Printed Gujarati Text Detection and Recognition (PGTDR)

Sayali Ambure
Dept. of Information Technology
Fr. C. Rodrigues Institute of Technology, Navi Mumbai, India
sayaliambure15@gmail.com

Lisha Kothari
Dept. of Information Technology
Fr. C. Rodrigues Institute of Technology, Navi Mumbai, India
lishakothari02@gmail.com

Niraj Patil
Dept. of Information Technology
Fr. C. Rodrigues Institute of Technology, Navi Mumbai, India
nirajpatil849@gmail.com

Tejas Patil
Dept. of Information Technology
Fr. C. Rodrigues Institute of Technology, Navi Mumbai, India
teja36155@gmail.com

Dr. Archana Shirke
Dept. of Information Technology
Fr. C. Rodrigues Institute of Technology, Navi Mumbai, India
archana.shirke@fcrit.ac.in

Dr. Shashikant Dugad
Dept. of High Energy Physics
Tata Institute of Fundamental Research, Mumbai, India
shashi@tifr.res.in

***Abstract*—India is the land of many historic manuscripts and parchments. More than one-third Indian manuscripts are written in Gujarati, according to statistics from the National Mission for Manuscripts. Since handwritten data on paper is difficult to maintain, digitizing these manuscripts and storing them in a database proves helpful in their preservation. The project will use machine learning models trained on a custom dataset to recognize Gujarati text and convert it into a machine-readable format. The project presents an effective, robust and high-precision system to conveniently digitize these manuscripts, using YOLOv8 for character detection with 98% precision and DenseNet model for text recognition with 99.2% accuracy. The system is also capable of detecting text blocks in newspaper articles and this proof of concept will be extremely efficient for detection of texts in Gujarati manuscripts.**

Keywords—Gujarati, YOLO, handwritten, digitize, EfficientNet

I. Introduction

Text recognition is the process of transforming a picture of text into actual text so that it may be used by any technology. The transformed text can then be applied to additional processing. Applications for text recognition may be found in security, healthcare, retail, text translation, and inventory management. This novel method called Printed Gujarati Text Detection and identification (PGTDR) makes use of DenseNet for text extraction and identification and YOLOv8 to identify individual Gujarati characters. The aim of this paper is to examine existing systems and evaluate the efficiency of various YOLO models and deep learning frameworks comprehensively and propose the most appropriate ways to address a variety of new input images. You Only Look Once [1] (YOLO) is an algorithm that locates and detects various objects in photos in real-time. DenseNet [2], which stands for "Densely Connected Convolutional Network," is a foundational neural network design that has had a considerable impact on deep learning and computer vision. DenseNet, developed by Huang et al. in 2016, is notable for its unique connectivity pattern, which is characterized by dense connections between layers. In a densely linked block, every layer in a DenseNet receives direct input from all preceding layers, facilitating effective feature reuse and gradient flow during training without encountering the vanishing gradient problem.

II. Related Work

Many efforts have been put into the development of a robust system for digitisation of Gujarati text recognition. Joel D' Silva et al. present an innovative approach in which the text is horizontally divided into three segments. The words are categorized into three distinct zones: one housing vowel diacritics positioned above the base letter, another focusing on the base letter itself, and a third accommodating vowel diacritics below the base letter. Following the completion of post-processing, the final output is derived from the integration of these three specified zones.

PGTDR on the other hand, makes use of bounding boxes generated from the detection module of YOLOv8

979-8-3503-5981-7/23/$31.00 ©2023 IEEE

to detect individual characters and matches them to a labeled set generated using TRDG to accurately recognise the character. The final output is the prediction of the text post detection and recognition by the system.

III. Proposed System

PGTDR's primary objective is to digitally format and extract text from input pictures. The system suggests a method where text is extracted using an object detection algorithm YOLOv8. The system has 3 main components - input data, training, and inference. The input data module consists of the images and related annotations from the training dataset. The training dataset contains generated images and labels for each image. Every character detected in the boundary coordinates of an image is recorded in a text file called an annotation. The training module receives the input and cleans and optimizes the data. The neural network extracts the character traits. The weight files needed to represent the trained model are created after an appropriate number of iterations. The entire process is elucidated in detail in this section -

A. Data Generation and Annotation Process

Text Recognition Data Generator (TRDG) [3] creates images of words in various fonts and formats. The images are generated using the GUI utility for ease of access to TRDG against the command line and for each word, an image is created using the font, backdrop, and additional alterations like skewing or blurring. TRDG can also generate the bounding boxes for each character in the image. Figure 1 shows a dictionary of words created using the TRDG utility

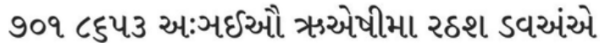

Fig. 1. Dictionary of words

B. Training and Testing

These pictures are then trained using the YOLOv8 model. Every character detected in the boundary coordinates of an image is recorded in an annotated text file. This is then fed as an input to the YOLO algorithm to pinpoint the objects, in this case, characters, that it seeks to identify. The system creates two folders named “train” and “valid” which contain the preliminary generated images and their labeled counter-parts. The following three primary routes are necessary for the model: Paths for Datasets, Training, and Validation

The user defines the dataset path, and the model then determines the training and validation paths from that. The training module receives the input images and annotations and uses them to clean up and improve the data. A sufficient number of iterations later, training is finished, and weight files corresponding to the trained model are produced. We trained four iterations of the YOLO algorithm to select the most suitable model based on their performance —YOLOv5, YOLOv6, YOLOv7, and YOLOv8. The two YOLO versions that were carefully examined are YOLOv5 and YOLOv8 after a preliminary examination. 200 images were first utilized to train the YOLOv5 model, and finally 1000 photos were employed. 200 photos in total were used to train the YOLOv8 model at first, then 6000 images across 100 epochs were used afterwards..

C. Recognition

After the initial text detection phase performed by YOLOv8, which generates bounding boxes around text regions, we extract the identified text regions from the original image. This extraction process determines the exact part of the image specified by the bounding box coordinates provided by YOLOv8. Specifically, for each text area, we isolate the relevant pixels from the original image based on the box coordinates. After region extraction, a crucial preprocessing phase is initiated to prepare these regions for detection. A DenseNet-based model is then used to recognise text from the images. In this context, DenseNet acts as an OCR (Optical Character Recognition) tool. It is designed to recognize individual characters within text areas. The final output then shows us the recognised form of the input images. Lasty, the proposed system is also capable of recognizing blocks of text from newspapers. Figure 2 shows its ability to detect text blocks. This is an important feature since most manuscripts contain paragraphs and separated shlokas. The system’s ability to clearly demarcate paragraphs will help build a similar capability for manuscripts and ancient scrolls.

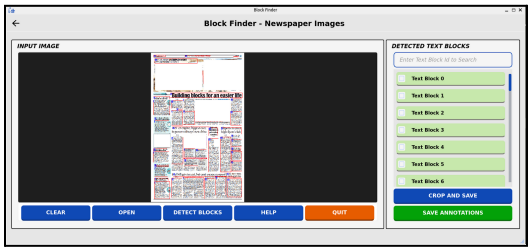

Fig. 2. Detection of Text Blocks

IV. Results

The dataset was trained using YOLOv5, YOLOv6, YOLOv7 and YOLOv8 algorithms. Of all the 4 models, the YOLOv8 gives the maximum mAP with less time taken for training. as shown in table 1.

TABLE I: Details of the 4 models trained

	YOLO v5	YOLO v6	YOLO v7	YOLO v8
Time to Train to 30 epochs	7min 48s	14 min 2s	16min 40s	12min 25s
Model Parameters (Million)	20.87 M	17.2 M	37.2 M	25.8 M
Inference Time	19ms	22ms	19ms	16.1 ms

Diverging from its predecessor, which relied on predicting an object's offset from predefined anchor boxes, YOLOv8 introduces the novel approach of directly predicting the object's center. This innovation simplifies the object detection process significantly. Furthermore, with the aim of expediting the computationally intensive Non-Maximum Suppression (NMS) step, a critical component in refining candidate detections during inference, anchorless detection reduces the volume of box predictions. As portrayed in Figure 3, the system conducts non-overlapping detection on the input image, with the results visually represented in Figure 4.

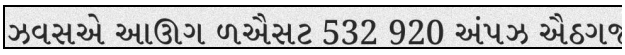

Fig. 3. Input Image

Fig. 4. Boundary-Boxes around separate characters

The YOLOv8 model demonstrated impressive precision in its detections, achieving a rate of 98.97%, along with a commendable F1-score of 85.66%. Figure 5 illustrates the label versus prediction training results, while Figure 6 provides a comprehensive training summary. Within this framework, the box-loss metric gauges the model's proficiency in precisely locating an object's center and defining an accurate bounding box around it. Obj-loss, on the other hand, quantifies the likelihood that the object is correctly positioned within the recommended region. Lastly, the cls-loss metric informs us about how effectively the model identifies the correct object class. Detailed analysis, as revealed through precision-recall and mean average precision (map) plots, demonstrates that the model's performance exhibits a plateauing effect at approximately 40 epochs.

Fig. 5. Label of images and predicted results after training

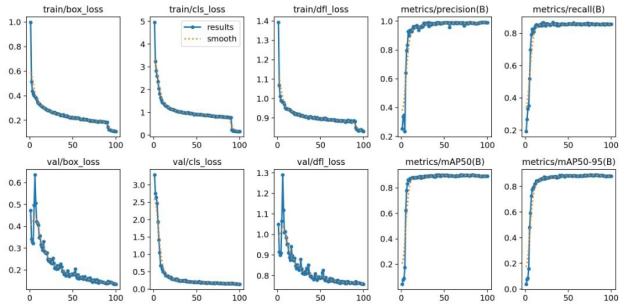

Fig. 6. Performance Overview

The final Summary of training is as follows -

1. number of images- 6000
2. number of epochs- 100
3. time for preprocess- 0.1ms / image
4. time for inference- 2.7ms / image
5. time for postprocess- 1.6ms / image

Subsequently, the following steps include recognition and post-processing operations. To achieve this, a comprehensive exploration and training of various deep learning models were conducted, with the aim of assessing their performance and selecting the most apt model tailored to our specific use case. The comparative analysis of our findings are elucidated in this section.

A. EfficientNet

EfficientNet [4], a neural network architecture, has emerged as a pivotal innovation in the field of deep learning. It was introduced by Tan and Le in 2019 as a scalable and efficient model that significantly improves model efficiency while maintaining state-of-the-art performance on various computer vision tasks. EfficientNet leverages a novel compound scaling method that simultaneously scales an image's depth, width, and resolution, allowing it to adapt to different computational constraints. This adaptability makes EfficientNet versatile, suitable for applications ranging from mobile

devices with limited computational resources to high-performance servers. As illustrated in Figure 7, which showcases the training accuracy and loss trends over 10 epochs.

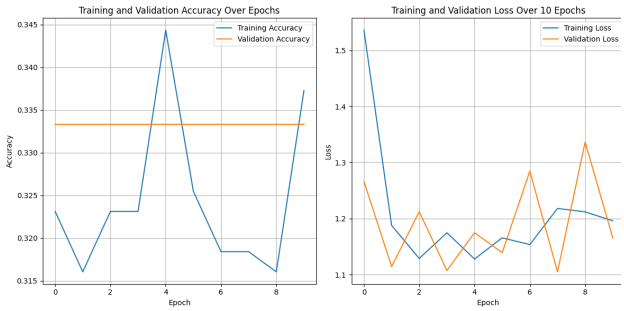

Fig. 7. Accuracy and Loss for EfficientNet Model

B. DenseNet

As illustrated in Figure 8, which showcases the training accuracy and loss trends over 10 epochs, DenseNet excels in optimizing model training and attaining superior performance when compared to EfficientNet, making it a compelling choice in the realm of deep learning architectures.

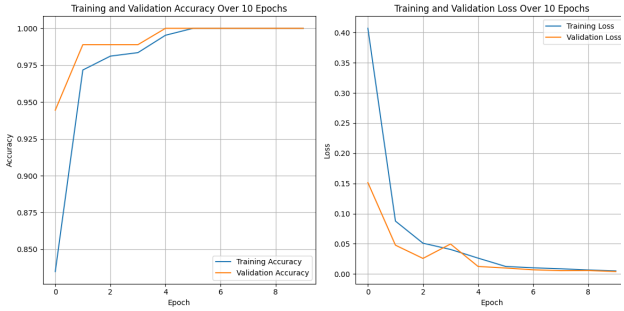

Fig. 8. Accuracy and Loss for DenseNet Model

C. NasNet

NasNet [5], or Neural Architecture Search Network, is a groundbreaking neural network architecture known for its automated design process. It uses a neural architecture search algorithm to automatically discover and optimize network architectures. This approach has the potential to significantly improve both performance and efficiency. Figure 9 below illustrates the training accuracy and loss trends over 10 epochs, highlighting the relative performance of NASNet.

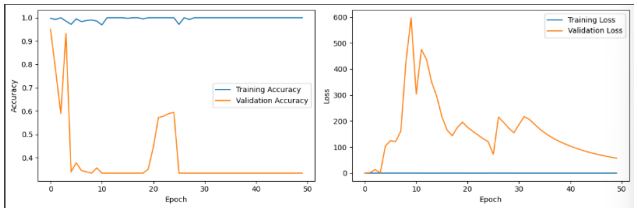

Fig. 9. Accuracy and Loss of NasNet Model

D. ResNet

ResNet [6], short for Residual Network, represents a pivotal advancement in deep neural network architecture. It was introduced by Kaiming He et al. in 2015, and it addresses the vanishing gradient problem by introducing the concept of residual connections or skip connections. These skip connections allow the network to skip one or more layers, making it easier to train very deep networks. ResNet's design has demonstrated remarkable success in various computer vision tasks, such as image classification, object detection, and segmentation. Figure 10 provides a visual representation of the training accuracy and loss trends over 10 epochs, highlighting the performance of ResNet compared to other models.

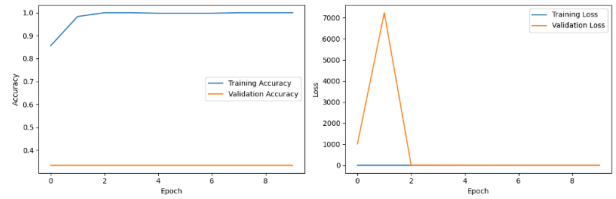

Fig. 10. Accuracy and Loss for ResNet Model

E. VGG16

VGG16 [7], part of the Visual Geometry Groups architecture series, is characterized by its simplicity and depth. It consists of a stack of convolution layers, each using a 3x3 convolution filter. Despite its straightforward design, VGG16 has made significant contributions to the field of computer vision. However, in our study, we found that performance was below average, as shown in Figure 11

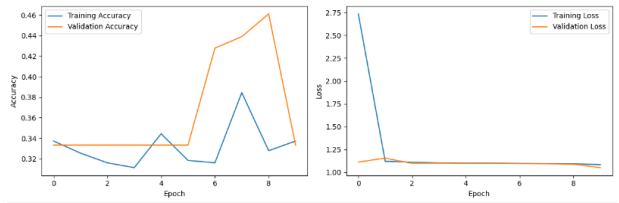

Fig. 11. Accuracy and Loss forVGG16 Model

F. InceptionV3

InceptionV3 [8] is highly regarded for its precision, resilience, and capacity to effectively process intricate and diverse datasets. Its design and performance characteristics render it the top choice for computer vision applications, as substantiated by the results presented in Figure 12. Further details pertaining to the training of all the models can be found in Table II.

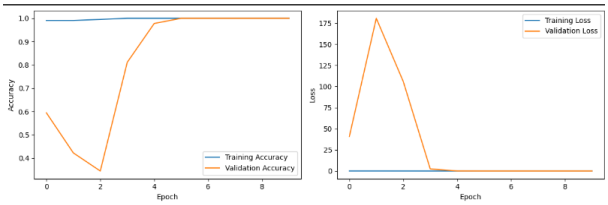

Fig. 12. Accuracy and Loss for InceptionV3 Model

Table II. Model Performance Summary

Model	Accuracy (%)	Inference Speed (s)
EfficientNet	33.33%	6.60
DenseNet	99.2%	7.32
InceptionV3	98%	9.05
VGG16	57%	7.71
NASNet	35%	10.62
ResNet	31%	10.89

As seen, DenseNet and InceptionV3 provided the best accuracy rates for our custom dataset. Since DenseNet gave us better accuracy at an expense of lower inference speed, it has been used for further executions on the complete dataset. Our text extraction and recognition process concludes with an impressive overall accuracy of 99.2%. Figure 13 visually presents both correctly identified characters and any occasional misclassifications, along with the recognized characters from the input image and their extracted text counterparts for easy assessment.

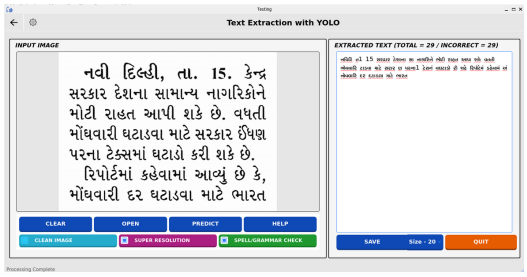

Fig 13. Final Output

V. Conclusion

Text recognition stands as a pivotal pursuit in the landscape of deep learning, especially as the preservation of ancient knowledge contained in manuscripts gains ever-increasing significance with the passage of time. This project introduces an efficient and precise approach to identify typed Gujarati script and transform it into a digital format. To strike a balance between accuracy and processing speed, we adopt the robust Progressive Growing Text Detection and Recognition (PGTDR) methodology. Our study spans a thorough exploration of various YOLO models, assessing their performance, and a comprehensive deployment of deep learning models for text detection. The YOLO model articulated in this research demonstrates remarkable accuracy in both text extraction and recognition. It is worth noting that the model's current configuration is optimized for typed scripts but holds the potential to extend its utility to handwritten scripts. In essence, our research underscores the utmost significance of digitizing and conserving Gujarati scripts.

References

[1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[2] "Review: DenseNet — Dense Convolutional Network (Image Classification)" Nov 25, 2018. [Online]. Available: https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

[3] E. Belval, "TextRecognitionDataGenerator," GitHub, Nov. 11, 2022.https://github.com/Belval/TextRecognitionDataGenerator

[4] Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by Ultralytics (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics

[5] R. Wang, Y. Li, Y. Duan and T. Tan, "EfficientNet-YOLOv5: Improved YOLOv5 Based on EfficientNet Backbone for Object Detection on Marine Microalgae," 2022 6th International Conference on Universal Village (UV), Boston, MA, USA, 2022, pp. 1-5, doi: 10.1109/UV56588.2022.10185489.

[6] D. R. N. Ashlin, Y. Vijayalata and A. Negi, "Document Text Analysis and Recognition of Handwritten Telugu Scripts," 2022 IEEE 4th International Conference on Cybernetics, Cognition

and Machine Learning Applications (ICCCMLA), Goa, India, 2022, pp. 462-466, doi: 10.1109/ICCCMLA56841.2022.9989012.

[7] J. Tao, Y. Gu, J. Sun, Y. Bie and H. Wang, "Research on vgg16 convolutional neural network feature classification algorithm based on Transfer Learning," 2021 2nd China International SAR Symposium (CISS), Shanghai, China, 2021, pp. 1-3, doi: 10.23919/CISS51089.2021.9652277.

[8] "Inception V3 CNN Architecture Explained)" Oct 24, 2021. [Online]. Available: https://medium.com/@AnasBrital98/inception-v3-cnn-architecture-explained-691cfb7bba08