MARATHI OPTICAL CHARACTER RECOGNITION

Utkarsh Pandey - U23AI040
Aryan Sawant - U23AI042
Devang Vala - U23AI044
Jeet Tandel - U23AI045
Krish Rathod - U23AI049

INTRODUCTION

Optical Character Recognition (OCR) helps in converting handwritten or printed text into machine-readable text.

While OCR for printed text is mature, handwritten Indic scripts like Marathi still pose challenges due to:

- Complex character shapes
- Shirorekha (headline)
- Lack of clean segmentation between words and characters

We propose an OCR model tailored for handwritten Marathi sentences.

MOTIVATION/BACKGROUND

Optical Character Recognition (OCR) is the technology used to convert printed, typed, or handwritten text into machine-readable digital text. It's widely used in document digitization, translation, and data automation.

Timeline Overview:

1929 – Early Invention: Gustav Tauschek builds one of the first OCR machines in Germany.
1950s – Commercial Use: IBM develops OCR for typed English text.
1970s – Postal Systems: OCR used for sorting ZIP codes.
1980s–1990s – Handwriting Focus: Research into handwritten digit recognition; MNIST becomes a standard dataset.
2000s – Machine Learning Era: Use of SVMs and K-NN improves recognition, especially for handwritten text.
2010s – Deep Learning Revolution: CNNs enable accurate recognition of cursive, non-Latin scripts, and noisy images.
2015–Present – Full Sentence Recognition: Models like CRNNs and Transformers allow end-to-end sentence OCR with improved context through NLP integration.

Our motivation: Enable reliable sentence-level recognition from real handwritten Marathi inputs.

RELATED WORKS (EXISTING WORK)

[1] Gujarati Handwritten Character Recognition - *Jyoti Pareek et al. (2020)*

- Developed an offline OCR system for Gujarati characters
- Employed CNN and MLP classifiers
- Achieved 97.21% accuracy using CNN
- Highlighted challenges in HCR vs. printed OCR
- Focused on character-level recognition, not complete word/sentence conversion

🔗 https://doi.org/10.1109/SKIMA.2015.7400041

[2] *Devanagari OCR Using Deep Learning - Acharya, Pant & Gyawali (2015)*

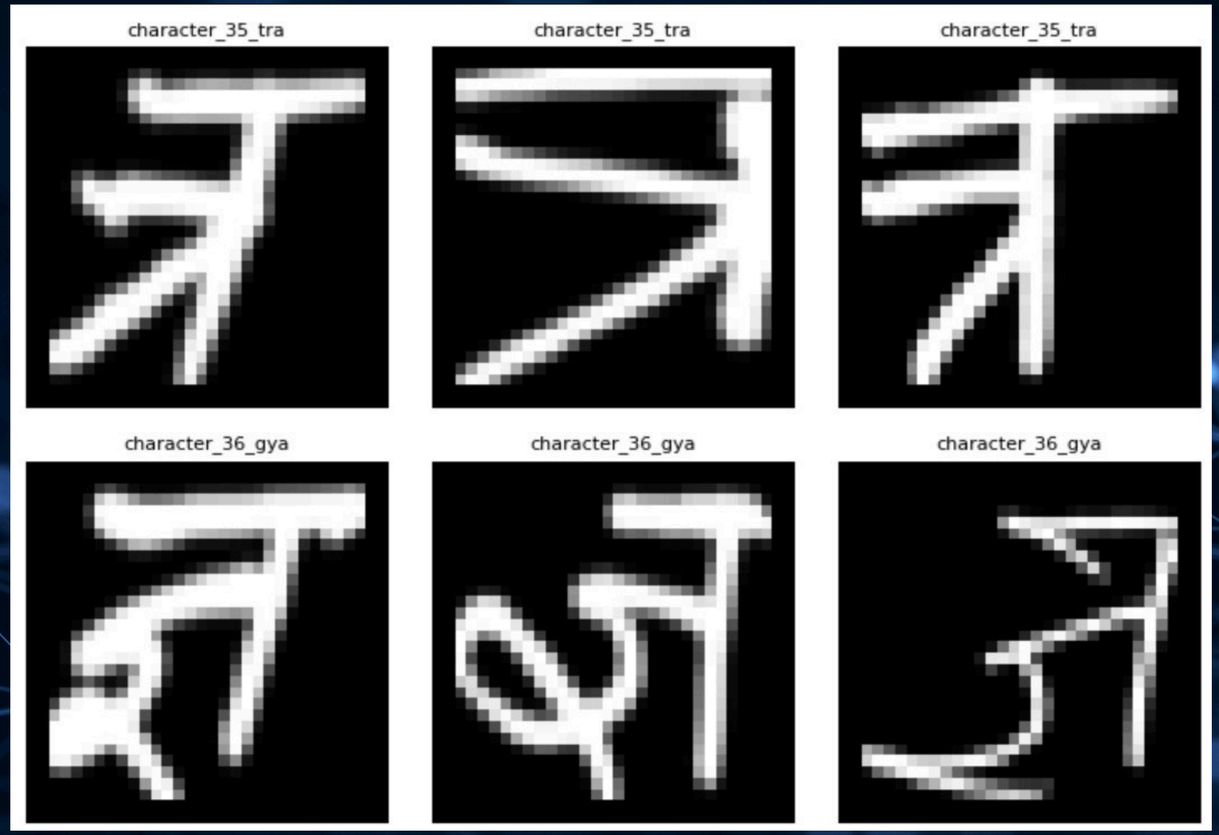
- *Proposed CNN-based OCR system for Devanagari script*
- *Created a large-scale dataset with 46 character classes*
- *Used dropout and convolutional layers to prevent overfitting*
- *Achieved high accuracy on character-level recognition*

🔗*https://www.sciencedirect.com/science/article/pii/S187705092031022X?via%3Dihub*

DATASET OVERVIEW

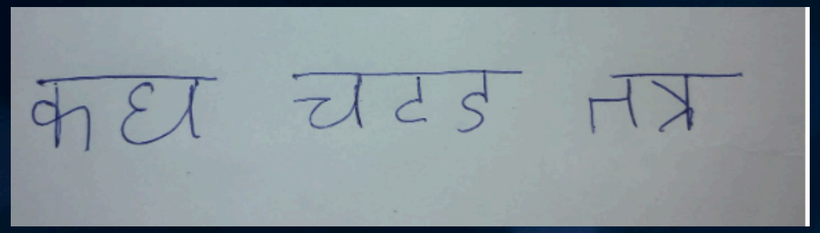
Acharya, S., & Gyawali, P. (2015). Devanagari Handwritten Character Dataset [Dataset]. UCI Machine Learning Repository.

https://doi.org/10.24432/C5XS53



(Sample Image for Dataset)

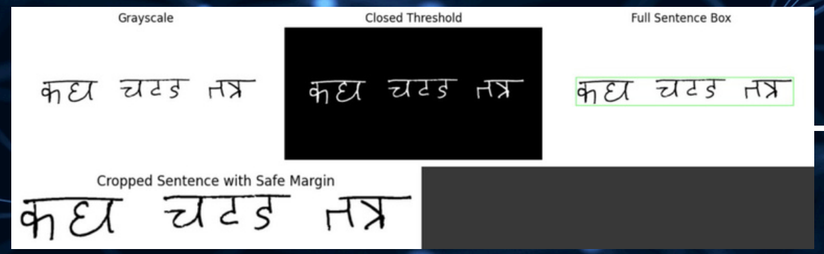
FLOWCHART OF MECHANISM



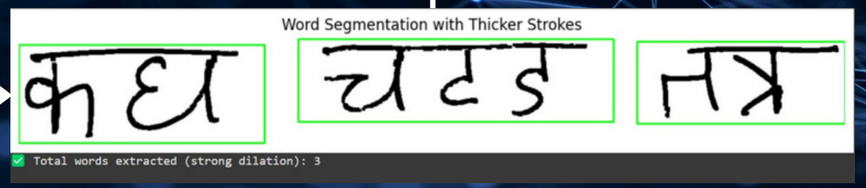
Read a Handwritten Sentence Image

Preprocessing:
- Grayscale
- Thresholding
- Denoising
↓
Detect Text Region using Contours
↓
Crop ROI (Text Area)

OpenCV

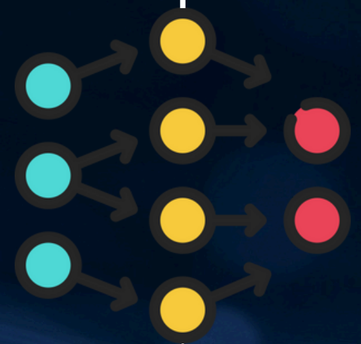


Crop Just the Text
(Remove Background/Noise)

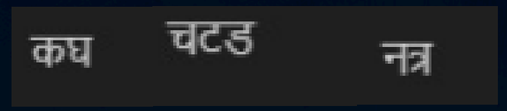


Split into Words
- Horizontal Projection
- Contour Detection

Split Each Word into Characters
- Vertical Projection



Classify Each Character
using Trained CNN Model



Show and Print
Predicted Devanagari Text

IMPLEMENTATION ENVIRONMENT & RESULT

Tools & Libraries Used:

Python: Primary programming language.

OpenCV: Image processing tasks (preprocessing, segmentation).

TensorFlow/Keras: Building and training the OCR CNN model.

Matplotlib: Visualization for debugging segmentation results.

Preprocessing Techniques:

- **Gaussian Blur:**
  Reduce noise and variations in handwriting thickness.

- **Adaptive Thresholding:**
  Binarize images considering local pixel neighborhood, making it robust to lighting variations.

- **Dilation:**
  Connect broken character parts before contour detection.

IMPLEMENTATION ENVIRONMENT & RESULT

- **CNN Model Building**

```
model = keras.Sequential(
    [
        keras.Input(shape=INPUT_SHAPE),

        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),

        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),

        layers.Flatten(),
        layers.Dropout(0.5),                              # Dropout for regularization
        layers.Dense(256, activation="relu"),             # Add a dense layer
        layers.Dropout(0.3),                              # Optional dropout
        layers.Dense(num_classes, activation="softmax"),  # Output layer
    ]
)

model.summary()
```

IMPLEMENTATION ENVIRONMENT & RESULT

- **Noise Removal:**

Small components (like dots, noise) filtered out based on area or width.

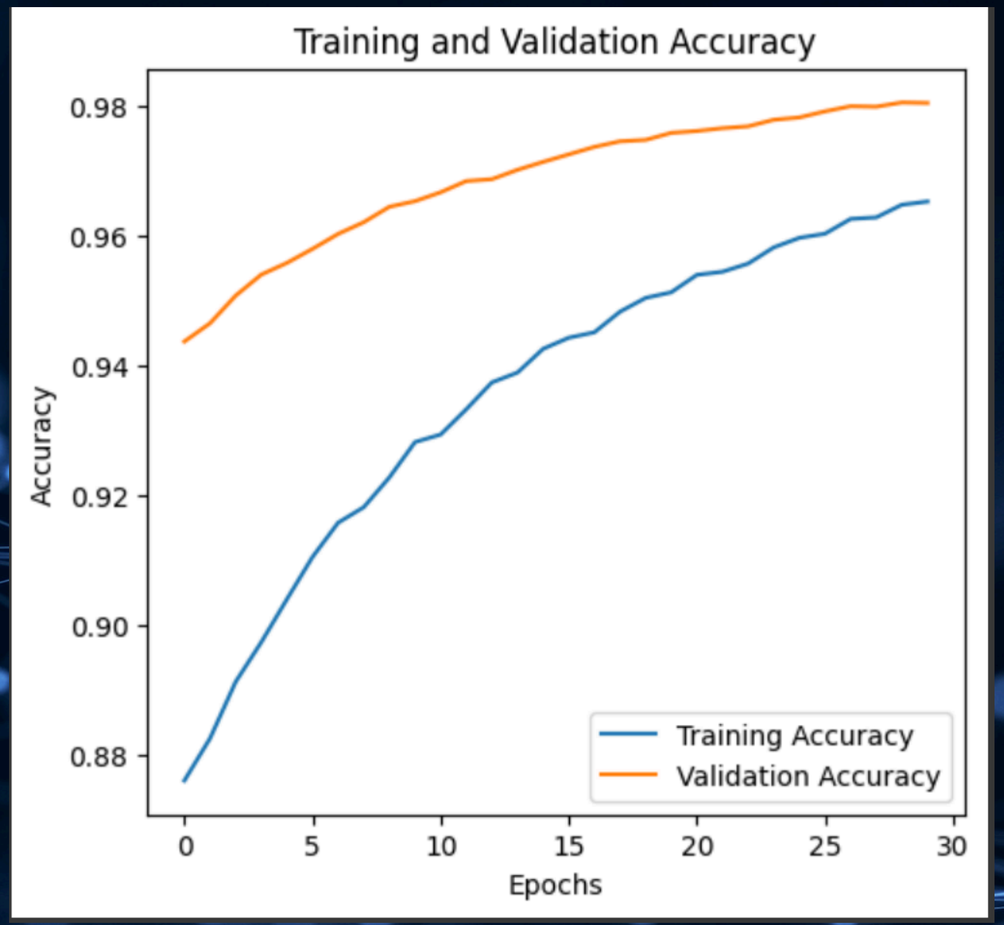
- **Model Input Format:**

Each segmented character image is:

- Grayscale inverted (background black, text white).
- Resized to 32×32 pixels.
- Normalized pixel values (0-1 range).

Model Results:

- **Validation Accuracy:** 98.60%
- **Testing/Real-World Accuracy:** 98.79%



CONCLUSION

Achievements:

- Used a combination of Shirorekha-based segmentation and CNN classification.
- Achieved satisfactory character recognition even in challenging handwritten data.

Challenges Faced:

- Variation in handwriting styles.
- Connected or overlapping characters.
- Dealing with matras (vowel signs) and modifiers.

FUTURE WORK

Improve Segmentation:

- To build an end-to-end OCR pipeline for handwritten Marathi sentences.
- Better detection for touching or overlapping characters.
- Smarter Shirorekha removal that preserves modifiers.

Expand Dataset:

- Collect more handwritten samples from diverse writers.
- Train with a wider range of writing styles for better generalization.

Thank You