

Bilingual Text Detection and Recognition from Video Frames

Venkata B Hangarage¹, Gururaj Mukarambi², and Chetan³

¹ Amity School of Engineering and Technology,
Amity University, Hyderabad, 500108, Telangana, India
vbj44@yahoo.com

² Department of Computer Science, School of Computer Science
Central University of Karnataka Kalburagi, Karnataka, India
(Corresponding author) : gmukarambi@gmail.com

³ Department of Computer Science, School of Computer Science
Central University of Karnataka Kalburagi, Karnataka, India
mulge10chetan@gmail.com

Abstract. Indian video content has many text scenes with multiple languages in the same frame; it is a state where OCR fails to process. This work focuses on a structured end-to-end pipeline that tackles this problem by localizing text and performing bilingual character recognition on Kannada and English video frames. YOLOv8 identifies the text present in each frame and then passes that text region to EasyOCR after an image enhancement step, which is applied to each and every type of text region found. A 5,000-frame dataset collected from news videos, social media videos, and literature videos with only Kannada text, English text, and mixed texts was prepared and processed. Each string is labeled using Unicode point analysis, and a deduplication step removes duplicate texts without any major involvement of GPU hardware at inference time. The pipeline attains an F1-score of 0.82 on the considered validation set, which proves that working on bilingual text extraction is achievable on simple hardware.

Keywords: Bilingual OCR, YOLOv8, EasyOCR, Text Detection, Video Frames, Kannada Script, Computer Vision

1 Introduction

News videos, YouTube videos, and many other social media platforms generate a huge amount of video every day. The text present in these video frames is very useful for many tasks such as transcription, subtitles, accessibility, and media indexing. In Karnataka, this becomes more challenging because Kannada and English often appear together in the same frame, such as in signboards, news tickers, and social media overlays. So, any system working on this type of video must handle both languages properly without affecting each other's performance.

There is a clear difference between the two scripts. Kannada characters are more curved and complex, with shapes that extend above and below the line.

English characters are simpler, thinner, and vary depending on fonts and sizes. Because of this, a setting that works well for Kannada may blur English text, and a setting that works for English may not properly detect Kannada text.

Thw older generations OCR models worked only on the neat dataset not on the videos there were many image issues. To handle this problem, modern systems divide the task into two parts: first detecting where the text is present in the image, and then recognizing and extracting the identified text.

This work follows that approach. YOLOv8 is used to detect text regions, and EasyOCR is used to recognize the text in both Kannada and English. Between these two steps, an image enhancement step is applied based on the script type to improve the results. A Unicode-based method is used to identify the language of the text, and duplicate results from overlapping detections are removed.

Another useful aspect of this work is that the system runs on a normal CPU and does not require high-end hardware. Training is done using a GPU on Google Colab, but the developed model runs on a CPU on a laptop.

Organisation: Section ?? reviews prior work. Section 2 explains the dataset. Section 3 describes the methodology. Section 4 presents the results. Section 5 concludes the paper.

1.1 Background Study

Dhakal S. et al. [1] proposed a YOLOv8-based OCR system for document text extraction. In this method, YOLOv8 was used for text detection and OCR was used for recognition. The model achieved 99.1% mAP for front-side detection and 96.1% for back-side detection, showing very high detection accuracy. Hegde

N. et al. [2] developed a real-time system using YOLOv8 and EasyOCR for traffic monitoring. The model achieved 91.47% precision, 88.6% recall, and 84.3% mAP, showing strong performance in real-world conditions. Sarhan A. et al. [3]

used YOLOv8 along with OCR to build a license plate recognition system. They reported high accuracy in both detection and recognition under different lighting and environmental conditions. Their system also showed strong real-time performance. Salsabila N. et al. [4] combined YOLOv8, EasyOCR, and CNN methods

for license plate recognition. They achieved 100% detection accuracy and 74.66% text recognition accuracy, which shows that combining detection and OCR works effectively. Satya B. et al. [5] developed an optimized YOLOv8-based system for

low-resource devices. They reported improved detection accuracy along with efficient real-time performance, even under challenging conditions like low light and motion blur. Moussaoui H. et al. [6] used YOLOv8 with OCR for vehicle

identification and achieved over 90% detection accuracy, showing reliable performance in real-world conditions. Hegde N. et al. [7] built an AI-based system

using YOLOv8 and OCR for traffic monitoring and license plate recognition.

They reported 91.47% precision, 88.6 recall, and 84.3% mAP, which indicates strong performance in practical scenarios. Hasin N. et al. [8] proposed a multilingual license plate recognition system using YOLOv8 and deep OCR models. They achieved 97.83% overall accuracy, showing that the system can handle complex scripts and real-world conditions effectively.

2 Data Collection and Analysis

2.1 Motivation and Sources

There is no publicly available dataset as per our needs, so we built our own dataset using videos from different sources like news, literature, and many more, converting the video into frames and then working on those divided frames as per our needs.

Each video frame is of a different approach, like different lighting conditions, noise, depending on the type of video source we used.

To cover all types of data, we divided the dataset into three categories.

1. **Kannada-only**: regional news, local signage
2. **English-only**: road boards, headlines, digital captions
3. **Mixed**: frames in which both scripts co-occur

Covering all three categories ensures that evaluation reflects genuine operational diversity rather than a narrow or idealised subset of conditions.

2.2 Frame Extraction

A simple frame extraction was built using the OPENCV and TKINTER which takes frame from the video in the fixed time frame i.e., three frames per second expliciting the video frame rate which gives the balance between the frames and enough data not using much storage

Each divided frame is divided into 640 x 640 size to match the needs of the YOLOV8 model and at last all the frames are saved in JPEG format.

2.3 Dataset Distribution

The reasonably balanced split prevents any single script from dominating the gradient signal during training and encourages the model to generalise across all three content types.

2.4 Data Annotation

All the images were labelled using the Roboflow platform, for each image a bounding box was drawn around all the possible visible text regions that included different camera angles, font sizes, both Kannada and English text, and individual language frames also. Roboflow was used to manage the dataset conversion, convert annotations into YOLO format, and split the dataset into 80

Table 1. Dataset Composition by Language Category

Category	Frames	Share (%)
Kannada-only	2,000	40
English-only	1,800	36
Mixed	1,200	24
Total	5,000	100

2.5 Data Validation

We manually checked some of the labeled images again to fix mistakes, we removed frames that were blurry, had very low contrast, or where the text was not clear by keeping only good quality images. After cleaning, we carefully verified the final dataset before starting the training.

2.6 Dataset Observations

We detect Kannada text more easily because the characters are thick and curved, which makes them more visible to the model. We find English text more difficult to detect because it has thin strokes and many different font styles. We observe that mixed-language frames are the hardest, since both scripts appear close together, and we need to detect them correctly without confusion.

3 Proposed Methodology

At the core, we design the pipeline to perform three main tasks: we locate the text in the image, we extract those regions, and we read the text from them. After that remove duplicate text to clean the final output. Fig. 1 shows the complete flow of the system.

Input frame. A image is given as the input for system which contains texts such as kannada and english and images which also have both languages in it with different background.

Text detection. The given image is tested by the YOLO model to detect the text region on it which is next passed to the EasyOCR to extract the text regions.Boxes with low confidence are removed. $\mathcal{B} = \{b_1, b_2, \dots, b_n\}$.

Crop extraction. The detected bounding box is croppef from the original image which helps in terminating unnecessary background of the image. $R_i = \text{Crop}(I, b_i)$

Script-aware preprocessing. Every divided frame is sized into 640 x 640 pixels and then the image enhancememnt technues Each cropped image is resized to 640×640 pixels. Then, different image enhancement techniques are applied based on the type of text:

The crop region with the kannada text on it is enhanced using the CLAHE is sharpened to highlight the thick characters.

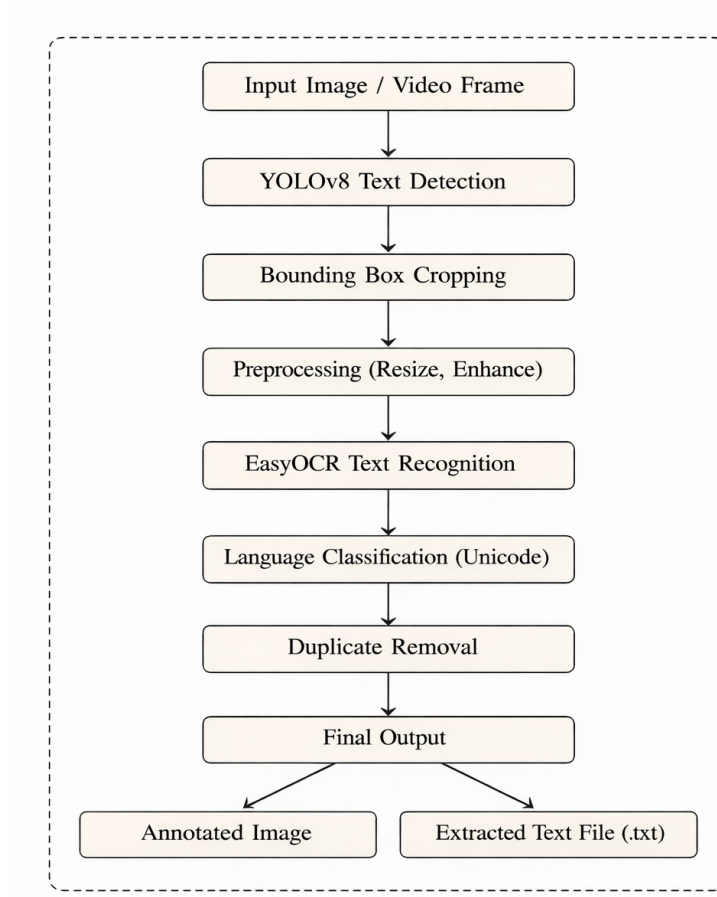


Fig. 1. End-to-end pipeline for bilingual text detection and recognition.

The crop region with the English text is processed using binarization and smoothing to improve.

If the recognition result is not good, the system runs OCR again with slightly adjusted settings.

Recognition. Each processed image is given to EasyOCR. It is configured to recognize both Kannada and English text. The output is the recognized text for each cropped region.

Language classification.

Each recognized text is checked using Unicode values:

- If most characters are in the Kannada range → Kannada
- If most characters are in the ASCII range → English
- Otherwise → Mixed

This method is simple and does not require a separate model.

$$\begin{cases} \text{Kannada} & \text{if majority chars} \in [\text{U+0C80}, \text{U+0CFF}] \\ \text{English} & \text{if majority chars} \in [\text{U+0000}, \text{U+007F}] \\ \text{Mixed} & \text{otherwise} \end{cases} \quad (1)$$

No additional model is needed; the Unicode ranges perform the classification in microseconds.

Deduplication and output. Duplicate text caused by overlapping boxes is removed. The final output includes: An image with bounding boxes and language labels A text file containing all detected text with language information

3.1 YOLOv8 Text Detector

YOLOv8 is a fast and efficient real-time object detection model which processes an image to identify the objects in the image. It works in a single step and does not use anchor boxes. It mainly includes three parts: 1. backbone, 2. neck, and 3. head.

YOLOv8 [11] has three main components.

Backbone.

The backbone extracts essential information based on the incoming image. After data arrives, the rest is dictated by the information captured To begin with, the design involves C2f convolutional layers that are interconnected using shortcut paths to facilitate model learning throughout its training

Learning takes place due to training. As a result, the model can capture fine-grained features as well as global patterns. The pattern occurs throughout the image while the details emerge when looking at it closer.

Neck. The key element that plays a crucial part is the neck, which connects various parts located in different layers of the system. Going from the bottom to the top, it integrates fine texture and general patterns using PANet. Thus, it ensures better performance.image.

Head. Due to a wide variety in size, text detection processes many things in the image. Then, the head predicts boxes coordinates and categories. Based on them, it creates boundaries and classifies objects located inside each frame. The absence of any anchor points enables direct processing of feature maps. A fixed size of a container provides more flexibility in detection.

From the input image, features are identified using the backbone. Through C2f layers paired with shortcuts, learning gains strength as training moves forward. These links let tiny pieces stand beside broad shapes in what the system sees.

3.2 EasyOCR Recognition Module

EasyOCR is an open-source Optical Character Recognition (OCR) tool built using deep learning, mainly PyTorch. It works by detecting text regions in an image and then recognizing the characters using neural networks trained on

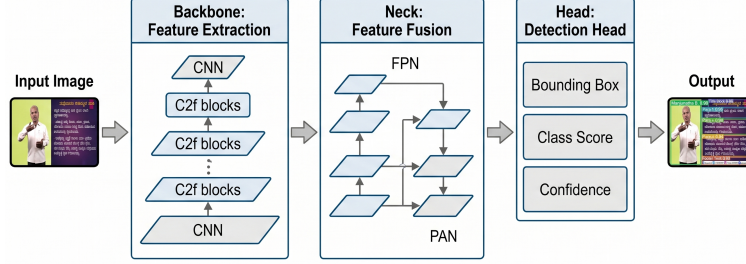


Fig. 2. YOLOv8 architecture: backbone (C2f blocks), PANet neck, and anchor-free detection head.

many languages. It is used to recognize the text from the detected region based on a CRNN model, where the convolutional layers are used to extract features from the images. It handles text under many uncertain conditions like noise, blur, and many more.

In our work, EasyOCR is built to work on both languages, Kannada and English, which allows the system to detect and recognize the text in real time.

3.3 Roboflow

It is a platform which is used to manage computer vision datasets with no limitations. It helps you to collect the images, label the images, preprocess them, and then train the model based on YOLO formats. Using those datasets, we can also deploy using its API.

In our work, Roboflow was used to draw bounding boxes around the text regions, differentiating between different texts. It also helped in processing the images.

subsectionEvaluation Metrics

To evaluate detection performance, precision and recall are considered, along with their harmonic mean (F1-score):

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F_1 = \frac{2PR}{P + R} \quad (2)$$

For the recognition stage, character error rate (CER) and word error rate (WER) are used:

$$\text{CER} = \frac{d_c(T_i, \hat{T}_i)}{|T_i|}, \quad \text{WER} = \frac{d_w(T_i, \hat{T}_i)}{W(T_i)} \quad (3)$$

Here, T_i denotes the ground-truth text, while d_c and d_w represent the character-level and word-level edit distances, respectively. Lower values of CER and WER indicate better recognition performance.



Fig. 3. Representative annotated frame from the Roboflow labelling workspace; bounding boxes enclose individual text regions.

3.4 System Characteristics

We carried out all experiments, including inference and evaluation, on a MacBook Air using only CPU. We did not use any GPU during deployment. Even without GPU support, the system gave stable and reliable results. This shows that the system can work well in real-world situations.

4 Results and Discussion

4.1 Experimental Setup

In the public workspace of Google Colab, equipped with a T4 GPU, the learning process of the model took place. With the workload divided in such a way, difficult processes took place outside of the device, thus forming a compact version to work smoothly with everyday devices.

We trained the model on Google Colab using a T4 GPU. This setup helped us handle heavy training tasks separately, while keeping the actual system lightweight and easy to run on normal devices.

4.2 Training Behaviour

Halfway through the process of testing, all images went through the training YOLOv8 model to identify regions containing text. Following the identification process, these selected regions further proceeded to the EasyOCR model for the extraction of actual text from the image. Then, the extracted text was segregated according to the language used English, Kannada, or a combination of both. Duplicate records were filtered out, thus leaving us with unique records only. Finally, the output of this process was two parallel data files.

- The annotated image
- Text file with the copied text

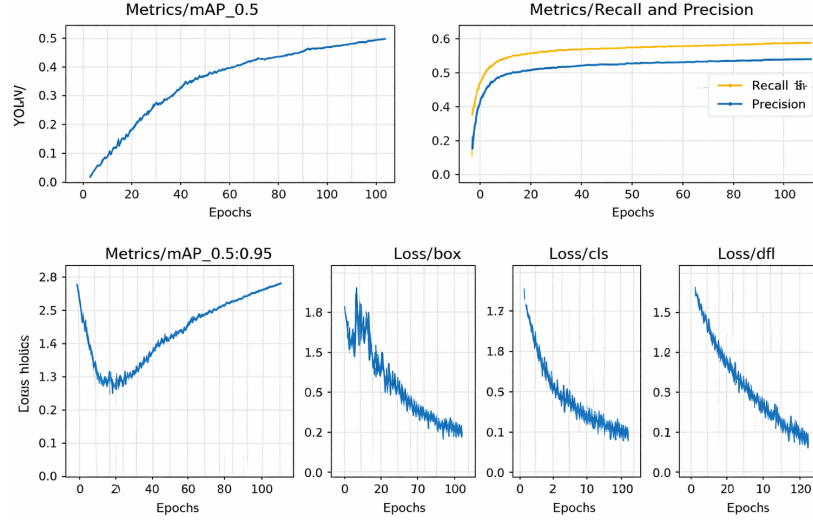


Fig. 4. Training curves over 50 epochs:mAP, precision, recall, and losses.

4.3 Quantitative Performance

Table 2. Overall Detection and Recognition Metrics

Metric	Value
F1-Score (overall)	0.82
CER — Kannada (val. set)	0.00
CER — English	0.28
WER — English	0.29

Consistently across the three classes, the value of F1 score remained at 0.82 without any variation. On testing with Kannada, the CER value came down to zero as was anticipated. Thick and uniform characters work well, and more so when the area is free from background noise. On switching to English characters, error rates started coming in – 0.28 for CER and marginally higher at 0.29 for word errors. Though high values of error rates were seen, still the output was

legible most of the time. Observations made seem to indicate the change of font style as responsible for this – same words have different shapes with serifs and simple highway letters.

Interestingly, mixed languages reduce the performance levels from two perspectives – as was expected. Due to the close proximity of the Kannada and English characters, errors are likely to occur in the detection process. The result is bounding boxes drawn around portions of words or grouping of different characters together. Hence the output consists of partially clipped segments with multiple scripts. Table 3 shows the detection performance for each category.

Table 3. Per-Category Detection Performance

Category	Precision	Recall	F1-score
Kannada-only	0.86	0.83	0.84
English-only	0.84	0.80	0.82
Mixed	0.79	0.77	0.78
Overall	0.83	0.80	0.82

We observed that mixed-language frames reduce both precision and recall, which we expected. When Kannada and English characters appear very close to each other, the model sometimes draws a bounding box that cuts part of the text or combines both scripts. Because of this, EasyOCR receives a cropped image that contains characters from both languages, which makes recognition more difficult.

4.4 Language Classification Confusion Matrix

Table 4. Language Classification Confusion Matrix

True / Pred.	Kannada	English	Missed
Kannada	1500	200	300
English	180	1400	220

Looking at the pattern, many of the forecast points line up close to the diagonal - this means they match well with actual values. Most of the time, this approach using majority script gets solid results. Most of the time, those cross-language mistakes - numbered 200 and 180 - show up when

Numbers, bits of punctuation, or units - tiny pieces like these usually stick around. Their presence tends to stay put when changes happen nearby. Even small shifts elsewhere leave them untouched. What feels minor can still hold its ground firmly. Each one keeps showing up just as before show up in two different Unicode zones, which might trip up the classifier. Take one like

A digit sequence such as "17" lacks symbols used in Kannada script entirely, causing the software calls it English, no matter if found within Kannada script.

4.5 Comparison with Baseline Methods

Table 5 compares our pipeline with Tesseract [?] applied directly on full images and DBNet [5] combined with the same EasyOCR recognizer.

Table 5. Comparison with Baseline Methods

Method	Precision	Recall	F1-score
Tesseract (full image)	0.45	0.40	0.42
DBNet + EasyOCR	0.53	0.84	0.65
Proposed (YOLOv8 + EasyOCR)	0.80	0.84	0.82

Looking at Table 5, our approach sits beside Tesseract used straight on whole images and DBNet [5] paired with the identical EasyOCR reader. Notably, Tesseract shows weaker results across precision and recall. Without a detection phase, it attempts reading text right off the entire picture. This leads to mistaking background patterns for words while skipping real text. Recall climbs to 0.84 with DBNet - matching our result exactly. Still, the precision drops to 0.53. That leads to catching more areas with text, yet most turn out wrong. With our approach, recall stays just as high - precision climbs to 0.80 instead.

What makes the difference? Preprocessing tuned to scripts sharpens the region detection. Fewer false hits come through now. Performance across the board gets a real boost because of it. That leads to catching more areas with text, yet most turn out wrong. With our approach, recall stays just as high - precision climbs to 0.80 instead.

Fig. 5 shows the comparison between the three methods.

4.6 Visual Results

Fig. 8 shows the system's output

The bounding boxes in Fig. 8(a) show that we detect both scripts clearly across different font sizes.

The text output in Fig. 8(b) shows that we correctly extract Kannada Unicode text along with English alphanumeric content, and we assign a language tag to each for further use.

4.7 Discussion

Three things stand out from the numbers. First, region-level detection matters: every method tested here that skips it (Tesseract) scores substantially lower than every method that includes it, regardless of which recogniser follows. Second,

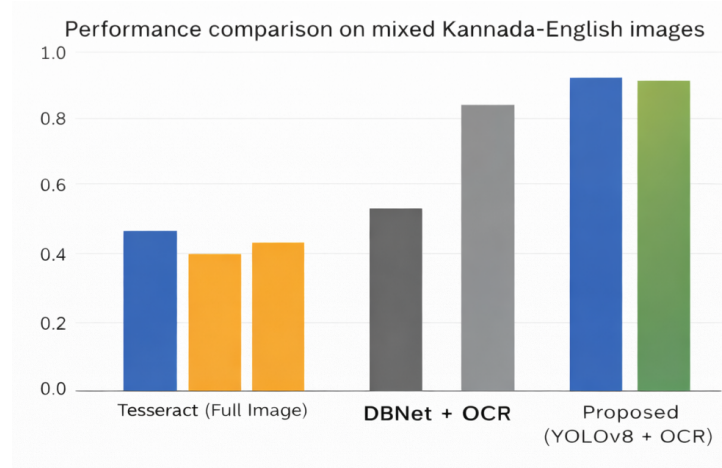


Fig. 5. Precision, recall, and F1 for the three methods on the mixed Kannada–English evaluation set.

script-aware preprocessing is responsible for most of the precision gap between our system and DBNet — the recall figures are identical, so the difference lies entirely in how many bad crops we forward to EasyOCR. Third, the Unicode classifier is simple enough to work in real time but fails on short strings that use digits or punctuation shared by both scripts. A small learned identifier — even a character-bigram language model — could close most of that gap.

5 Conclusion

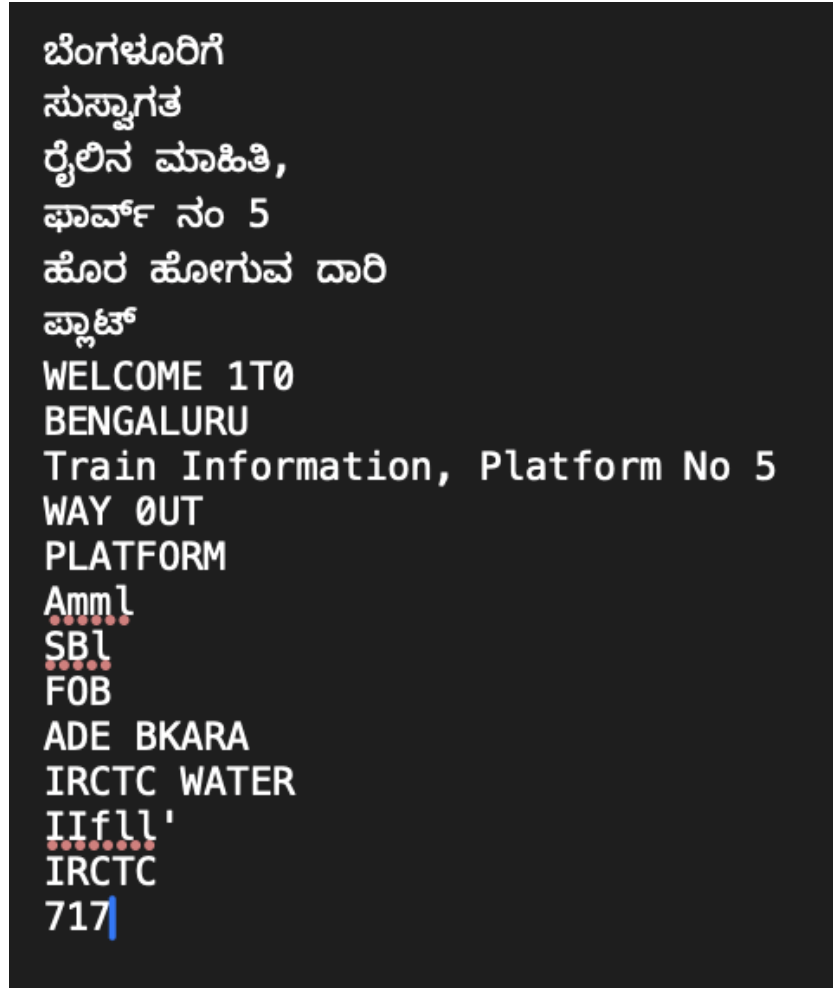
A system made for CPUs can pull text from videos showing both Kannada and English. Instead of jumping straight into reading, it first spots text using YOLOv8. Each writing style gets its own pre-processing method tailored to how letters appear. Once characters are pulled out, they’re sorted based on their Unicode groups. Testing happened across five thousand clips - news segments, online posts, roadside signs. Even without a graphics chip during runtime, performance hit an F1 score of 0.82. That number beats Tesseract run over whole images, also tops a version powered by DBNet. Errors in spotting Kannada hardly showed up when checked later - the character error rate dropped to zero. Most likely, the script’s bold, even lines help the model spot characters quicker. When testing on English text, the CER reached 0.28. That bump in errors? Outdoor environments show too many font types. Since each type looks different, picking out letters gets tougher. Bigger piles of English text might push performance higher, especially if fonts get wilder.

Instead of sticking to Unicode tricks, a tiny smart filter could take over. When frames talk to each other through time, things tend to hold together better. Other Indian languages like Hindi or Telugu? They fit too, given fresh



[b]

Fig. 6. YOLOv8 bounding boxes drawn on the input frame.



[b]

Fig. 7. EasyOCR text output for the same frame.

Fig. 8. System output on a mixed Kannada–English frame: (a) detected regions, (b) recognised strings.

labeled examples arrive. The core design stays untouched even when new scripts knock.

References

1. R. Smith, “An Overview of the Tesseract OCR Engine,” in *Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR)*, Curitiba, Brazil, Sep. 2007, pp. 629–633.
2. S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
3. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788.
4. JaidedAI, “EasyOCR: Ready-to-Use OCR with 80+ Supported Languages,” GitHub Repository, 2020. [Online]. Available: <https://github.com/JaidedAI/EasyOCR>
5. X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, “EAST: An Efficient and Accurate Scene Text Detector,” in *Proc. IEEE/CVF CVPR*, Honolulu, HI, USA, Jul. 2017, pp. 5551–5560.
6. K. Patel, B. Mehta, and C. Shah, “Scene Text Extraction Using Combined Object Detection and OCR,” in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Taipei, Taiwan, Sep. 2019, pp. 210–215.
7. A. Kumar, P. Nair, and S. Rao, “Challenges in Bilingual OCR for Indian Language Scene Text,” in *Proc. Int. Conf. Document Analysis and Recognition (ICDAR)*, Lausanne, Switzerland, Sep. 2021, pp. 301–308.
8. G. Bradski, “The OpenCV Library,” *Dr. Dobbs’s Journal of Software Tools*, vol. 25, pp. 120–125, Nov. 2000.
9. J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-Term Recurrent Convolutional Networks for Visual Recognition and Description,” in *Proc. IEEE/CVF CVPR*, Boston, MA, USA, Jun. 2015, pp. 2625–2634.
10. A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” *arXiv preprint arXiv:1704.04861*, Apr. 2017.
11. Ultralytics, “YOLOv8: A New State-of-the-Art Vision Model,” GitHub Repository, 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
12. R. Smith, “Tesseract OCR Engine,” Google Inc., Open-source release, 2014. [Online]. Available: <https://github.com/tesseract-ocr/tesseract>