[1,2]Buddaraju Revathi
[3] M. V. D. Prasad
[4]Naveen Kishore Gattim

Performance Comparison of various CNN Models for Recognition of Handwritten Telugu Text

Abstract: - Deep Learning (DL) stands as a pivotal field in the exploration of pattern recognition, offering unparalleled potential for addressing difficult machine learning challenges. The recognition of characters in the Telugu language using Optical Character Recognition (OCR) presents challenges due to the complex structures of characters, the presence of confusing characters, and overlapping characters. Convolutional Neural Networks (CNNs) exhibit proficiency in extracting features from training images and determining subtle differences in character shapes. The potency of robust CNN architectures has significantly elevated recognition rates, especially for Indian scripts. Our goal was to evaluate how well CNN based models adapt and perform with accuracy in a challenging script recognition context. In this paper, we introduce a character segmentation algorithm designed to address the challenges posed by overlapping characters, considering the Telugu language-specific features. Our approach involves a preprocessing stage to identify page boundaries and detect words within lines, employing edge detection algorithms. Subsequently, characters are extracted from words on the page using a character segmentation algorithm tailored for the Telugu language and the characters are recognized using trained deep learning models. In addressing the distinctive traits of the training data, we utilize a built based upon the Inception and ResNet models, incorporating adjustments in layers. The model's performance undergoes thorough validation using a standard dataset, and it is benchmarked against established models in the respective field.

Keywords: Deep learning, Convolution Neural Network, ResNet, Inception, Optical character recognition, Feature extraction.

I. INTRODUCTION

OCR technology plays a crucial role in safeguarding cultural legacies, advancing education, streamlining governance, rejuvenating languages, improving communication, and upholding legal standards, where regional languages hold sway. Its integration into regional languages significantly contributes to the broader socioeconomic progress of these regions [1]. The Telugu is a significant South Indian language, has been understudied, leading to inadequate solutions for recognizing its handwritten script. To address this gap, we have employed advanced deep learning models, well-known for their effectiveness in image recognition tasks.

Telugu language has hundreds of characters. There are 56 important base characters. Out of which 16 are vowels and the rest are consonants. For each consonant there exists gunintham which is formed by adding the symbolic representation of vowel with consonants. In addition, there are so many compound characters which are formed by adding symbolic ligature of a character to the base character. Most of the base characters have similar and intricate structures. Due to this the Telugu handwritten OCR has remained unaddressed problem.

This study makes significant strides in advancing OCR technology and plays a pivotal role in fostering linguistic diversity, preserving cultural heritage, enhancing education, and ensuring access to information in native languages. Our research involves an in-depth analysis of various CNN models applied to a dataset comprising Telugu handwritten texts. The primary aim is to evaluate the performance of these models and establish benchmarks for accuracy and efficiency in recognizing Telugu scripts.

OCR is implemented in five stages namely preprocessing, segmentation, feature extraction, training, validation and testing, recognition [2]. The preprocessing stage stands as a pivotal step, focusing on preparing the input image or document to ensure precise character recognition. This phase is dedicated to refining image quality, rectifying distortions, and eliminating any noise. Segmentation is the pivotal process of partitioning an image with

[1] Research Scholar, Department of ECE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh, India-522502.

E-mail: buddaraju.revathi@gmail.com

[2] Assistant Professor, Department of Electronics and Communication Engineering, SRKR Engineering College, Bhimavaram-534204.

[3] Associate Professor, ECE Department, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur,Andhra Pradesh, India-522502.

[4] Associate Professor, ECE Department, Sasi Institute of Technology and Engineering, Tadepalligudem.

text into smaller segments or sections, each corresponding to a single character, word, or line. By dissecting the text into discrete characters, OCR algorithms can concentrate on recognizing each character separately, thereby significantly enhancing accuracy [3]-[4].

Feature extraction is a pivotal process where input data, like images containing text, is transformed into significant features. Neural networks excel in this task by autonomously identifying relevant features from the data, thereby improving the accuracy and efficiency of the extraction process [5]-[6]. Training and testing form indispensable components of OCR system development. Training allows the system to acquire knowledge and adaptability, while testing serves to validate its performance, guaranteeing its ability to process varied, real-world data accurately and reliably.

Telugu OCR faces the problems with character segmentation and similarity of characters. For example, యు, ము; స, న; ధ, ద; ర, ర. These characters are called confusion characters in Telugu. A character segmentation technique has been developed to address segmentation issues and preserve vital features, especially for overlapping characters. To differentiate between similar characters, the ResNet Inception cascade model was employed, mitigating overfitting concerns often associated with deep models.

II. RELATED WORK

Enhancing OCR recognition rates is achieved through the segmentation of overlapping characters. In the context of Bangla and Devanagari scripts, Utpal Garain et.al [7] tackled character segmentation challenges, particularly in cases where characters are touching. Their approach involved employing fuzzy multifactorial analysis to effectively separate touching characters in these scripts. G. Louloudis et.al [8] utilized a Gaussian mixture model to compute distance between words and character spacing for handwritten word segmentation. This approach, based on Euclidean distance and convex hull-based metric, demonstrated superior performance on standard datasets compared to conventional methods. Jeewonoog Ryu et.al [9] employed dissimilarity scores among inter and intra-word gaps to segment words in handwritten scripts. Modeling the similarity score for inter-word gaps, along with individual gap likelihoods as a binary quadratic problem, the parameters were estimated by structured Support Vector Machine. This method outperformed on both Latin and Indian languages. Vishal Rajput et.al [10] developed a character segmentation technique by analyzing pixel values vertically to identify local minima and maxima points. Adjusting the graph adaptively based on peak intensity values widths from the vertical projection profile enabled correct segmentation points. While effective for connected text, the algorithm may require revision for handling overlapping characters.

With the evolution of neural networks, there has been a surge in research interest in OCR for native languages, driven by their proficiency in feature extraction. Haifeng Zhao et.al [11] conducted OCR research for Kannada and English scene text detection. The approach was based on VGG-Net with fewer units in the convolutional layer, employing diverse strategies for initialization to achieve improved accuracy rates. Minesh Mathew et.al [12] tackled text recognition in Telugu, Malayalam, and Devanagari scene images using a CNN-RNN model trained end-to-end on images. This model demonstrated proficiency in recognizing text by detecting words within the images. Konkimalla Chandra Prakash et.al [13] focused on Telugu OCR for printed text, utilizing connected components for segmenting images at the character level. CNN was employed for feature extraction of base characters, vattu, and gunintham. The Adam optimizer and CNN as a classifier were integral components of the approach. Syed Yasser Arafat et.al [14] employed faster RCNN alongside CNNs to localize text in Urdu scene images. Ligature alignment prediction through a regression residual network and the use of two-stream deep neural networks contributed to a practical recognition rate of 76.6%. Yongjie Zou et.al [15] leveraged Bi-LSTM with ambient position for license plate character detection. Popular CNN architectures with a spatial attention mechanism were used to extract character features, achieving significant recognition rates on both regular and irregular license plates.

A classification technique for recognizing handwritten Urdu characters and digits was introduced by A. Rasheed et.al [16] leveraging the principles of transfer learning with pre-trained CNNs to enhance accuracy. In the domain of handwritten Ethiopic texts, R. Malhotra et.al [17] harnessed deep neural networks, employing a comprehensive end-to-end approach for seamless feature extraction and efficient subsequent detection. The model's core incorporates an attention technique combined with connectionist temporal classification, and the architecture features seven CNNs and two recurrent neural networks.

Segmenting overlapping Telugu characters requires careful consideration of language features, particularly when dealing with compound characters. Previous research in this domain has been limited. Therefore, we have devised an algorithm tailored to segment overlapping characters by modifying projection profiles. Traditional handwritten text detection methods using machine learning struggle with the intricacies of language structure, resulting in lower recognition rates. To address this challenge, we leverage deep learning models for feature extraction, leading to significantly improved recognition rates.

III. IMPLEMENTATION PROCESS

3.1 Recognition of words in an image

The test page which must be checked for word recognition rate is given as input to OCR. The bilateral filter is utilized to smooth images while safeguarding edge details [18] specifically crafted for grayscale images. This non-linear filter performs a mathematical operation that involves a weighted average of pixel values within a given neighbourhood. These weights are determined by both spatial proximity and intensity similarity, creating a harmonious balance between spatial closeness and similarity in pixel intensity. The bilateral filtering is mathematically given by (1)

$$BF(I, \sigma_{sp}, \sigma_{in}) = \frac{1}{W_r} \sum_{s \epsilon neighborhood(r)} I(s) \cdot w(r,s) \quad (1)$$

where I is an image in grey scale with coordinates of pixels at (r, s). σ_{sp} and σ_{in} regulates special dimensions of filter and similarity intensity. $w(r,s)$ is kernel of bilateral filter given by (2),

$$w(r,s) = w_s(r,s) \cdot w_i(I(r), I(s)) \quad (2)$$

where $w_s(r,s)$ and $w_i(I(r), I(s))$ represent the spatial and intensity components of the weight, respectively.
The adaptive Gaussian thresholding technique employs a threshold function that considers variations in illumination to detect page edges. This approach enables effective identification of the edges of the pages by considering changes in illumination. The process includes computing mean, $\mu(r,s)$ and standard deviation $\sigma(r,s)$ in neighbourhood of pixel intensities at *(r, s)* of image I given by (3) and (4).

$$\mu(r,s) = \frac{1}{WS} \sum_{i=r-w}^{r+w} \sum_{j=s-w}^{s+w} I(i,j) \quad (3)$$

$$\sigma(r,s) = \sqrt{\frac{1}{WS} \sum_{i=r-w}^{r+w} \sum_{j=s-w}^{s+w} (I(i,j) - \mu(r,s))^2} \quad (4)$$

where *WS* is window size and *w* are the half window size.
The threshold of each pixel is computed by (5)

$$T(r,s) = \mu(r,s) - l \cdot \sigma(r,s) \quad (5)$$

where l controls variations in sensitivity and user specified value. Classify the pixels by using thresholding as given by (6)

$$Binary\ result(r,s) = \begin{cases} 1, & if\ I(r,s) > T(r,s) \\ 0, & otherwise \end{cases} \quad (6)$$

This flexible method empowers the threshold to adjust throughout various sections of the image, adeptly handling variations in illumination and refining the accuracy of edge detection.
To identify words on the page, eliminate page noise by employing the Sobel operator, which utilizes 5x5 Gaussian filtering in the process given by (7)

$$g(y,z) = G_\sigma(y,z) * f(y,z) \quad (7)$$

where $G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{y^2+z^2}{2\sigma^2}\right)$

The pixel directions and gradients of the edges can be calculated by (8)

$$G(y,z) = \sqrt{((g_y^2(y,z) + g_z^2(y,z))}$$

$$\theta(y,z) = tan^{-1}\left(\frac{g_y(y,z)}{g_{z(y,z)}}\right) \quad (8)$$

where $g_y(y,z)$ and $g_z(y,z)$ represents horizontal and vertical derivatives. Afterward, contours are produced, resulting in numerous bounding boxes that overlap. Union and intersection techniques are then employed to obtain a bounding box around each word, and the count of these bounding boxes determines the total number of words on the specific page.

3.2 Character segmentation and Feature extraction

Character segmentation is pivotal during the preprocessing stage of OCR, significantly influencing the overall accuracy and efficiency of the recognition process [19]. Handwritten Telugu text typically features spaces between characters. While techniques such as vertical projection profiles can effectively segment words into characters when there is no overlapping. But handwriting speed or individual writing styles often lead to character overlap. To address this issue, we have devised a segmentation algorithm capable of accurately isolating overlapping characters. This algorithm preserves crucial character features, ensuring that the model can recognize the characters with precision.

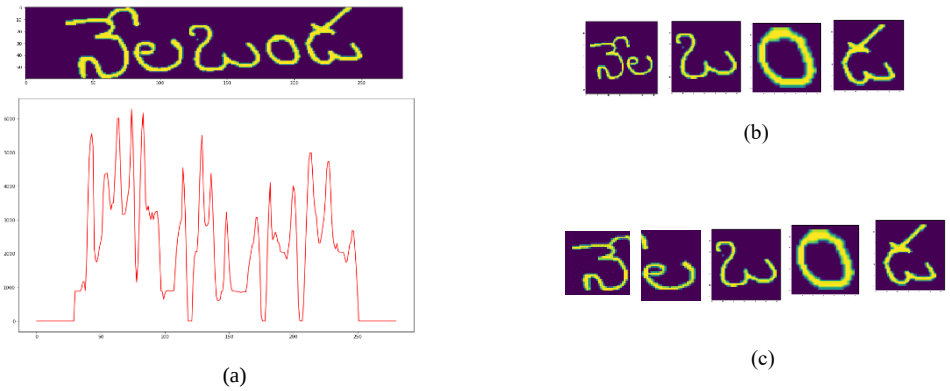
1. Input the bounding box. Compute the image width as well as height.
2. Consider bounding box as image denoted by $I(z,y)$.For every column z in the image, compute the vertical projection profile P(z) by summing the pixel intensities along the vertical axis given by (9)

$$P(z) = \sum_{z=1}^{H} I(z,y) \qquad (9)$$

where H represents the height of the image.

3. Let $P(0), P(1), P(2), \ldots\ldots$ indicate the sum of pixel intensities along the height of image column wise. Find the values at which $P(z) = 0$. The points whose P(z)=0 represents gaps between characters. So, split them at those point.
4. The above process splits the words to characters if there is proper spacing between them. If there are overlapping characters, then $P(z) \neq 0$ and this algorithm will not separate them. So, after splitting the bounding box to segments calculate the width of each segment. Let $\{WD_1, WD_2, \ldots WD_n\}$ represents width of each segment. The value of n depends on the segments in each bounding box.
5. Let WD_m represents the width of small segment. If $WD_n >= 2xWD_m$ then compute $P(z) = \sum_{i=-10}^{10} \sum_{z=1}^{H} I(z,y)$. Then split the image where P(z) is minimum. This algorithm effectively splits the overlapping characters in Telugu language.

Fig. 1 shows segmentation outputs for a word. Fig 1 (a) shows the input word and its sum of vertical pixel intensity plot. Fig. 1 (b) shows the segmentation based on the plot, where the overlapping characters are not segmented. Fig. 1 (c) shows the properly segmented character after passing through our algorithm.



(a) (b) (c)

Fig 1. Word segmented to character (a) Projection graph for word (b) Words segmented by using projections (c) Segmentation output of proposed character segmentation algorithm

Feature extraction plays a crucial role in OCR, with CNNs serving as potent tools for extracting features from images. Let I denote the input image, K represent the applied filter, and F be the resulting output feature map. The convolution operation at a specific spatial location (g,h) in the output feature map is defined by (10)

$$F(g,h) = \sum_{r=0}^{R-1} \sum_{s=0}^{S-1} I(g+r, s+h).K(r,s) \qquad (10)$$

where *r,s* are filter dimensions. $I(g+r, s+h)$ are pixel intensities of input image at locations $(g+r, s+h)$. $K(r,s)$ represents weights of filter at location (r,s). Summation is carried out across all elements of the filter, and the outcome is allocated to the corresponding position (g,h) in the output feature map. This procedure is reiterated for all spatial locations within the output feature map.

Activation functions are crucial in empowering neural networks to grasp intricate, non-linear associations, thereby promoting successful training and ensuring the model's adaptability across diverse tasks. In CNNs ReLU is the most popularly used one. If *Z* denotes the input to the activation function, potentially originating from the output of a convolutional or fully connected layer before activation, the ReLU activation *V* is calculated elementwise as given by (11),

$$V(g,h,k) = \max(0, Z(g,h,k)) \tag{11}$$

where $V(g,h,k)$ is value of activation function at position (g,h) in the kth feature map and $Z(g,h,k)$ is the corresponding input value to the activation function.

Pooling serves to decrease the spatial dimensions of the input volume while preserving crucial information. The max-pooling operation can be mathematically expressed at location *(g,h)* given by (12)

$$M(g,h,t) = max_{m=0}^{t-1} max_{n=0}^{n-1} I(R \cdot g + m, R \cdot h + n) \tag{12}$$

where *t* is pool window size, *R* is stride, *I* is input feature map and *M* is output feature map.

After a series of convolution and pooling operations, flattening is often applied before feeding the output of convolutional or pooling layers into a fully connected layer. The bottom segment of the CNN involves a SoftMax layer, producing class probabilities utilized in classification by fully connected layers. This can be expressed by (13)

$$R_t = \frac{e^{s_t}}{\sum_{p=1}^{n} e^{s_t}} \tag{13}$$

Where R_t represents output for class *t* and *n* is total number of classes.

The cross-entropy loss computed between the predicted probabilities R_t and actual probabilties P_t for each class is given by (14)

$$L(P,R) = -\sum_{q=1}^{n} P_t \cdot \log(R_t) \tag{14}$$

where *L* is the entropy loss.

The SoftMax activation transforms raw scores into probabilities, and the cross-entropy loss measures the disparity between these predicted probabilities and the actual probabilities. This guides the optimization process throughout training, aiming to minimize this difference for improved model performance. The Adam optimizer adapts the network's weights and biases to minimize a specified loss function. The update rules for the weights and biases using the Adam optimizer are outlined below in (15), (16), (17)

$$u_y = \alpha_1 \cdot u_{y-1} + (1-\alpha_1) \cdot \nabla_\theta LP(\theta) \tag{15}$$

$$p_y = \alpha_2 \cdot p_{y-1} + (1-\alpha_2) \cdot (\nabla_\theta LP(\theta))^2 \tag{16}$$

$$\theta_y = \theta_{y-1} - \frac{\eta}{\sqrt{p_y}+\varepsilon} \cdot u_y \tag{17}$$

where $LP(\theta)$ loss function, η is learning rate, $\nabla_\theta LP(\theta)$ is gradient loss u_y, p_y are the gradients moving averages and their squares. α_1, α_2 are decay rates and ε value is a constant to avoid division by zero. The Adam optimizer adjusts the model parameters by considering the gradients of the cross-entropy loss. This process enhances training effectiveness and optimizes the model's parameters for improved performance.

Following segmentation of words into characters, the resulting test page is inputted into the trained model for recognition. This test serves to compute the system's word recognition rates, considering that the model is exclusively trained with characters and character recognition rates are obtained from the generated plots.

We have employed five CNN models and assessed their performance on characters using a Telugu dataset. The models include ResNet 34, ResNet 50, ResNet 101, and Inception-ResNet, Inception V1 [20]-[22]. These are shown in Fig. 2, Fig 3, Fig.4, Fig.5, Fig.6.

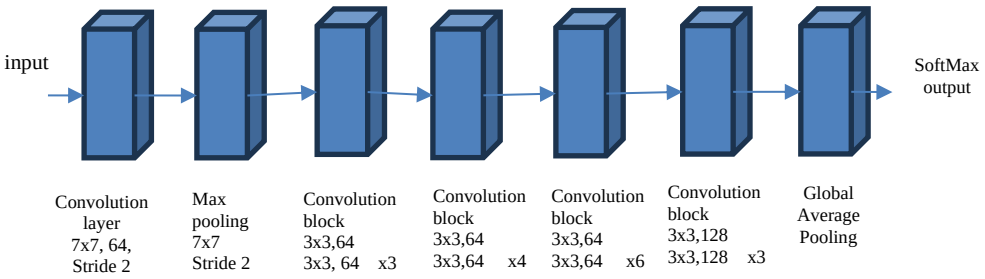


Fig. 2 Architecture of ResNet34

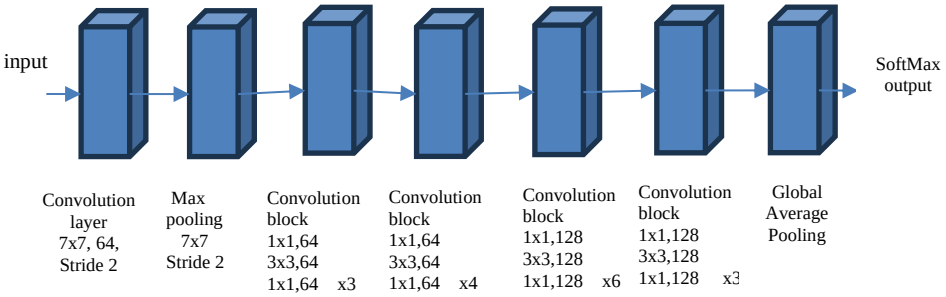


Fig. 3 Architecture of ResNet50

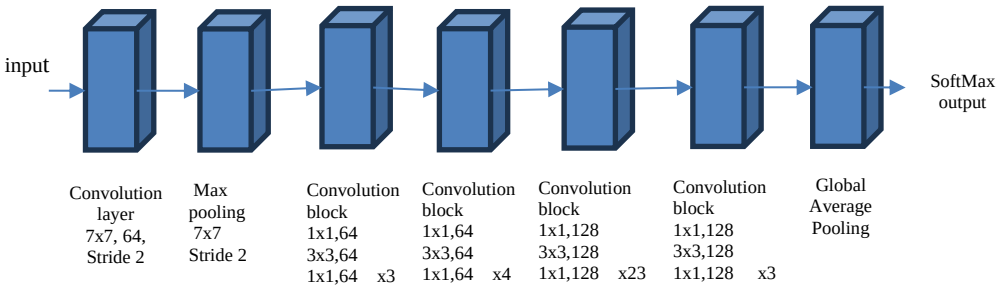


Fig. 4 Architecture of ResNet101

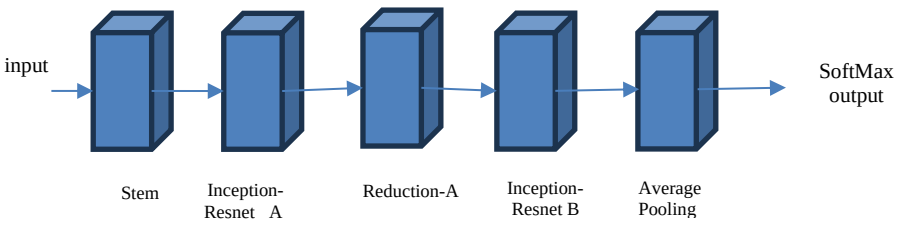


Fig. 5 Architecture of ResNet-Inception

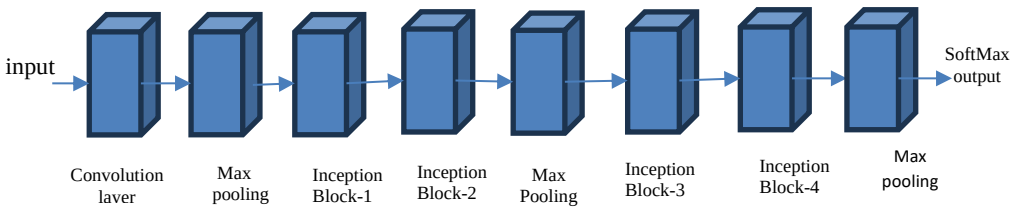


Fig. 6 Architecture of Inception V1

Inception modules excel at extracting multi-scale features by integrating a variety of filter sizes within a single layer. This method accurately approximates a diverse range of filter sizes, enabling the neural network to effectively capture both intricate and large-scale features. ResNet addresses the challenge of vanishing gradients in deep networks by employing strategic residual connections. These connections optimize the training of extremely deep networks by facilitating more direct gradient flow, allowing the model to learn complex features. Importantly, ResNet's proficiency in mastering identity mappings further amplifies the training of exceptionally deep neural networks.

ResNet34 is comprised of 34 layers, making it faster to train and demanding fewer computational resources than deeper models. It is well-suited for tasks involving fewer complex datasets or scenarios with limited computational resources. With 50 layers, ResNet50 achieves greater depth than ResNet34. It exhibits improved performance on complex datasets compared to its shallower counterpart. ResNet50 is a popular choice for various computer vision tasks, striking a balance between model depth and computational efficiency. Comprising 101 layers, ResNet101 attains greater depth than both ResNet34 and ResNet50. It excels in performance on highly complex datasets and tasks demanding a more intricate feature representation. However, due to its increased depth, ResNet101 typically necessitates more computational resources and time for training.

Integrating Inception and ResNet enhances the network's ability to comprehend mixed details and multi-scale features concurrently. Tasks involving patten boundary definition rely on precise details, while recognizing patterns across diverse sizes and orientations demands multi-scale capabilities. By capitalizing on the strengths of both architectures, the combined model surpasses the performance of using Inception or ResNet in isolation.

These distinct architectures serve as a regularization technique, mitigating overfitting by diversifying the acquired features. Integrating these architectures may extend training time due to the heightened complexity of the model. Furthermore, in complex architectures, the meticulous selection of hyperparameters becomes paramount.

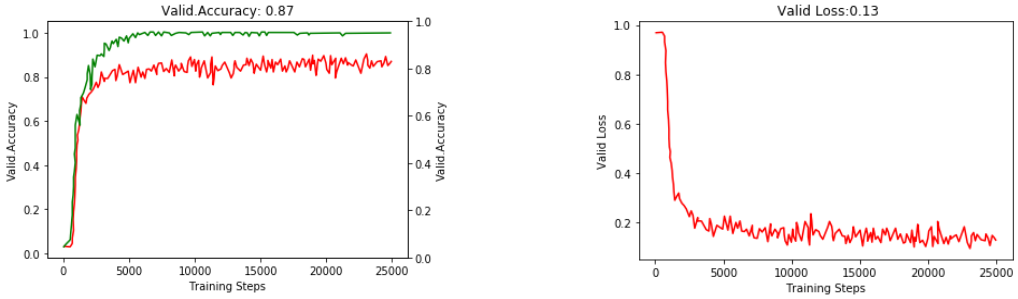
3.3 Dictionary model Development

The utilization of dictionary models improves the system's recognition rates. Telugu, in particular, poses challenges due to similar characters, leading individuals to unintentionally write one character instead of another. Instances include యు and ము, స and న, ధ and ధ, ఠ and ర, ప and వ, న and స. In these scenarios, dictionary models rectify characters within words, addressing both human and machine errors. As a result, we observe improved word detection rates.

Step 1: Let the predicted word be denoted as input and compute its length.
Step 2: Examine the dictionary words with lengths equal to or greater than the predicted word.
Step 3: For each word selected from the dictionary, determine the count of matching characters between it and the predicted word.
Step 4: Replace the predicted word with the dictionary word that exhibits the maximum character matching.

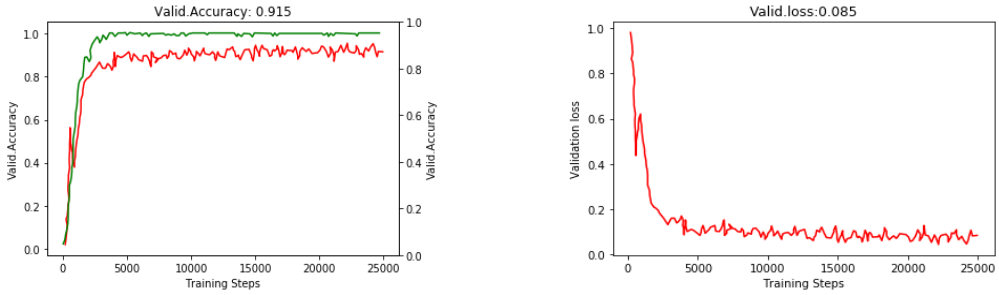
IV. Results and Discussions

Character training for models involves utilizing the Telugu dataset obtained from IEEE Dataport [23]. The training set comprises 11,602 images, while an additional 2,565 images are set aside for validation. The evaluation of OCR effectiveness is performed on the Telugu IEEE Dataport handwritten dataset, employing Inception V1, ResNet 34, ResNet 50, and ResNet 101 and Inception-ResNet. The models undergo training for 25,000 steps. Fig. 7 shows the plots of Inception V1. Fig. 7 (a) displays the accuracy plot for Inception V1, while Fig. 7(b) illustrates the corresponding loss plot. Inception V1 has attained an accuracy of 87% with a validation loss of 13%. The performance of ResNet 34 is visually depicted in Fig. 8. Fig. 8 (a) presents both train and validation accuracy results, while Fig. 8 (b) provides insights into the loss plot. Notably, ResNet 34 achieves a remarkable validation accuracy of 91.5%, with a corresponding validation loss of 8.5%.



(a) (b)

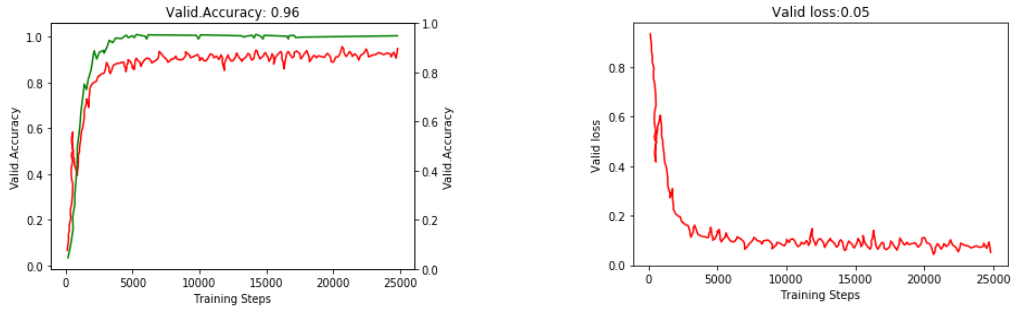
Fig. 7 Performance of Inception V1 (a) Accuracy plot (b) Loss plot



(a) (b)

Fig. 8 Performance of ResNet34 (a) Accuracy plot (b) Loss plot

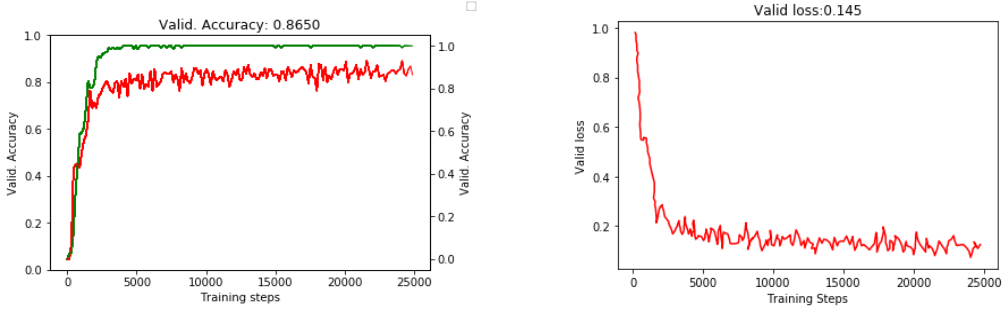
Fig. 9 presents performance plots of ResNet 50. Fig. 9 (a) presents the train and validation accuracy plots of ResNet 50, while Fig. 9(b) illustrates the validation loss of ResNet 50. The model attains a valid accuracy of 96% and loss of 5%.



(a) (b)

Fig. 9 Performance of ResNet 50 (a) Accuracy plot (b) Loss plot

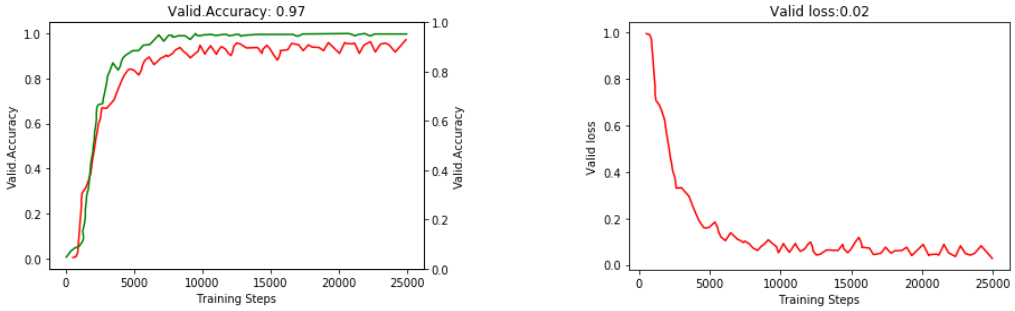
Fig. 10 shows the plots of ResNet 101. Fig. 10(a) illustrates the training and validation accuracy outcomes for ResNet 101. Fig. 10 (b) offers a visual representation of the validation loss. Notably, ResNet 101 attains a validation accuracy of 86.5%, accompanied by a loss of 14.5%. The model's performance on the Telugu dataset highlights overfitting issues within the deep CNN architecture.



(a) (b)

Fig. 10 Performance of ResNet 101 (a) Accuracy plot (b) Loss plot

Fig. 11 shows the plots of ResNet-Inception. The training and validation accuracy outcomes for Inception-ResNet are depicted in Fig 11(a) while Fig. 11(b) visually presents the validation loss. Notably, Inception-ResNet achieves a commendable validation accuracy of 97%, coupled with a validation loss of 2%. The model exhibits top-notch performance on the Telugu dataset, effectively avoiding overfitting issues.



(a) (b)

Fig. 11 Performance of ResNet- Inception (a) Accuracy plot (b) Loss plot

Fig. 12 illustrates a comparison of validation loss among Inception V1, ResNet34, ResNet50, ResNet101, and Inception-ResNet models. A lower validation loss indicates better model performance. Consequently, the graph distinctly indicates that the Inception-ResNet model exhibits significantly lower loss compared to the other models.

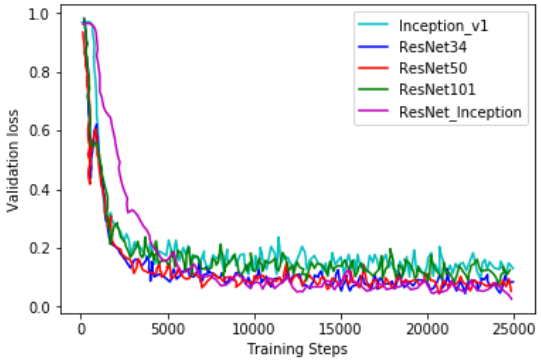


Fig.12 Comparison of loss plots

All the ResNet models underwent training with Telugu characters. The input for the models consists of scanned images, wherein the page boundaries are identified, excluding the background and correcting image orientations. Following this, words are detected and segmented into individual characters. The CNN models then recognize these characters, and the words are reconstructed. In character recognition, the model may misclassify characters due to their similarities and variations in handwritten styles, leading to a potential decrease in the word recognition rate. To address this, the integration of a dictionary is implemented to rectify character misclassifications, resulting in an improvement in word accuracy rates.

Our evaluation involved 1000 sentences handwritten by various individuals, and we computed the word recognition rates for the three models both with and without the use of a dictionary. Table 1 presents details on the total parameters, character recognition rates, and word recognition rates for different architectures. The word recognition rates are depicted with and without the application of dictionaries. It is evident from the table that models utilizing dictionaries enhance the accuracy rate and mitigate issues arising from similarly structured characters.

Table 1. Comparison of parameters and accuracy rates of the models

Model	Parameters	Valid Character Accuracy	Word Accuracy	Word Accuracy with dictionary
Inception V1	56,31,016	87%	76%	78%
ResNet 34	13,32,480	91.5%	80%	81.5%

ResNet-50	21,01,312	96%	84%	86.5%
ResNet-101	50,07,552	86.5%	73%	75%
ResNet Inception Cascade	55,43,120	97%	85%	87.5%

From Table 1, Inception V1 is equipped with 56,31,016 trainable parameters and demonstrates a character recognition rate of 87%, along with a word recognition rate of 76% without a dictionary and 78% with a dictionary. ResNet 34, featuring 13,32,480 trainable parameters, achieves a character accuracy rate of 91.5% and a word accuracy rate of 80% without a dictionary, improving to 81.5% with the use of a dictionary. ResNet 50, comprising 21,01,312 trainable parameters, attains a character accuracy rate of 96%. Additionally, it achieves a word accuracy rate of 84% without a dictionary and 86.5% with a dictionary. ResNet 101, with 50,07,552 trainable parameters, achieves a character accuracy rate of 86.5%. Furthermore, it attains a word accuracy rate of 73% without a dictionary, which increases to 75% with the aid of a dictionary. Inception-ResNet, featuring 55,43,120 trainable parameters, excels with a character accuracy rate of 97% and a word accuracy rate of 85% without a dictionary, surging to 87.5% with the inclusion of a dictionary.

Out of the five models, Inception-ResNet performs better on Telugu language. ResNet 50 performance is almost nearer to the Inception-ResNet and it has less trainable parameters compared to ResNet 50. ResNet 34 with fewer parameters achieved a significant recognition rate.

We have considered a test set and evaluated the performances of all ResNet models. The valid accuracy and test accuracy of different ResNet models is shown in Table 2. On both the validation set and test set ResNet-Inception model performed better. The graphical representations of validation and test accuracy are shown in Fig. 13.

Table 2. Validation and test accuracy rates of the ResNet models.

Model	Valid Character Accuracy	Test Character Accuracy
ResNet 34	91.5	88.011
ResNet-50	96	88.39
ResNet-101	86.5	81.82
ResNet Inception	97	90.29

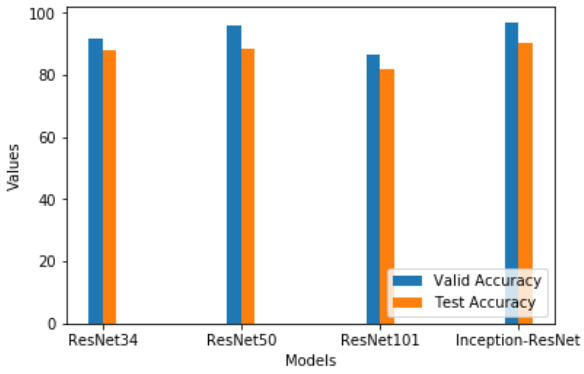


Fig. 13 Validation and Test accuracy rates of ResNet models

The confusion matrix holds a central position in machine learning and classification tasks, providing a crucial instrument to evaluate the effectiveness of a classification model. Confusion matrix is computed for all the models and the parameters like Precision, recall and F1 score calculated and tabulated in Table 3. The comparison of these parameters is shown in Fig. 14.

Table 3. Confusion matrix parameters for the ResNet models.

Model	Test Accuracy	Precision	Recall	F1 Score
ResNet34	88.011	0.87782	0.864606	0.867587
ResNet50	88.39	0.8782	0.861862	0.8601
ResNet101	81.82	0.830228	0.806937	0.798304

ResNet- Inception	90.29	0.904412	0.890236	0.886741

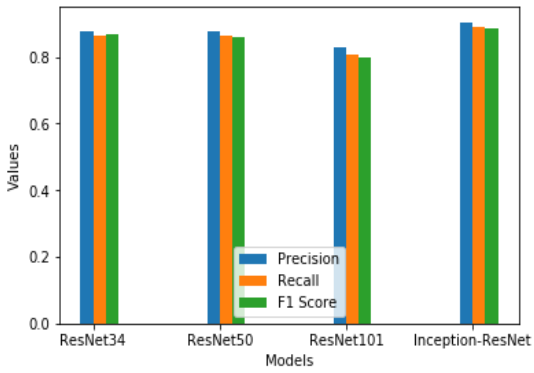


Fig. 14 Comparison of confusion parameters for ResNet models.

Table 4 illustrates the progress in OCR for native languages such as Telugu [12], [24], -[25], Tamil [26]-[27] through various techniques, all achieving notable recognition rates. Particularly in Telugu, the proposed method surpasses other approaches by achieving character recognition rate of 97% and word recognition rate of 87.5%.

Table 4. OCR accuracy rates for native languages

Name of the Author	Type of text	Technique	Accuracy
Panyam Narahari	Handwritten Telugu Characters	Zoning method- Nearest Neighbour	78%
Devarapalli Koteswara Rao	Handwritten Telugu Text	Hidden Markov Models (HMMs), Akshara Models & Akshara Bigram Language models	74%
Minesh Mathew	Scene Telugu Text detection	Hybrid CNN-RNN (CRR)	86.2%
		Hybrid CNN-RNN (WRR)	57.2%
,Noushath Shafi	Handwritten Tamil characters	CNN with 2 Convolutional layers, 2 max pooling layers and 2 fully connected layers	88%
S. Kowsalya	Handwritten Tamil characters	Optimal neural network with Weight optimization using elephant herding optimization	93%
Proposed	Handwritten Telugu Text	CNN, ResNet-Inception (CRR)	97%
		CNN, ResNet-Inception (WRR)	87.5%

V. CONCLUSION

Advancements in OCR for regional languages are frequently impeded by resource constraints. Overcoming this hurdle involves implementing robust preprocessing techniques to eliminate noise and rectify page skew, resulting in a significant enhancement of input quality. The accuracy of systems handling handwritten texts relies heavily on effective character segmentation, particularly when faced with overlapping characters. The adoption of a segmentation approach that accounts for language-specific features can substantially increase word recognition rates. The suggested segmentation algorithm has demonstrated its proficiency in preserving crucial character features while accurately dissecting words into individual characters, even when confronted with overlapping characters for Telugu text.

In the realm of Telugu language recognition, Inception-ResNet stands out as the premier performer. ResNet 50 closely trails in performance, approaching the capabilities of Inception-ResNet with the added advantage of fewer trainable parameters. ResNet 34, boasting a reduced parameter count, achieves a commendable recognition rate. However, Inception, characterized by a broader network and more trainable parameters, faces challenges in handling complex structured characters and unbalanced classes. Despite these challenges, its performance remains competitive compared to other ResNet models.

ResNet-101 excels at extracting sophisticated features from data. Nevertheless, when grappling with limited datasets, these models encounter difficulties in extracting relevant features. The inadequacy of data points hampers

the network's ability to recognize significant patterns, impacting overall performance. So ResNet models excel in recognition of handwritten Telugu text, when the model selection must be based on the dataset used.

REFERENCES

[1] Munish Kumar, M. K. Jindal, R. K. Sharma, "Review on OCR for Handwritten Indian Scripts Character Recognition," Communications in Computer and Information Science, pp.268–276, 2011. doi:10.1007/978-3-642-24055-3_28.

[2] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 8, pp.787–808, 1990. doi:10.1109/34.57669.

[3] Fujisawa, H., Nakano, Y., &Kurino K, "Segmentation methods for character recognition: from segmentation to document structure analysis," Proceedings of the IEEE, vol. 80, no. 7, pp.1079–1092, 1992. doi:10.1109/5.156471

[4] Ryu, H. I. Koo, and N. I. Cho, "Word segmentation method for handwrit-ten documents based on structured learning," IEEE Signal Process. Letters., vol. 22, no. 8, pp.1161–1165, 2015. doi:10.1109/LSP.2015.2389852.

[5] Hadar I. Avi-Itzhak, Thanh A. Diep, and Harry Garland, "High accuracy optical character recognition using neural networks with centroid dithering," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no. 2, pp.218–224, 1995. doi:10.1109/34.368165.

[6] R. Ptucha, F. Petroski Such, S. Pillai, F. Brockler, V. Singh, and P. Hutkowski, "Intelligent character recognition using fully convolutional neural networks," Pattern Recognition, vol. 88, pp.604–613, 2019.doi:10.1016/j.patcog.2018.12.017.

[7] U. Garain, B.B. Chaudhuri, "Segmentation of touching characters in printed Devnagari and Bangla scripts using fuzzy multifactorial analysis," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Review), vol. 32, no. 4, pp. 449 – 459, 2002. doi: 10.1109/TSMCC.2002.807272.

[8] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis, "Text line and word segmentation of handwritten documents," Pattern Recognition, vol. 42, no. 12, pp.3169–3183, 2009. doi:10.1016/j.patcog.2008.12.016.

[9] Jewoong Ryu, Hyung Il Koo, & Nam Ik Cho., "Word Segmentation Method for Handwritten Documents based on Structured Learning," IEEE Signal Processing Letters, pp.1161–1165, 2015. doi:10.1109/lsp.2015.2389852.

[10] Vishal Rajput, N. Jayanthi, S. Indu, "An efficient character recognition algorithm for connected Handwritten Documents" In book: Document Analysis and Recognition, pp.97-105, 2019. doi:10.1007/978-981-13-9361-7_9.

[11] Haifeng Zhao, Yong Hu, Jinxia Zhang, "Reading Text in Natural Scene Images via Deep Neural Networks," 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR), pp.43-48, 2017. doi:10.1109/acpr.2017.25.

[12] Minesh Mathew, Mohit Jain, C.V. Jawahar, "Benchmarking Scene Text Recognition in Devanagari, Telugu and Malayalam," 14th IAPR International Conference on Document Analysis and Recognition, vol.9, pp.9-15, 2017.doi: 10.1109/ICDAR.2017.364.

[13] K. Chandra Prakash, Y. M. Srikar, G. Trishal, S. Mandal and S. S. Channappayya, "Optical Character Recognition (OCR) for Telugu: Database, Algorithm and Application," 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, pp. 3963-3967, 2018. doi: 10.1109/ICIP.2018.8451438.

[14] S. Y. Arafat and M. J. Iqbal, "Urdu-Text Detection and Recognition in Natural Scene Images Using Deep Learning," IEEE Access, vol. 8, pp. 96787-96803, 2020. doi: 10.1109/ACCESS.2020.2994214.

[15] Y Zou, Y Zhang, J Yan, X Jiang, T Huang, H Fan, Z & Cui," A Robust License Plate Recognition Model Based on Bi-LSTM," IEEE Access, vol. 8, pp.211630-211641, 2020. doi: 10.1109/ACCESS.2020.3040238.

[16] A. Rasheed, N. Ali, B. Zafar, A. Shabbir, M. Sajid and M. T. Mahmood, "Handwritten Urdu Characters and Digits Recognition Using Transfer Learning and Augmentation with AlexNet," in IEEE Access, vol.10, pp.102629-102645, 2022. doi: 10.1109/ACCESS.2022.3208959.

[17] R. Malhotra and M. T. Addis, "End-to-End Historical Handwritten Ethiopic Text Recognition Using Deep Learning," IEEE Access, vol. 11, pp. 99535-99545, 2023. doi: 10.1109/ACCESS.2023.3314334.

[18] Paris, S., Kornprobst, P., Tumblin, J., & Durand, F., "Bilateral Filtering: Theory and Applications," Foundations and Trends in Computer Graphics and Vision. 2008; vol. 4, no .1, pp.1–74. doi:10.1561/0600000020

[19] Himadri Nandini Das Bebartta and Sanghamitra Mohanty, "Algorithm for segmenting script-dependant portion in a bilingual Optical Character Recognition system," Pattern Recognition and Image Analysis, vol. 27, no. 3, pp.560–568, 2017. doi:10.1134/s1054661817030142.

[20] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.770-778, 2016. doi: 10.1109/CVPR.2016.90.

[21] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke. Alex Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), pp. 4278-4284, 2017. arXiv:1602.07261.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, "Going Deeper with Convolutions," 2014. arXiv:1409.4842.

[23] Muni Sekhar Velpuru, Tejasree G, Ravi Kumar M, "Telugu Handwritten Character Dataset," Dec 30,2020. IEEE Dataport. https://Dx.Doi.Org/10.21227/Mw6a-D662.

[24] P. N. Sastry, T. R. V. Lakshmi, N. V. K. Rao, T. V. Rajinikanth and A. Wahab, "Telugu Handwritten Character Recognition Using Zoning Features," 2014 International Conference on IT Convergence and Security (ICITCS), pp.1-4, 2014. doi: 10.1109/ICITCS.2014.7021817.

[25] Devarapalli Koteswara Rao, Atul Negi, "Orthographic Properties Based Telugu Text Recognition Using Hidden Markov Models," 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol.9, pp. 9-15, 2017. doi: 10.1109/ICDAR.2017.327.

[26] N. Shaffi and F. Hajamohideen, “uTHCD: A New Benchmarking for Tamil Handwritten OCR,” IEEE Access, vol. 9, pp.101469-101493, 2021. doi: 10.1109/ACCESS.2021.3096823.

[27] Kowsalya, S., & Periasamy, P. S, “Recognition of Tamil handwritten character using modified neural network with aid of elephant herding optimization,” Multimedia Tools and Applications, pp.25043-25061, 2019. doi:10.1007/s11042-019-7624-2.