benchmark systems on multi-oriented and curved text. Each text instance is annotated with a free-form polygon (variable number of vertices), an orientation label (*horizontal*, *vertical*, or *curved*), and a word-level transcription. This combination of polygon annotation and orientation labels makes Total-Text a particularly demanding benchmark that requires a system to simultaneously solve three sub-tasks:

1. Localising text at the pixel or region level
2. Extracting a tight polygon boundary around each word
3. Reading the transcription within that polygon.

Building a single end-to-end model that jointly addresses all three sub-tasks from scratch is prohibitively expensive given our computational constraints. We therefore adopt a divide-and-conquer strategy, splitting the pipeline into two self-contained contributions that are described and evaluated separately, then discussed together.

## 1.1 Contributions

The two contributions of this project are as follows.

**Contribution 1 — Detection and orientation (this report, Sections 4–7):** We design a complete detection pipeline built from scratch. A U-Net architecture is trained on Total-Text training images to

produce binary text-region masks. A classical contour extraction algorithm then converts each mask into a variable-length polygon. A separate lightweight convolutional classifier is trained to assign one of three orientation labels (c, h, v) to each detected polygon. The complete pipeline is evaluated on Total-Text test images using the DetEval protocol.

**Contribution 2 — Full recognition pipeline (Sections 8–9):** We fine-tune two pre-trained models on Total-Text: a Mask R-CNN instance segmentation model (Detectron2 framework) for polygon-level text localisation, and a TrOCR transformer model for word-level transcription. The combined system is evaluated on both IoU-based metrics and the DetEval protocol.

### 1.2 Report Organisation

Section 2 reviews related literature on scene text detection and recognition. Section 3 describes the Total-Text dataset and evaluation metrics. Sections 4–7 detail the U-Net detection pipeline. Section 8 describes the Mask R-CNN and TrOCR recognition pipeline. Section 9 presents evaluation results for both pipelines, and Section 10 discusses findings and limitations. Section 11 concludes.

## 2 BACKGROUND AND RELATED WORK

### 2.1 Traditional Scene Text Detection

Early methods for text detection relied on hand-crafted features and classical machine learning. Connected component methods such as MSER (Maximally Stable Extremal Regions) [5] and Stroke Width Transform (SWT) [6] identify candidate character regions by exploiting the observation that text strokes have relatively uniform width. These components are then grouped by spatial proximity and geometric consistency.

While effective on clean, high-contrast images, such methods degrade rapidly in the presence of complex backgrounds, low contrast, and critically, curved text, where character grouping assumptions break down.

### 2.2 Deep Learning for Text Detection

The introduction of deep convolutional networks revolutionised scene text detection. Fully Convolutional Networks (FCN) [7] provided the first deep segmentation approach applicable to text. TextBoxes [8] adapted single-shot detectors with long anchor boxes for text. EAST (Efficient and Accurate Scene Text Detector) [9] introduced a direct regression approach that predicts rotated bounding boxes at each pixel.

**Segmentation-based approaches** became dominant as they naturally handle arbitrary polygon shapes. PixelLink [10] predicts both text/non-text labels and pixel linkages to separate adjacent instances. PSENet [11] generates progressively expanding kernels to distinguish touching instances. DBNet [12] introduces a differentiable binarisation module that learns a dynamic threshold map.

**For curved text specifically**, TextSnake [13] represents text instances as a sequence of overlapping disks along the text center-line, allowing it to capture arbitrary curvature. This work was among the first to specifically target curved and wavy text.

### 2.3 U-Net for Segmentation

U-Net [2] was originally proposed for biomedical image segmentation. Its symmetric encoder-decoder structure with skip connections allows it to combine high-level semantic information from deep layers with precise spatial detail from shallow layers. This property makes it particularly suited to segmentation tasks where exact boundary localisation matters — a requirement directly applicable to text region detection. Several text detection works have adopted U-Net-style architectures as their backbone, benefiting from the skip connections that preserve fine-grained boundary information [11].

### 2.4 Instance Segmentation with Mask R-CNN

Mask R-CNN [3] extends Faster R-CNN by adding a parallel branch that predicts a binary segmentation mask for each detected region of interest. The Feature Pyramid Network (FPN) backbone enables detection at multiple scales, which is important for text that appears at varying sizes. When applied to scene text, Mask R-CNN provides instance-level masks, from which tight polygon boundaries can be extracted — making it a natural fit for datasets with polygon-level annotation such as Total-Text.

### 2.5 Scene Text Recognition

Scene text recognition (reading) has evolved from sliding-window classifiers to recurrent sequence models. CRNN [14] combines convolutional feature extraction with bidirectional LSTM and CTC decoding, setting the standard for sequence-based recognition. Attention mechanisms were subsequently introduced to align visual features with character outputs without requiring explicit segmentation.

**TrOCR** [4] replaces both the convolutional feature extractor and the recurrent decoder with transformers. The encoder is a Vision Transformer (ViT) pre-trained on large image corpora, and the decoder is a GPT-2 style autoregressive language model. Fine-tuned on text recognition datasets, TrOCR achieves strong performance on irregular and curved text, making it well-suited to the Total-Text recognition task.

### 2.6 End-to-End Systems

Recent works have pursued end-to-end trainable systems that jointly learn detection and recognition. ABCNet [15] uses Bezier curves to represent text boundaries and couples this with a recognition branch. Mask TextSpotter [16] adds a character-level segmentation branch on top of Mask R-CNN. While these systems offer the benefit of joint optimisation, they require large annotated datasets and substantial computational resources. Our divide-and-conquer strategy, while forgoing joint optimisation, allows each component to be developed and evaluated independently.

## 3 DATASET AND EVALUATION

### 3.1 Total-Text Dataset

The Total-Text dataset [1] provides 1,255 training images and 300 test images sourced from natural scene photographs. Images include street signs, product packaging, menus, and digital displays, with a deliberate focus on scenes containing curved or multi-oriented text. In total, the dataset contains 9,330 annotated text instances across the training set.


**Figure 1:** Total-Text Dataset: Annotation Types

### 3.1.1 Annotation Format

Each text instance is annotated in a plain-text file using the following format:

```
x: [[206 251 386 542 620 646 550 358 189 140]],
y: [[ 633 811 931 946 926 976 1009 989 845 629]],
ornt: [u'c'],
transcriptions: [u'PETROSAINS']
```

The `x` and `y` arrays define the vertices of the polygon boundary in image pixel coordinates. The polygon vertex count is variable (typically 4 to 14 vertices), reflecting the curvature of the text. The `ornt` field takes one of three values: `c` (curved), `h` (horizontal), or `v` (vertical). Instances that are illegible or ambiguous carry the special label `#`, indicating they should be excluded from recognition evaluation but not from detection evaluation.

### 3.1.2 Dataset Statistics

Both training and test splits share a consistent distributional profile, confirming that the benchmark provides a representative and unbiased evaluation setting.

**Orientation distribution.** The training set is strongly dominated by curved instances (c: 58.1%) and horizontal instances (h: 41.9%), with no vertical instances (v: 0.0%). The test set mirrors this pattern closely: curved instances account for 56.3%, horizontal for 43.3%, and vertical for a negligible 0.3%. The near-absence of vertical text in both splits reflects the composition of the source images, which are predominantly outdoor scene photographs rather than, for example, vertically-oriented signage. The dominance of curved instances is precisely what makes Total-Text a challenging benchmark relative to ICDAR datasets where horizontal text predominates.

**Polygon vertex count.** Figure 2 shows the polygon vertex count distributions for both splits. The distribution is strongly right-skewed in both cases: the majority of polygons have 4 to 7 vertices, with a mode at 6. Training polygons have a minimum of 3 vertices, a maximum of 15, and a mean of 5.2; test polygons range from 4 to 18 vertices with a mean of 5.5. The slightly higher maximum and mean vertex count in the test set indicates the presence of a few particularly complex curved instances that require finer polygon approximation. Both means are well within the range of 4–12 vertices that our contour-based polygon extractor typically produces, suggesting that the extraction algorithm is appropriately calibrated for the annotation density of this dataset.

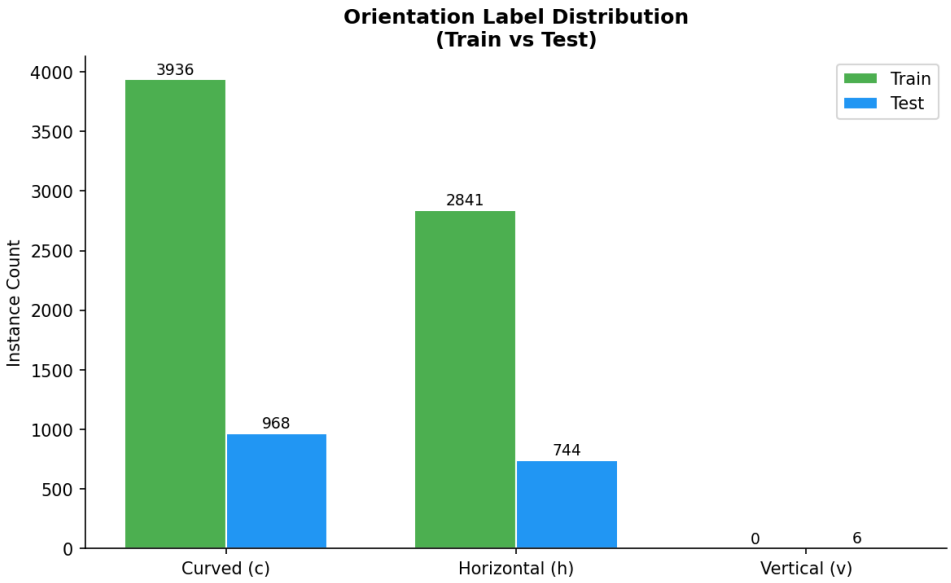


**Figure 2:** Total-Text Dataset: Label Distribution

This variability in polygon complexity directly motivates the adaptive epsilon strategy adopted in Stage 2 (Section **??**): rather than using a fixed vertex budget, the Douglas-Peucker simplification

tolerance is set proportionally to the contour perimeter, allowing the extracted polygons to naturally match the complexity of each instance.

| Split | Images | Instances | Curved (%) | Horizontal (%) | Vertical (%) |
|---|---|---|---|---|---|
| Train | 1,255 | 6777 | ~58.1 | ~41.9 | 0.0 |
| Test | 300 | 1,718 | ~56.3 | ~43.3 | ~0.3 |

**Table 1:** Total-Text dataset statistics.

## 3.2 Evaluation Metrics

### 3.2.1 IoU-Based Metrics

For detection evaluation at the instance level, we compute pairwise Intersection over Union (IoU) between predicted and ground-truth polygon masks:

$$\mathrm{IoU}(P, G) = \frac{\mathrm{Area}(P \cap G)}{\mathrm{Area}(P \cup G)} \tag{1}$$

A greedy matching procedure assigns each prediction to the ground-truth instance with the highest IoU, subject to a matching threshold of 0.5. From the resulting true positives (TP), false positives (FP), and false negatives (FN), standard precision, recall, and F1 scores are computed:

$$\mathrm{Precision} = \frac{TP}{TP + FP}, \quad \mathrm{Recall} = \frac{TP}{TP + FN}, \quad F_1 = \frac{2 \cdot P \cdot R}{P + R} \tag{2}$$

### 3.2.2 DetEval Protocol

The DetEval protocol, derived from the ICDAR evaluation standard, implements a two-threshold matching scheme that is more nuanced than simple IoU matching. For each predicted detection $D_i$ and ground-truth region $G_j$, two overlap ratios are defined:

$$\sigma(D_i, G_j) = \frac{\mathrm{Area}(D_i \cap G_j)}{\mathrm{Area}(G_j)}, \qquad \tau(D_i, G_j) = \frac{\mathrm{Area}(D_i \cap G_j)}{\mathrm{Area}(D_i)} \tag{3}$$

Here $\sigma$ measures the fraction of the ground-truth region covered by the detection (a recall-side criterion), and $\tau$ measures the fraction of the detection that falls within the ground-truth region (a precision-side criterion). A detection–ground-truth pair is accepted as a match if and only if $\sigma \geqslant 0.8$ and $\tau \geqslant 0.4$ simultaneously.

**One-to-one matching.** The standard case: a single detection $D_i$ is matched to a single ground-truth region $G_j$ when both thresholds are satisfied. This detection is counted as a true positive, contributing to both recall and precision.

**One-to-many matching (split detection).** When a single ground-truth word is split across multiple smaller detections — for instance, if the segmentation model breaks a long curved word into two fragments — none of the individual fragments may satisfy $\sigma \geqslant 0.8$ on their own, even though collectively they cover the ground truth well. In the original DetEval specification [18], a one-to-many configuration can still yield a partial match credit. In our implementation, we follow a strict one-to-one greedy assignment: matches are sorted by $\sigma \times \tau$ and assigned greedily without splitting, so fragmented detections are penalised as false positives and the corresponding ground truth as a false negative.

**Many-to-one matching (merged detection).** Conversely, when adjacent text instances merge into a single large predicted polygon — the dominant failure mode of our U-Net pipeline — one detection $D_i$ may have high $\tau$ (it lies mostly within one of the ground truths) but low $\sigma$ for each individual ground truth (it covers none of them by 80% or more). Such merged detections are counted as false positives for every ground-truth instance they partially overlap, directly reducing precision.

**Handling of illegible instances (#).** Ground-truth instances annotated with the orientation label # are illegible or ambiguous and are excluded from the recognition evaluation. For detection evaluation,

however, they represent real physical text regions and should not penalise a system for detecting them. We therefore apply a *detection filtering* step before evaluation: any predicted polygon whose intersection-over-detection ratio with a #-labelled ground-truth region exceeds 0.5 is silently removed from the prediction list rather than being counted as a false positive. This follows the convention of the official Total-Text evaluation script and ensures that predictions which correctly identify illegible text are not penalised.

The primary reported metric is the harmonic mean (H-mean) of DetEval precision and recall:

$$\text{H-mean} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

#### 3.2.3 *Frame-Level Segmentation IoU*

During training, we monitor the mean IoU between predicted binary masks and ground-truth binary masks as a proxy for segmentation quality before polygon extraction.

## 4 PREPROCESSING AND DATA PREPARATION (DETECTION PIPELINE)

### 4.1 Ground-Truth Mask Generation

The U-Net model is trained on binary masks derived from the polygon annotations. For each training image, all annotated polygons (including those labelled #) are filled onto a blank canvas to produce a binary text-region mask. Pixels inside any polygon are labelled 1 (text); all remaining pixels are labelled 0 (background). This formulation frames Stage 1 as a binary semantic segmentation problem.


**Figure 3:** Total-Text Dataset: Annotation Types

### 4.2 Image Resizing

All images are resized to $512 \times 512$ pixels before being fed to the U-Net. This uniform size is necessary because fully convolutional networks are resolution-agnostic in principle, but efficient batching during training requires a fixed spatial resolution. The predicted masks are subsequently rescaled back to the original image dimensions before polygon extraction, preserving coordinate fidelity.

### 4.3 Data Augmentation

To improve generalisation on the relatively small Total-Text training set (1,255 images), we apply the following augmentations using the Albumentations library:


**Figure 4:** Total-Text Dataset: Data Augmentation

- **Horizontal flip** (probability 0.5): Mirrors the image and its corresponding mask. Valid because text orientation is generally preserved under horizontal reflection.
- **Random rotation** ($\pm 15$): Introduces slight angular variation without producing unrealistic configurations.
- **Colour jitter** (brightness, contrast, saturation, hue): Reduces sensitivity to illumination conditions. Applied only to the image, not the mask.
- **Normalisation**: Images are normalised using ImageNet mean and standard deviation values to match the statistical distribution expected by pre-trained convolutional feature extractors.
- **Random scaling** (scale factor in $[0.75, 1.25]$): The image and its corresponding mask are jointly rescaled by a randomly sampled factor before being centre-cropped or padded back to $512 \times 512$. This simulates text appearing at different distances from the camera and prevents the model from relying on absolute character size as a cue for text detection.

## 5 Stage 1: Text Region Segmentation

### 5.1 U-Net Architecture

The segmentation backbone is a U-Net [2] trained from scratch with a base channel width of 32. The architecture follows the original symmetric encoder-decoder design with four resolution levels and skip connections at each level.

#### *5.1.1 Encoder*

The encoder consists of four convolutional blocks (ConvBlocks), each containing two consecutive $3 \times 3$ convolutions followed by Batch Normalisation and ReLU activation. Max pooling ($2 \times 2$, stride 2) is

applied between blocks to progressively halve the spatial resolution. The channel widths double at each level: $32 \to 64 \to 128 \to 256$.

#### 5.1.2 Bottleneck

A fifth ConvBlock at the lowest resolution processes features at $32 \times 32$ (for $512 \times 512$ input) with 512 channels. This bottleneck forces the network to compress spatial information into high-level semantic representations before reconstruction.

#### 5.1.3 Decoder

The decoder mirrors the encoder. At each level, a $2 \times 2$ transposed convolution upsamples the feature map, which is then concatenated with the corresponding encoder skip connection and processed by a ConvBlock. This design progressively recovers spatial resolution while combining semantic context with fine-grained local detail:

$$\mathbf{d}_k = \text{ConvBlock}\left([\text{Up}(\mathbf{d}_{k+1}) \,\|\, \mathbf{e}_k]\right) \tag{5}$$

where $\mathbf{e}_k$ is the encoder output at level $k$, $\text{Up}(\cdot)$ is bilinear upsampling followed by convolution, and $\|$ denotes channel concatenation.

#### 5.1.4 Output Head

A final $1 \times 1$ convolution maps the decoder output to a single-channel logit map, followed by sigmoid activation to produce a per-pixel probability of text presence.

| Stage | Block | Output Channels | Spatial Size |
|---|---|---|---|
| Encoder | ConvBlock 1 | 32 | $512 \times 512$ |
| | ConvBlock 2 | 64 | $256 \times 256$ |
| | ConvBlock 3 | 128 | $128 \times 128$ |
| | ConvBlock 4 | 256 | $64 \times 64$ |
| Bottleneck | ConvBlock 5 | 512 | $32 \times 32$ |
| Decoder | DecBlock 4 | 256 | $64 \times 64$ |
| | DecBlock 3 | 128 | $128 \times 128$ |
| | DecBlock 2 | 64 | $256 \times 256$ |
| | DecBlock 1 | 32 | $512 \times 512$ |
| Output | Conv $1 \times 1$ + Sigmoid | 1 | $512 \times 512$ |

**Table 2:** U-Net architecture summary for Stage 1 segmentation.

### 5.2 Training Details

#### 5.2.1 Loss Function

We train with binary cross-entropy (BCE) loss computed over all pixels. The ground-truth labels are soft (0 or 1) and BCE naturally handles the per-pixel binary classification:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{HW} \sum_{i,j} \left[ y_{ij} \log \hat{y}_{ij} + (1 - y_{ij}) \log(1 - \hat{y}_{ij}) \right] \tag{6}$$

where $y_{ij} \in \{0, 1\}$ is the ground-truth label and $\hat{y}_{ij} \in (0, 1)$ is the predicted probability for pixel $(i, j)$.

#### 5.2.2 Optimiser and Scheduler

We use AdamW [19] with learning rate $10^{-3}$ and weight decay $10^{-4}$. A cosine annealing scheduler reduces the learning rate to near zero over the training run, providing smooth convergence without

abrupt steps. An early-stopping callback monitors validation IoU with patience 7 epochs in maximisation mode, terminating training when no improvement is observed.

#### 5.2.3 *Training Results*

The model was trained for up to 80 epochs. Training stopped early at epoch 60 due to the early-stopping criterion.

| Metric | Train (final) | Val (best) | Val epoch |
|---|---|---|---|
| Loss (BCE) | 0.1109 | 0.1855 | 55 |
| IoU | 0.7669 | 0.6241 | 53 |
| IoU gap | | 0.1169 | |

**Table 3:** Stage 1 segmentation training results

Training BCE loss decreases steadily from 0.5392 at epoch 1 to 0.1109 at epoch 60. The best validation loss of 0.1855 occurs at epoch 55, slightly ahead of the best validation IoU epoch (53), reflecting the typical lag between loss minimisation and metric optimisation. The train/val IoU gap of 0.1169 indicates moderate overfitting consistent with the limited training set size.

## 6 STAGE 2: POLYGON EXTRACTION FROM PREDICTED MASKS

### 6.1 Contour Detection

After thresholding the sigmoid output at 0.5 to obtain a binary mask, we extract polygon boundaries using OpenCV's `findContours` function with the `RETR_EXTERNAL` flag (retrieves only outermost contours) and `CHAIN_APPROX_SIMPLE` encoding (suppresses redundant collinear points). Each connected component in the binary mask yields one contour.

### 6.2 Polygon Approximation

Raw contours from pixel-level masks contain a large number of vertices (one per boundary pixel). To obtain compact polygons suitable for downstream processing and DetEval evaluation, we apply the Douglas-Peucker algorithm [20] via `cv2.approxPolyDP`:

$$\epsilon = 0.01 \times \text{arcLength}(\text{contour}) \tag{7}$$

This adaptive $\epsilon$ retains approximately 1% of the contour perimeter as the maximum permissible deviation between the simplified polygon and the original contour. For typical text regions, this produces polygons with 4 to 12 vertices — comparable in density to the Total-Text ground-truth annotations.

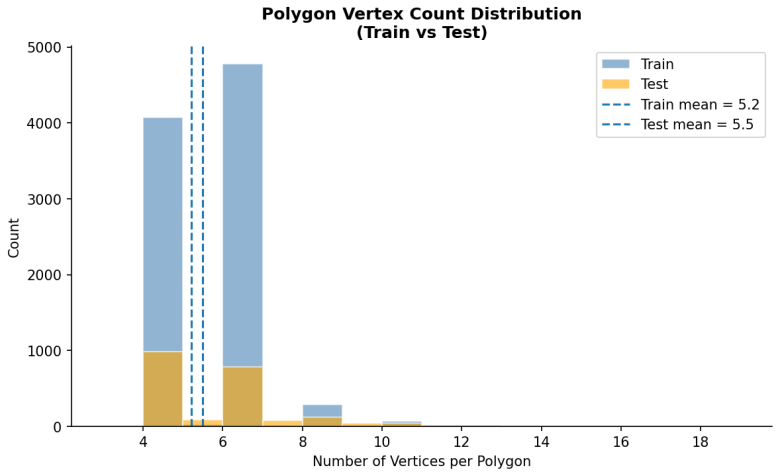


**Figure 5:** Total-Text Dataset: Polygon Vertex Count Distribution

Contours with an area smaller than 100 pixels and polygons with fewer than 3 vertices are discarded as noise. All polygon coordinates are rescaled from the $512 \times 512$ processing resolution to the original image dimensions using multiplicative scale factors derived from the image's true height and width.

### 6.3 Limitations of Contour-Based Polygon Extraction

The contour-based approach is a classical, non-learned method. While computationally efficient, it has two notable limitations in the context of curved text. First, it can only produce straight-line segments between vertices; it cannot represent smooth curves without a large number of vertices. Second, when two text instances are spatially adjacent or overlapping, their masks may merge into a single connected component, causing the extractor to produce a single polygon encompassing multiple words instead of separate polygons per instance.

These limitations are partially mitigated by the quality of the upstream segmentation mask, but they fundamentally limit the achievable recall for tightly-spaced text. A learned polygon regression head (e.g., a transformer decoder that directly predicts vertex sequences [21]) would address both limitations but was outside the scope of this work.

## 7 Stage 3: Orientation Classification

### 7.1 Problem Formulation

Having extracted polygon regions from the binary mask, we classify the text orientation of each region into one of three classes: **c** (curved), **h** (horizontal), and **v** (vertical). This mirrors the `ornt` annotation field in the Total-Text ground truth. Instances annotated with # (illegible) are excluded from the classifier's training and evaluation data.

### 7.2 Dataset Construction

For each training image, we iterate over the annotated instances and extract bounding-box crops from the original image guided by the polygon vertices. Specifically, for a polygon with x-coordinates $\{x_i\}$ and y-coordinates $\{y_i\}$, the bounding box is:

$$[x_1, x_2] = [\max(0, \min_i x_i),\ \min(W, \max_i x_i)], \quad [y_1, y_2] = [\max(0, \min_i y_i),\ \min(H, \max_i y_i)] \tag{8}$$

The crop $\mathbf{I}[y_1 : y_2, x_1 : x_2]$ is taken from the original image rather than the predicted mask, preserving texture information that is informative for orientation. The orientation label assigned by the

ground-truth annotation provides the classification target. All crops are resized to $64 \times 64$ pixels for uniform input to the classifier.

### 7.3 Classifier Architecture

We design a lightweight convolutional neural network for the three-class classification task. The network takes a $64 \times 64 \times 3$ RGB crop as input and produces a three-dimensional softmax output vector. The architecture consists of three convolutional blocks (Conv2D $\rightarrow$ BatchNorm $\rightarrow$ ReLU $\rightarrow$ MaxPool) followed by global average pooling and two fully connected layers. Cross-entropy loss is used for training.

### 7.4 Training Details

The orientation classifier is trained using AdamW with learning rate $10^{-3}$ and cosine annealing. An early-stopping callback monitors validation accuracy with patience of 7 epochs, terminating training at epoch 24 out of a maximum of 40.

| Metric | Train (final) | Val (best) | Val epoch |
|---|---|---|---|
| Accuracy | 0.9579 | 0.9313 | 24 |

**Table 4:** Stage 3 orientation classifier training results.

The classifier achieves 93.13% validation accuracy, indicating that orientation is a learnable and relatively tractable property from bounding-box image crops, even with a compact architecture. The train/val accuracy gap of 2.66 percentage points suggests good generalisation with minimal overfitting.

### 7.5 Full Inference Pipeline

At inference time, the three stages are chained as follows:

1. The U-Net segmentation model processes the input image and produces a binary text-region mask (Stage 1).
2. Connected contours are extracted and approximated into variable-length polygons (Stage 2).
3. Each polygon's bounding-box crop is classified into one of three orientations by the convolutional classifier (Stage 3).
4. The result for each image is written to a plain-text file in DetEval format: one polygon per line, encoded as alternating $y, x$ coordinate pairs.

Because transcription is beyond the scope of this pipeline, we adapt the DetEval evaluation protocol to assess detection quality alone, comparing predicted polygons against ground-truth polygons. The evaluation script was modified to read ground-truth annotations in the Total-Text plain-text format (described in Section 3.1) rather than the default `.mat` binary format, and to read predictions from plain-text files rather than structured matrices. Polygon-level intersection functions (`iod`, `area_of_intersection`, `area`) from the official Total-Text evaluation repository are retained unchanged.

## 8 TEXT RECOGNITION PIPELINE: MASK R-CNN AND TROCR

While this U-Net based pipeline represents an example of from-scratch design for text detection, it does not address the transcription task. This section proposes an additional text recognition pipeline

based on two fine-tuned pre-trained models. Both text detection and recognition pipelines share the same overall objective, which is to perform text spotting on Total-Text, but differ from each other in design philosophy, with one focusing on from-scratch design with interpretability, and the other focusing on pre-trained models for end-to-end text spotting, including transcription

## 8.1 Stage 1: Instance Segmentation with Mask R-CNN

### 8.1.1 Architecture

Mask R-CNN [3] extends the Faster R-CNN object detector by adding a third branch, parallel to the bounding-box regression and classification heads, that predicts a binary segmentation mask for each proposed region of interest (RoI). The backbone is a ResNet-50 network combined with a Feature Pyramid Network (FPN) [17], which produces multi-scale feature maps at four resolution levels ($P_2$ to $P_5$). The FPN allows the detector to handle text instances at a wide range of scales within a single forward pass.

The model is configured for single-class detection (text vs. background) with the mask prediction branch enabled:

- **Backbone:** ResNet-50 + FPN, initialised from COCO pre-trained weights.
- **RoI heads:** Single class (`NUM_CLASSES = 1`), score threshold 0.4 at inference.
- **Mask branch:** FCN-based mask head producing $28 \times 28$ binary instance masks per proposal.

Two variants of the model were trained on Total-Text: a *pixel-level* model fine-tuned on pixel-accurate segmentation annotations, and a *region-level* model fine-tuned on text region bounding-box annotations. At inference, the region-level model is used preferentially.

### 8.1.2 Polygon Extraction

After Mask R-CNN produces a set of instance masks and their associated confidence scores, we extract polygon boundaries from each mask using the same contour-based procedure described in Section 6 (`findContours` + Douglas-Peucker approximation). For each detected instance, the tightest polygon is retained.

## 8.2 Stage 2: Text Recognition with TrOCR

### 8.2.1 Architecture

TrOCR [4] is a transformer-based encoder-decoder model for optical character recognition. The encoder is a Vision Transformer (ViT) pre-trained on large-scale image data that maps an input image patch sequence to a sequence of contextualised feature vectors. The decoder is a GPT-2 style autoregressive language model that generates the output character sequence token by token, attending to the encoder representations via cross-attention:

$$P(y_1, \ldots, y_T \mid \mathbf{x}) = \prod_{t=1}^{T} P(y_t \mid y_1, \ldots, y_{t-1}, \text{Encoder}(\mathbf{x})) \tag{9}$$

The model was fine-tuned on Total-Text word-image and transcription pairs. At inference, each predicted text instance polygon is cropped to a tight bounding box, resized to the TrOCR input resolution, and passed through the recognition model to produce a predicted word string.

### 8.2.2 Inference Procedure

The full inference procedure for a single image is:

1. Run Mask R-CNN on the input image to obtain a set of $(\text{mask}_i, \text{score}_i)$ pairs.

2. Extract the polygon boundary from each instance mask.
3. For each polygon, compute the axis-aligned bounding box and crop the corresponding region from the original image.
4. Pass the crop through TrOCR to obtain a transcription string.
5. Return the set of (polygon, confidence, text) tuples.

### 8.3 Evaluation Adaptation

The evaluation notebook computes both the IoU-based metrics (greedy matching at IoU threshold 0.5) and the DetEval-style metrics (recall threshold 0.8, precision threshold 0.4) directly from the predicted and ground-truth instance masks. Ground-truth masks are obtained from the provided binary text-region annotation images, and connected component labelling is used to separate individual instances.

## 9 RESULTS AND DISCUSSION

### 9.1 Detection Pipeline (U-Net)

#### *9.1.1 Segmentation Quality*

Table 3 summarises the binary mask segmentation performance during training. The best validation IoU of 0.6241 at epoch 53 indicates that the U-Net successfully learns to identify text regions, albeit with a non-trivial train/val gap attributable to the limited training set size and the challenge of highly curved or densely packed text.

#### *9.1.2 DetEval Detection Performance*

After full pipeline inference (mask prediction to polygon extraction to orientation classification), we evaluate the resulting predictions against the Total-Text test ground truth using the modified DetEval protocol:

| System | Precision | Recall | F-Score |
|---|---|---|---|
| U-Net + contour polygons (DetEval) | 0.5129 | 0.5839 | 0.5461 |

**Table 5:** Detection pipeline performance on Total-Text test set under the DetEval protocol.

The recall of 0.5839 exceeds precision (0.5129), suggesting that the pipeline tends to detect genuine text regions but also produces spurious predictions. The most common cause is merged contours: when two adjacent text instances are insufficiently separated in the binary mask, they form a single connected component that generates one large polygon counted as a false positive against each individual ground-truth instance.

#### *9.1.3 Orientation Classification*

The orientation classifier achieves 93.13% validation accuracy, indicating strong discriminability between the three orientation classes from bounding-box crops. Errors concentrate on ambiguous cases where a mildly curved word could reasonably be labelled either `c` or `h`.


**Figure 6:** Total-Text Dataset: Sample visualization

## 9.2 Recognition Pipeline (Mask R-CNN + TrOCR)

| Protocol | System | Precision | Recall | F1 / H-mean |
|---|---|---|---|---|
| IoU ($\geqslant 0.5$) | Mask R-CNN + TrOCR | 0.2980 | 0.8129 | 0.4361 |
| DetEval (ICDAR) | Mask R-CNN + TrOCR | 0.2258 | 0.6159 | 0.3304 |

**Table 6:** Recognition pipeline performance on Total-Text test set.

The recognition pipeline exhibits a pronounced recall-precision asymmetry: recall is high (0.8129 under IoU), meaning the model successfully detects most ground-truth text instances, but precision is low (0.2980), indicating a large number of false-positive detections. This pattern is characteristic of detection models operating at a low score threshold: at threshold 0.4, Mask R-CNN accepts many low-confidence proposals that correspond to non-text or duplicate regions.

The DetEval H-mean of 0.3304 is lower than the IoU-based F1 (0.4361) because the DetEval recall threshold of 0.8 is more demanding than the IoU threshold of 0.5 — requiring that a detection cover at least 80% of the ground-truth polygon area.


**Figure 7:** Total-Text Dataset: Sample visualization with transcription

## 9.3 Comparative Analysis

| Pipeline | Protocol | Precision | Recall | F / H-mean | Transcription |
|---|---|---|---|---|---|
| U-Net (ours) | DetEval | 0.5129 | 0.5839 | 0.5461 | No |
| Mask R-CNN + TrOCR | IoU | 0.2980 | 0.8129 | 0.4361 | Yes |
| Mask R-CNN + TrOCR | DetEval | 0.2258 | 0.6159 | 0.3304 | Yes |

**Table 7:** Summary comparison of both pipelines on the Total-Text test set.

The U-Net pipeline achieves a higher DetEval F-score (0.5461) than the recognition pipeline (0.3304). However, this comparison is not entirely fair: the U-Net pipeline is evaluated on detection quality alone (no transcription required), while the recognition pipeline is evaluated on a stricter end-to-end criterion that implicitly penalises any mismatch between predicted and actual instance boundaries.

More importantly, the two pipelines are designed for different sub-tasks. The U-Net pipeline is optimised for producing clean, compact polygon boundaries, while the Mask R-CNN pipeline is optimised for high-recall instance discovery to minimise missed words before handing off to TrOCR. The high recall of the Mask R-CNN pipeline (0.8129) makes it well-suited as the detection front-end of an end-to-end text spotting system, at the cost of lower precision that would need to be improved through score threshold tuning or non-maximum suppression.

## 9.4 Error Analysis

**U-Net pipeline errors:** The dominant failure mode is merged detections caused by spatially adjacent text instances sharing a connected region in the binary mask. This systematically reduces precision

because one predicted polygon corresponds to multiple ground-truth instances. A secondary cause of errors is highly curved text: the sigmoid mask is spatially smooth and may produce thin connections between curved words and background regions that survive thresholding, yielding irregular contours.

**Recognition pipeline errors:** The primary source of false positives is the low confidence threshold (0.4) that was chosen to maximise recall. Many false positives correspond to low-confidence detections of background texture, partial characters, or repeated detections of the same word region at slightly different scales due to FPN multi-scale processing. Raising the threshold would improve precision substantially but at the cost of recall.

# 10 BROADER DISCUSSION

## 10.1 The Divide-and-Conquer Trade-off

Our two contributions illustrate a fundamental trade-off in complex vision pipelines: building from scratch offers transparency and control at the cost of limited capacity, while leveraging pre-trained models offers strong performance at the cost of reduced interpretability and dependence on external resources.

The U-Net pipeline makes every component explicit and observable: the binary mask quality can be inspected directly, the polygon approximation parameters can be tuned, and the orientation classifier can be evaluated independently. This modular transparency is valuable for debugging and for understanding which component is the bottleneck. The Mask R-CNN + TrOCR pipeline, by contrast, provides higher recall and adds transcription capability, but the behaviour of the pre-trained backbone is largely opaque.

## 10.2 Challenges Specific to Curved Text

Both pipelines struggle with curved text, albeit for different reasons. The U-Net pipeline's contour-based polygon extractor cannot represent smooth curves, producing polygonal approximations with sharp corners that poorly match the ground-truth polygon boundaries. The Mask R-CNN pipeline inherits better instance-level localisation from its pre-trained backbone but is still limited by the resolution of its $28 \times 28$ mask output, which must be upsampled to full resolution — introducing quantisation artefacts for fine-grained curved boundaries.

Dedicated curved text methods such as TextSnake [13] and ABCNet [15] address this by parameterising text boundaries as Bezier curves or center-line disks rather than polygons. These representations could be explored as extensions to both pipelines.

## 10.3 Data Scale and Pre-training

The contrast between the two pipelines also reflects the importance of pre-training at scale. The U-Net is trained on 1,255 images with no pre-trained backbone, while Mask R-CNN begins from weights pre-trained on the 118,000-image COCO dataset. The transfer of convolutional feature representations from COCO to Total-Text is largely beneficial because natural image features (edges, textures, shapes) are shared across domains.

Future work could improve the U-Net pipeline by replacing the from-scratch encoder with a pre-trained ResNet-50 backbone, as noted in the source code, while retaining the custom decoder. This would likely improve validation IoU substantially at the cost of some transparency regarding what was learned.

## 10.4 Evaluation Protocol Considerations

The DetEval protocol's two-threshold design penalises over-detection (large polygons covering background) through the precision-side threshold ($\tau \geqslant 0.4$) while rewarding coverage of ground-truth regions through the recall-side threshold ($\sigma \geqslant 0.8$). This design choice aligns with the use case of

downstream OCR: a detection that misses 30% of a word's pixels will likely produce a corrupted transcription, so partial detections are more harmful than they appear under simple IoU.

The modification of the evaluation script to accept plain-text ground-truth files (rather than the original MATLAB format) was essential for running evaluation without MATLAB and represents a practical contribution to the reproducibility of results on this benchmark.

## 11 CONCLUSION

This report has presented a two-pipeline approach to scene text detection and recognition on the Total-Text benchmark. The first pipeline, built from scratch, trains a U-Net segmentation model to produce binary text-region masks, extracts variable-length polygon boundaries using contour approximation, and classifies text orientation with a lightweight convolutional network. Evaluated under the DetEval protocol, this pipeline achieves an F-score of 0.5461, demonstrating that a from-scratch approach with carefully designed components can produce competitive detection results on a challenging curved-text benchmark.

The second pipeline augments detection with transcription by fine-tuning Mask R-CNN for instance segmentation and TrOCR for word recognition. This pipeline achieves a high recall of 0.8129 under IoU matching, confirming the effectiveness of pre-trained models for text localisation, while also exposing the precision challenges associated with low confidence thresholds and duplicate detections.

Jointly, the two pipelines represent complementary views of the same problem: one prioritising interpretable, from-scratch learning, and the other leveraging the capacity of large pre-trained models. The orientation classifier's strong validation accuracy (93.13%) confirms that orientation is a tractable auxiliary task that enriches the detection output without requiring significant additional resources.

Future work should explore: (1) replacing the U-Net's from-scratch encoder with a pre-trained backbone to close the train/val IoU gap; (2) using a learned polygon head or Bezier curve parameterisation to better capture curved boundaries; (3) tuning Mask R-CNN's confidence threshold to improve precision in the recognition pipeline; and (4) integrating the detection and recognition stages into a single jointly trained end-to-end model as annotated data scale permits.

## ACKNOWLEDGEMENTS

We thank Prof. Nguyen Duc Dung for guidance throughout this project. We acknowledge the creators of the Total-Text dataset and the maintainers of the Detectron2 and Hugging Face Transformers libraries.

## REFERENCES

[1] C.-K. Chng, C.S. Chan, and C.-L. Liu, "Total-Text: Towards Orientation Robustness in Scene Text Detection," *International Journal on Document Analysis and Recognition*, vol. 23, pp. 31–52, 2019.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Proc. MICCAI*, pp. 234–241, 2015.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *Proc. ICCV*, pp. 2961–2969, 2017.

[4] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, "TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models," *Proc. AAAI*, pp. 13094–13102, 2023.

[5] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004.

[6] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting Text in Natural Scenes with Stroke Width Transform," *Proc. CVPR*, pp. 2963–2970, 2010.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *Proc. CVPR*, pp. 3431–3440, 2015.

[8] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "TextBoxes: A Fast Text Detector with a Single Deep Neural Network," *Proc. AAAI*, pp. 4161–4167, 2017.

[9] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: An Efficient and Accurate Scene Text Detector," *Proc. CVPR*, pp. 5551–5560, 2017.

[10] D. Deng, H. Liu, X. Li, and D. Cai, "PixelLink: Detecting Scene Text via Instance Segmentation," *Proc. AAAI*, pp. 6773–6780, 2018.

[11] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao, "Shape Robust Text Detection with Progressive Scale Expansion Network," *Proc. CVPR*, pp. 9336–9345, 2019.

[12] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-Time Scene Text Detection with Differentiable Binarization," *Proc. AAAI*, pp. 11474–11481, 2020.

[13] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao, "TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes," *Proc. ECCV*, pp. 20–36, 2018.

[14] B. Shi, X. Bai, and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, 2017.

[15] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, "ABCNet: Real-Time Scene Text Spotting with Adaptive Bezier-Curve Network," *Proc. CVPR*, pp. 9809–9818, 2020.

[16] M. Lyu, J. Yang, H. Wu, Q. Yu, L. Li, J. Weng, and R. Liao, "Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes," *Proc. ECCV*, pp. 67–83, 2018.

[17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," *Proc. CVPR*, pp. 2117–2125, 2017.

[18] C. Wolf and J.-M. Jolion, "Object Count / Area Graphs for the Evaluation of Object Detection and Segmentation Algorithms," *International Journal on Document Analysis and Recognition*, vol. 8, pp. 280–296, 2006.

[19] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *Proc. ICLR*, 2019.

[20] D. H. Douglas and T. K. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature," *The Canadian Cartographer*, vol. 10, no. 2, pp. 112–122, 1973.

[21] M. Ye, J. Zhang, S. Zhao, J. Liu, T. Du, E. Chang, and Z. Luo, "DPText-DETR: Towards Better Scene Text Detection with Dynamic Points in Transformer," *Proc. AAAI*, pp. 3241–3249, 2023.