International Journal of Advances in Applied Sciences (IJAAS)
Vol. 13, No. 3, September 2024, pp. 487~496
ISSN: 2252-8814, DOI: 10.11591/ijaas.v13.i3.pp487-496

Optical character recognition for Telugu handwritten text using SqueezeNet convolutional neural networks model

Buddaraju Revathi[1], B N V Narasimha Raju[2], Ajay Dilip Kumar Marapatla[3],
Kagitha Veeramanikanta[1], Katta Dinesh[1], Maddirala Supraja[1]
[1]Department of Electronics and Communication Engineering, Sagi Rama Krishnam Raju Engineering College, Bhimavaram, India
[2]Department of Computer Science and Engineering, Sagi Rama Krishnam Raju Engineering College, Bhimavaram, India
[3]Department of CSE-AIML, and IoT, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India

Article Info

Article history:

Received Jan 5, 2024
Revised Mar 19, 2024
Accepted Apr 24, 2024

Keywords:

Feature extraction
Optical character recognition
Segmentation
SqueezeNet
Telugu handwritten text

ABSTRACT

Optical character recognition (OCR) is a process that recognizes and converts data from scanned images, including both handwritten and printed documents, into an accessible format. The challenges in Telugu OCR arise from compound characters, an extensive character set, limited datasets, character similarities, and difficulties in segmenting overlapping characters. To tackle these segmentation complexities, an algorithm has been developed, prioritizing the preservation of essential features during character segmentation. For distinguishing between structurally similar characters, we used convolutional neural networks (CNN) due to their feature-extracting properties. We have employed the CNN model, the SqueezeNet for feature extraction, resulting in an impressive character recognition rate of 94% and a word recognition rate of 80%.

Corresponding Author:

Kagitha Veeramanikanta
Department of Electronics and Communication Engineering
Sagi Rama Krishnam Raju Engineering College
Bhimavaram, West Godavari Dt, Andhra Pradesh-534204, India
Email: veeramanikanta2002@gmail.com

1. INTRODUCTION

Optical character recognition (OCR) is a procedure that transforms printed or handwritten text into data that can be accessible by machines [1]. It enables the digitization of documents by recognizing and extracting text, making it accessible for electronic processing and storage. The OCR technology effectively addresses drawbacks associated with native languages. Its capacity for digitization ensures that native language scripts are seamlessly integrated into the digital landscape, overcoming limitations of accessibility. By enabling efficient text search and retrieval, OCR contributes to improved data management and utilization. Standardization of native language representation in digital formats is achieved through OCR, fostering compatibility across diverse platforms [2].

The fourth most spoken language in India is Telugu. Telugu handwritten text recognition poses significant challenges owing to the script's intricate nature. These challenges stem from the script's complexity, which encompasses similar characters, a vast character set, including compound and overlapping characters, as well as issues with trainability. To deal with the segmentation problems Sahare and Dhok [3] introduced a resilient algorithm for the segmentation and recognition of characters in multilingual Indian document images, covering Devanagari and Latin scripts. The segmentation algorithm leverages structural properties and graph distance theory, validated through a support vector machine (SVM) classifier. Character recognition is enhanced by the computation of three novel geometrical shape-based features, resulting in outstanding performance with segmentation and recognition rates reaching 98.86% and 99.84%, respectively. Sharma and Dhaka [4] employed the pixel plot and trace and re-plot and re-trace method to segment offline English handwritten cursive scripts, prioritizing word, and character segmentation. This procedure involves skew and de-skew operations, enabling word segmentation on basing pixel space and subsequent character segmentation. The proposed approach, tailored to overcome various challenges, showcases effectiveness in extracting characters from English handwritten cursive scripts in offline settings. Rehman and Saba [5] evaluated a novel character segmentation algorithm, comparing its performance with existing methods, emphasizing accuracy and computational efficiency. Trained on identity access management (IAM) and tested on CEDAR benchmark databases, the algorithm focuses on cursive word segmentation through geometric features and ligature analysis. Employing a heuristic over-segmentation technique and a straightforward criterion for refining segmentation points based on character shape, the results demonstrate improved accuracy in cursive script segmentation with minimal computational complexity. To deal with the feature extraction process Sarkhel *et al.* [6] developed a novel feature-based approach in OCR, focusing on recognizing handwritten characters and numbers in the popular Indic scripts. By utilizing a multi-column multi-scale OCR.

With convolutional neural networks (CNNs) architecture and a multi-scale convolutional sampling technique, the method adeptly extracts robust features from diverse pattern images. With a multi-panel structure and percolated prediction model, the system attains state-of-the-art results across diverse Indic scripts, marking notable progress in the improvement of a comprehensive OCR system for India's multi-script environment. Fu *et al.* [7] have introduced a neural network for improving handwriting recognition in cases of limited user-specific training data. The system incorporates an overall rudimentary classifier, a character recognizer, and a user adaptation module. Through incremental reinforced and anti-reinforced learning, the recognition accuracy has increased from 44.2 to 90.2% over ten adapting cycles when tested on a 600-word Chinese character set. Asma and Zafar [8] utilized the leveraging CNN and long short-term memory network, to overcome the challenges of character identification across different font sizes and trained on ligature thickness graphs and raw images, the study achieved an overall network performance between 90 and 99.8%. Notably, Meta features extracted from ligature thickness graphs exhibit an average accuracy of 98.08%, surpassing the 97.07% achieved with raw images. Research by Wu *et al.* [9] enhances Chinese handwriting recognition through integrated over-segmentation and path search, emphasizing language and character shape models. It employs neural network language models (NNLMs), such as feedforward and recurrent to address the limitations of back-off N-gram LMs. Hybrid recurrent neural network LMs prove particularly effective, achieving a character identification rate and correct rate of 95.88 and 95.95%, respectively. Additionally, the integration of CNN-based models significantly enhances character-level accuracy rates. Saha *et al.* [10] introduced an English handwriting recognition system, utilizing a 40-point feature extraction method for character analysis. Employing a multilayer feed-forward neural network, the system proves effective in recognizing handwritten characters, showcasing its potential for converting handwritten documents into structured text and accurately recognizing handwritten names. Dongre and Mankar [11] introduced a Devnagari digit recognition method, employing statistical difference functions and 17 spatial attributes for numeral representation. This method utilizes 1,500 handwritten numerals for both training and testing, with results showing superior performance from Linear, Quadratic, and Mahalanobis discriminant functions. Dongre and Mankar further proposes a combination classifier, employing majority voting, which proves more effective than individual classifiers [11].

The foremost challenge confronting Telugu language OCR pertains to the intricacies of segmentation due to overlapping characters in handwritten text. Most of the research endeavors primarily concentrated on parsing printed or character-based text, failing to effectively address word-level segmentation issues. In response, we have pioneered a segmentation algorithm finely tuned to dissect words with remarkable precision and preserve the important features of the words thereby substantially enhancing segmentation efficacy. Moreover, navigating the landscape of feature extraction in Telugu OCR presents a formidable complexity. Numerous researchers have delved into diverse models for feature extraction, often grappling with unwieldy parameter sets [12] or declined recognition rates [13]. Recognizing the inherent advantages of CNNs in feature extraction we have strategically embraced the SqueezeNet neural network. Renowned for its slender architecture, SqueezeNet facilitates streamlined feature extraction without compromising accuracy, thus culminating in a notable advancement in the realm of Telugu OCR.

2. OPTICAL CHARACTER RECOGNITION PIPELINE

OCR involves a series of steps to accurately convert unaltered text into accessible text. Initially, the process begins with page detection, where the system identifies and isolates the regions containing text on the scanned or photographed document. Subsequently, word detection comes into play, breaking down the text into individual words. The next critical step is segmentation, where words are further divided into individual characters for precise analysis. Following segmentation, feature extraction occurs, capturing the distinctive characteristics of each character. Finally, the character recognition phase interprets the extracted features and assigns corresponding alphanumeric values, completing the OCR pipeline.

2.1. Page detection

Page detection aims to precisely identify the boundaries of individual pages within a document, allowing subsequent per-page processing and analysis. This entails recognizing image regions corresponding to distinct pages, irrespective of their layout, orientation, or size. Successful page detection is crucial for subsequent tasks like OCR, document categorization, and content extraction. The scanned test image undergoes conversion into a grayscale representation. The median blur operation reduces noise and smooth images. It substitutes the value of each pixel with the median value derived from the surrounding pixels. Let I (a, b) represent the intensity (pixel value) at the coordinates (a, b) in the test image. The median blur operation for a pixel at location (a, b) with a window of size $c \times c$is given in (1).

$$MB(I, a, b, c) = median\left\{I(a', b')\middle|a' \in \left[a - \frac{c}{2}, a + \frac{c}{2}\right], b' \in \left[b - \frac{c}{2}, b + \frac{c}{2}\right]\right\} \quad (1)$$

Where *MB* refers to median blur, median refers to the median function and denotes the range of values from a to b. The window size c is typically an odd number, ensuring a well-defined median value. Later, the Canny edge detection algorithm was employed to identify edges [14]. This operator is known for its ability to detect edges while minimizing the effects of noise by using Gaussian filtering which is given in (2) and the magnitude of the gradient can be obtained using (3).

$$\text{Gaussian filter } G(c, d) = \frac{1}{2\pi\sigma^2} e^{-(c^2+d^2)/2\sigma^2} \quad (2)$$

$$\text{Gradient magnitude}(c, d) = \sqrt{I_c(c, d)^2 + I_d(c, d)^2} \quad (3)$$

Where *(c, d)* is the location of the pixel,σ represents standard deviation, $I(c, d)^2$ represents intensity at (*c*, *d*). Contours delineate the boundaries of objects within an image. Let's denote the contour as *C*, and each point on the contour as given by (4).

$$C = \big(x(m), y(m)\big) \mid a \leq m \leq b \quad (4)$$

Where m is a parameter, a and b define the range of the parameter m. The specific form of *x(m)* and *y(m)* depends on the method used for contour detection. This results in detecting the edges of the page.

2.2. Word detection

Word detection is the work of pinpointing and isolating individual words within an image. In this, we applied the Sobel operator to perform edge detection by leveraging the gradient of image intensity for the test page [15]. This operation highlights areas with significant changes in intensity, indicative of potential edges given by (5) and (6). This technique forms a crucial component of our methodology for word detection within images.

$$I_a(a, b) = (I * G_a) = \sum_{da=-1}^{1} \sum_{db=-1}^{1} I(a + da, b + db) \cdot G_x(da, db) \quad (5)$$

$$I_b(a, b) = (I * G_b) = \sum_{da=-1}^{1} \sum_{db=-1}^{1} I(a + da, b + db) \cdot G_b(da, db) \quad (6)$$

Where G_a and G_b are horizontal and vertical Sobel kernels, *I (a, b)* represent the intensity at coordinates *(a, b).*

Subsequently, a union and intersection mechanism aim at merging boxes that intersect and consolidates those that overlap. This step ensures a cohesive representation of word regions. Bounding boxes are then delineated based on the gradient information, providing accurate localization of potential word areas. The number of bounding boxes is tallied to determine the total count of words on the page.

2.3. Segmentation algorithm

Segmentation entails the division of an image into meaningful sections, guided by shared attributes like color, intensity, or texture [16]. This simplifies image representation for further analysis and interpretation in computer vision applications. The steps involved in segmentation are as follows:

Step 1: Given a test image I, where I_{ij}, represents the pixel intensity at the position (i, j), the segmentation algorithm can be outlined as follows.
Step 2: Computing the column-wise sum of pixel intensities C_i is indicated in (7), which is the histogram of a given word [17].

$$C_i = \sum_{j=1}^{M} I_{ij} \tag{7}$$

Step 3: Identify segments where the same value repeats consecutively across columns as in (8).

$$segments = \{(i, w) | I_{ij} = I_{j(i+1)} \tag{8}$$

Step 4: Determine the minimum segment width from the identified segments.

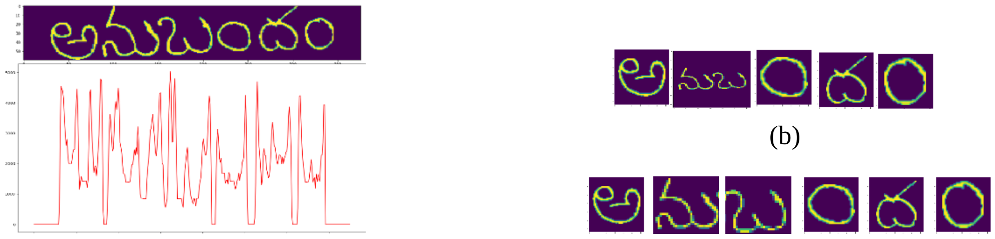
$$C_{min} = min(w) \tag{9}$$

Step 5: Compare C_{min} with widths of previously segmented blocks.

$$w > 2 \times C_{min} \tag{10}$$

Step 6: If any segment width is greater than double of minimum width then split the image at the column with the lowest pixel intensity from the middle, considering 10 columns on either side.
Where,C_i is Column-wise sum, *w* is width of segments, C_{min} is minimum segment width, $segments$ is segmented blocks.

The example segmentation images are shown in Figure 1. Where Figure 1(a) shows the histogram plot of the given word to the segmentation algorithm, Figure 1(b) shows the segmented characters based on spacing, and Figure 1(c) shows the segmented characters after applying the segmentation algorithm.



(a)

(b)

(c)

Figure 1. Segmentation of word in (a) histogram of the given word, (b) separation of characters based on spacing, and (c) segmented characters after applying segmentation algorithm

3. CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE

A CNN is a distinctive neural network tailored for handling tasks associated with data organized in a grid-like structure, such as images. It is composed of multiple layers, each fulfilling a distinct role in the processes of feature extraction and classification. The convolution layer is fundamental in the CNN, which employs the filters that slide over the input data to detect patterns. These filters learn features through weight adjustments during training, producing feature maps that highlight relevant information. The convolution operation is given in (11).

$$M(a, b) = \sum_c \sum_d N(a + c, b + d) \cdot K(c, d) \tag{11}$$

Where, *M (a, b)* represents the value at position *(a, b), K (c, d) is* the filter weight at position *(c, d),* and *N* is the input image.

Following convolution, a pooling layer reduces spatial dimensions, simplifying computation and focusing on essential features. Max pooling, a common technique, retains the maximum value within defined regions of the input. The rectified linear unit (ReLU) activation function brings in non-linearity by producing the input for positive values and zero for negative values [18]. This mitigates the vanishing gradient problem and accelerates training. To optimize the learning process, CNNs often employ the Adam optimizer which is given by (12).

$$\theta_{n+1} = \theta_n - \frac{\eta}{\sqrt{\hat{v}_n + \in}} \cdot \widehat{m}_n \tag{12}$$

Where θ_n is a parameter at time n,η is the learning rate, $\widehat{m}_n$ is biased first-moment estimate, $\hat{v}_n$ is biased raw second-moment estimate, and ∈ is a small constant to avoid zero.

At the final stage, a SoftMax layer is employed for classification tasks. This layer converts raw output scores into probabilities, facilitating multi-class predictions. Combined with the cross-entropy loss function, it guides the network's training process as given by (13).

$$Softmax(z)_a = \frac{e^{z_a}}{\sum_b e^{z_b}} \tag{13}$$

Where, z_a is a-th element of the input vector z, z_b is the b-th element of the input vector$z$, $Softmax(z)_a$ represents a-th element of the output vector after applying SoftMax.

The segmented characters from the text image are fed into a pre-trained model based on the SqueezeNet architecture. This model has been trained using a dataset of characters from IEEE DataPort. The pre-trained model then compares each segmented character with the characters it has been trained on and provides the best output based on this comparison. We have tested 1000 words and obtained the word level accuracy of the model.

SqueezeNet, a compact CNN architecture achieves high accuracy with significantly fewer parameters than traditional models such as visual geometry group (VGG), residual network (ResNet), and inception [19], [20]. Its reduced model size and lower memory footprint make it ideal for resource-constrained devices compared to these larger architectures. SqueezeNet offers faster inference and energy efficiency, crucial for real-time applications and mobile devices. Additionally, it excels in transfer learning tasks, providing a good balance between model complexity and performance. Overall, SqueezeNet’s compact design and efficient operations make it a versatile choice for various computer vision applications when compared to a wide range of other CNN architectures. The architecture of the SqueezeNet neural network is shown in Figure 2.

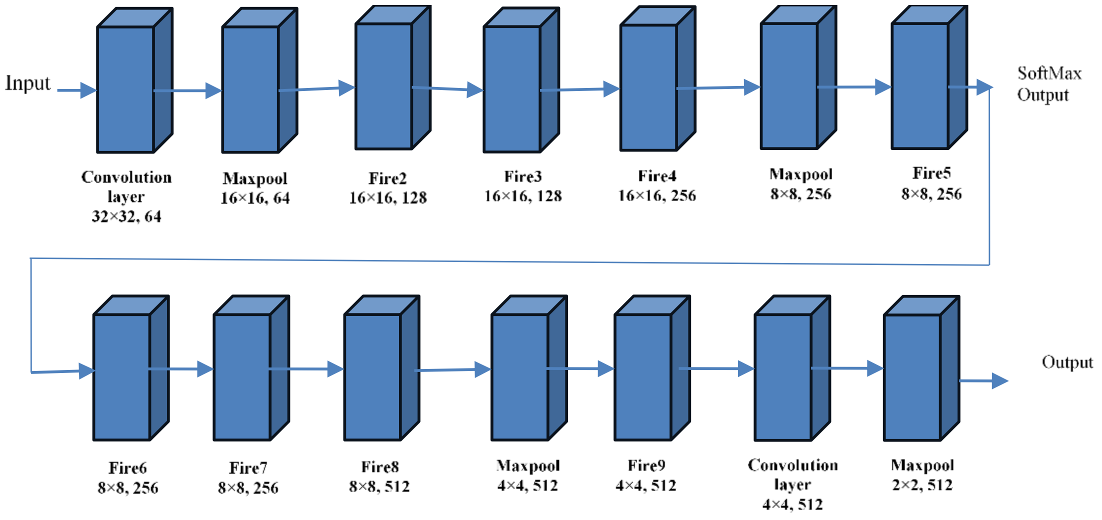
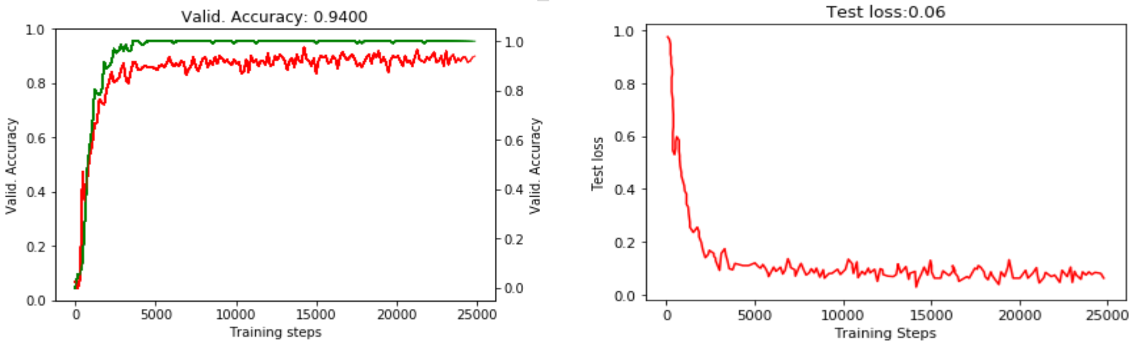


Figure 2. Architecture of SqueezeNet

The core module of the architecture lies in its fire module, a distinctive building block that integrates both convolutional and squeeze layers. The squeeze layer in the fire module compresses information through 1×1 convolutions, effectively reducing the number of channels. Simultaneously, the expand layer in the fire module captures complex features by utilizing a mix of 1×1 and 3×3 convolutions. This efficient design not only accelerates training and inference processes but also positions SqueezeNet as an ideal choice for applications with limited computational resources.

4. RESULTS

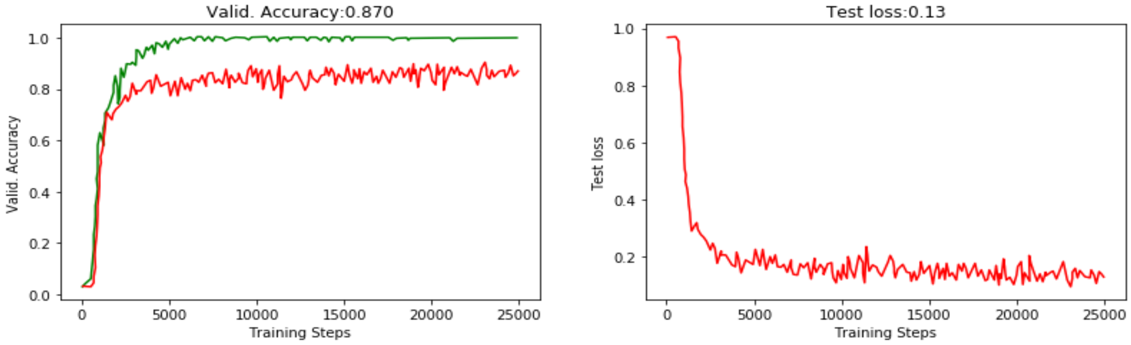
We have conducted training on the character dataset which is taken from the IEEE DataPort [21], utilizing a total of 11,602 images for the training phase. In the subsequent testing phase, a set of 2,565 distinct images is used to evaluate the model's generalization and predictive capabilities. All the images are reshaped to 64x64 for the training phase. The reshaped images are trained for the 25000 training steps for SqueezeNet, VGG19 [22] and ShuffleNet [23]. SqueezeNet is a compact yet powerful neural network architecture that has efficiency in the size of the model and computational resources. Its innovative design incorporates fire modules, enabling high-level feature extraction while maintaining a reduced parameter count, 861344. SqueezeNet excelled in our evaluation, achieving an impressive accuracy of 94% and a loss rate of 6%, demonstrating its suitability for resource-constrained environments. Figure 3 shows the plots of the SqueezeNet model, the character level accuracy plot of SqueezeNet is shown in Figure 3(a), and the character level loss plot of SqueezeNet is shown in Figure 3(b).



(a) (b)

Figure 3. Plots of SqueezeNet model in (a)accuracy plot and (b) loss plot

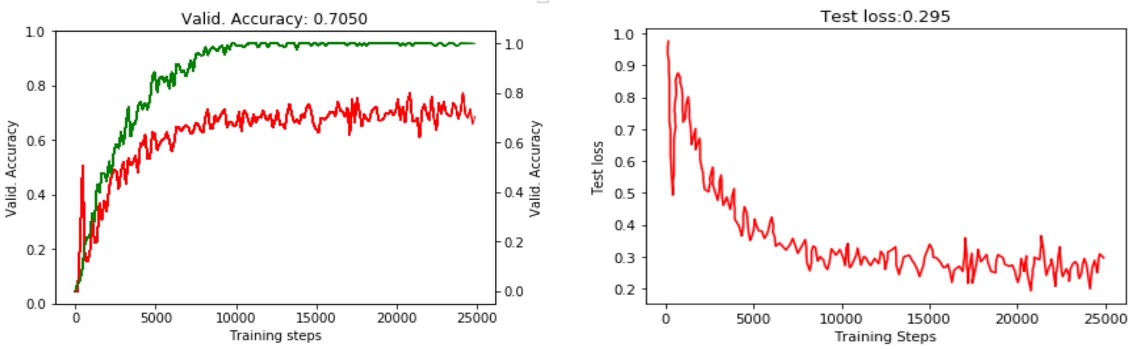
VGG19, a variant of the VGG architecture, is characterized by its deep and uniform structure with small 3x3 convolutional filters. While slightly trailing SqueezeNet, with an accuracy of 87%, and a loss rate of 13%, VGG19 remains a robust choice for image classification tasks. Its clear layer-wise architecture makes it well-suited for transfer learning and understanding hierarchical features within images, with 14402816 parameters. Figure 4 shows plots of the VGG19 model. The VGG19 character level accuracy plot is shown in Figure 4(a) and the character level loss plot is shown in Figure 4(b).



(a) (b)

Figure 4. Plots of the VGG19 model in (a) accuracy plot and (b) loss plot

ShuffleNet employs channel shuffling to optimize computation and memory usage, making it an efficient choice for various applications. In our evaluation, ShuffleNet demonstrated good capabilities with a 70.5% accuracy and loss rate of 29.5%. The total number of parameters in the ShuffleNet model is 1312400, which is very few compared to the VGG19 model. However, it appears to be less suited for character recognition tasks compared to SqueezeNet and falls behind in terms of accuracy. Figure 5 shows the plots of the ShuffleNet model. The ShuffleNet character level accuracy plot is shown in Figure 5(a) and the ShuffleNet Character level loss plot is shown in Figure 5(b).



(a) (b)

Figure 5. Plots of ShuffleNet model in (a) accuracy plot and (b) loss plot

In our analysis of the performance plots, it became evident that SqueezeNet outperformed other architectures, emerging as the top performer with an impressive accuracy of 94%. This underscores SqueezeNet's efficiency in capturing crucial features for the given task. VGG19, although achieving a respectable accuracy of 87%, lags slightly behind SqueezeNet in performance. On the other hand, ShuffleNet, with an accuracy of 70.5%, showcases its capabilities but appears to be less optimal for character recognition when compared to the superior performance of SqueezeNet and the solid performance of VGG19.

Upon reviewing the loss plots for SqueezeNet, VGG19, and ShuffleNet which is shown in Figure 6, it is evident that SqueezeNet achieved the lowest loss rate, followed by VGG19, and lastly, ShuffleNet with the highest loss rate. After a thorough analysis of all the models, it becomes clear that SqueezeNet is the most advantageous selection for our dataset, demonstrating superior performance in minimizing loss. The character level loss rate comparison of SqueezeNet, VGG19, and ShuffleNet are shown in Figure 6. The parameters and the accuracies comparison of the SqueezeNet, VGG19, and ShuffleNet models are shown in Table 1.

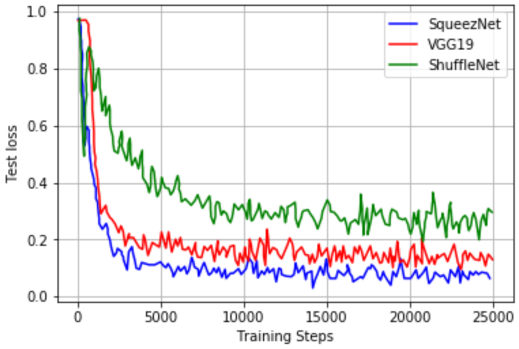


Figure 6. Comparison of loss rates of SqueezeNet, VGG19, and ShuffleNet

Table 1. Parameters and accuracies comparison

Model	Parameters	Character accuracy	Word accuracy
SqueezeNet	861344	94	80
VGG19	14402816	87	75
ShuffleNet	1312400	70.5	62

Table 1 shows the SqueezeNet neural network having 861344 parameters, achieves a notable character-level accuracy of 94% and word accuracy of 80% showcasing efficiency in both feature extraction and generalization with a smaller number of params. VGG19, with a more complex architecture having 14402816 parameters, achieves competitive character-level accuracy of 87% and word-level accuracy of 75%. ShuffleNet, designed for computational efficiency has 1312400 parameters, and sacrifices some accuracy, yielding a character-level accuracy of 70.5% and word-level accuracy of 62%. SqueezeNet stands out for its balanced performance and lower parameter count, making it a versatile choice for tasks where computational efficiency is crucial without compromising precision.

The accuracy comparison of the previous research OCR models [12], [13], [24], [25] is shown in Table 2. The table shows the comprehensive comparison of the identified OCR models for the Telugu language. The table provides valuable insights into their capabilities and potential applications in digitizing historical documents, automated form processing, and enhancing accessibility.

Table 2. Comparison of OCR models for the Telugu language

Name	Technique followed	Type of text	Achieved accuracy (%)
Prameela. N	SVM	Handwritten characters	80.6
Sarika. N	VGG-16	Handwritten characters	92
P.N. Sastri	Nearest neighbor	Handwritten characters	78
M. Mathew	HYBRID CNN-RNN	Printed text	75.6
Proposed model	SqueezeNet	Handwritten characters	94
		Handwritten text	80

From Table 2 the proposed model has achieved the highest character recognition rate as well as text recognition rate. Notably, the proposed SqueezeNet model outperformed all other models with an impressive accuracy of 94%, suggesting its effectiveness in OCR tasks, making it well-suitable for digitalization of Telugu documents.

5. CONCLUSION

The comprehensive analysis conducted on SqueezeNet, VGG19, and ShuffleNet within the realm of image classification has revealed distinct performance characteristics. Our research provides compelling evidence that SqueezeNet emerges as the most favorable choice, showcasing a superior loss rate and a commendable level of accuracy. Its compact architectural design, complemented by minimal parameters, renders it particularly suitable for scenarios where computational efficiency holds paramount importance. While VGG19 slightly trails in loss rate, it demonstrates competitive accuracy, notably excelling in tasks necessitating intricate feature extraction. In contrast, ShuffleNet, prioritizing computational efficiency, exhibits a trade-off with decreased accuracy. Delving into the nuanced trade-offs between accuracy, loss, and model complexity, our study unequivocally positions SqueezeNet as the optimal choice for Telugu handwritten text classification. Its adept balance between high accuracy and minimal parameters establishes it as a robust contender for image classification tasks, especially in resource-constrained environments. Our pioneered segmentation algorithm is finely tuned to dissect words with remarkable precision and preserve the important features of the words thereby substantially enhancing segmentation efficacy. Additionally, our findings underscore the significance of aligning model selection with the specific requirements of the task at hand. Emphasizing the necessity for a discerning trade-off between accuracy and computational efficiency, our research advocates for a meticulous approach to model selection in image classification endeavors.

REFERENCES

[1] J. Memon, M. Sami, R. A. Khan, and M. Uddin, “Handwritten optical character recognition (OCR): a comprehensive systematic literature review (SLR),” *IEEE Access*, vol. 8, pp. 142642–142668, 2020, doi: 10.1109/ACCESS.2020.3012542.

[2] M. Mathew and C. Jawahar, “An empirical study of CTC based models for OCR of Indian languages,” *arXiv preprint arXiv:2205.06740*, 2022, [Online]. Available: https://doi.org/10.48550/arXiv.2205.06740.

[3] P. Sahare and S. B. Dhok, “Multilingual character segmentation and recognition schemes for Indian document images,” *IEEE Access*, vol. 6, pp. 10603–10617, 2018, doi: 10.1109/ACCESS.2018.2795104.

[4] M. K. Sharma and V. P. Dhaka, “Segmentation of English Offline handwritten cursive scripts using a feedforward neural network,” *Neural Computing and Applications*, vol. 27, no. 5, pp. 1369–1379, Jul. 2016, doi: 10.1007/s00521-015-1940-x.

[5] A. Rehman and T. Saba, “Performance analysis of character segmentation approach for cursive script recognition on benchmark database,” *Digital Signal Processing*, vol. 21, no. 3, pp. 486–490, May 2011, doi: 10.1016/j.dsp.2011.01.016.

[6] R. Sarkhel, N. Das, A. Das, M. Kundu, and M. Nasipuri, “A multi-scale deep quadtree based feature extraction method for the recognition of isolated handwritten characters of popular indic scripts,” *Pattern Recognition*, vol. 71, pp. 78–93, Nov. 2017, doi: 10.1016/j.patcog.2017.05.022.

[7] H.-C. Fu, H-Y Chang, Y. Y. Xu, and H.-T. Pao, "User adaptive handwriting recognition by self-growing probabilistic decision-based neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 6, pp. 1373–1384, 2000, doi: 10.1109/72.883451.

[8] N. Asma and K. Zafar, "Comparative analysis of raw images and meta feature based urdu OCR using CNN and LSTM," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 1, 2018.

[9] Y.-C. Wu, F. Yin, and C.-L. Liu, "Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models," *Pattern Recognition*, vol. 65, pp. 251–264, May 2017, doi: 10.1016/j.patcog.2016.12.026.

[10] S. Saha, N. Paul, S. K. Das, and S. Kundu, "Optical character recognition using 40-point feature extraction and artificial neural network," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 4, 2013.

[11] V. J. Dongre and V. H. Mankar, "Devnagari handwritten numeral recognition using geometric features and statistical combination classifier," *International Journal on Computer Science and Engineering*, vol. 5, no. 10, pp. 856–863, 2013.

[12] N. Sarika, N. Sirisala, and M. S. Velpuru, "CNN based optical character recognition and applications," in *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, Jan. 2021, pp. 666–672, doi: 10.1109/ICICT50816.2021.9358735.

[13] N. Prameela, P. Anjusha, and R. Karthik, "Off-line Telugu handwritten characters recognition using optical character recognition," in *2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA)*, Apr. 2017, pp. 223–226, doi: 10.1109/ICECA.2017.8212801.

[14] P. Selvakumar and S. H. Ganesh, "Tamil character recognition using canny edge detection algorithm," in *2017 World Congress on Computing and Communication Technologies (WCCCT)*, Feb. 2017, pp. 250–254, doi: 10.1109/WCCCT.2016.68.

[15] C. Xiu, H. Yin, and Y. Liu, "Image segmentation of CV model combined with sobel operator," in *2020 Chinese Control And Decision Conference (CCDC)*, Aug. 2020, pp. 4356–4360, doi: 10.1109/CCDC49329.2020.9164450.

[16] M. Muralikrishna and D. V. R. Koti Reddy, "An OCR-character segmentation using routing based fast replacement paths in reach algorithm," in *2011 International Conference on Image Information Processing*, Nov. 2011, pp. 1–7, doi: 10.1109/ICIIP.2011.6108848.

[17] D. Crockett, "Direct visualization techniques for the analysis of image data: the slice histogram and the growing entourage plot," *International Journal for Digital Art History*, no. 2, 2016.

[18] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[19] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and<0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[20] M. Hassanpour and H. Malek, "Document image classification using SqueezeNet convolutional neural network," in *2019 5th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, Dec. 2019, pp. 1–4, doi: 10.1109/ICSPIS48872.2019.9066032.

[21] S. V. Muni, G. Tejasree, and K. M. Ravi, "Telugu handwritten character dataset," *IEEE Dataport*, 2020.

[22] N. S. Rani, A. C. Subramani, A. Kumar P., and B. R. Pushpa, "Deep learning network architecture based Kannada handwritten character recognition," in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, Jul. 2020, pp. 213–220, doi: 10.1109/ICIRCA48905.2020.9183160.

[23] Z. Li, Y. Su, Y. Zhang, H. Yin, J. Sun, and X. Wu, "Remote sensing image classification method based on improved ShuffleNet convolutional neural network," *Intelligent Data Analysis*, vol. 28, no. 2, pp. 397–414, Apr. 2024, doi: 10.3233/IDA-227217.

[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," in *2014 International Conference on IT Convergence and Security (ICITCS)*, Oct. 2014, pp. 1–4, doi: 10.1109/ICITCS.2014.7021817.

[25] M. Mathew, M. Jain, and C. V. Jawahar, "Benchmarking scene text recognition in Devanagari, Telugu and Malayalam," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Nov. 2017, pp. 42–46, doi: 10.1109/ICDAR.2017.364.

BIOGRAPHIES OF AUTHORS

Buddaraju Revathi is working as an assistant professor in Electronics and Communication Engineering (ECE) at Sagi Rama Krishnam Raju Engineering College, Bhimavaram. Areas of interest are Wireless Communication, Machine Learning, Deep Learning, and Image. Strong education professional with a Bachelor's degree in ECE. She can be contacted at email: buddaraju.revathi@gmail.com.

B N V Narasimha Raju currently working as an Assistant Professor in Computer Science and Engineering at Sagi Rama Krishnam Raju Engineering College. His academic pursuits center around Information Retrieval, Natural language Processing, Machine Translation, Artificial Intelligence, and Deep learning. He can be contacted at email: buddaraju.narasimharaju@gmail.com.

Ajay Dilip Kumar Marapatla an Assistant Professor in the Department of CSE-AIML and IoT at Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology. His research areas are the Internet of Things and Machine Learning. He has 12 Years of teaching experience. Currently, he is pursuing a Ph.D. from Pondicherry Central University, Pondicherry. He can be contacted at email: ajaydilipkumar_m@vnrvjiet.in; or dileepmarapatla@gmail.com.

Kagitha Veeramanikanta is currently studying the final year in the stream of Electronics and Communication Engineering (ECE) in Sagi Rama Krishnam Raju Engineering College, Bhimavaram. Areas of interest are IoT, artificial intelligence, and machine learning. He can be contacted at email: veeramanikanta2002@gmail.com.

Katta Dinesh is currently studying the final year in the stream of Electronics and Communication Engineering (ECE) in Sagi Rama Krishnam Raju Engineering College, Bhimavaram. Areas of interest are embedded, artificial intelligence, and machine learning. He can be contacted at email: kattadinesh3201@gmail.com.

Maddirala Supraja is currently studying the final year in the stream of Electronics and Communication Engineering (ECE) in Sagi Rama Krishnam Raju Engineering College, Bhimavaram. Areas of interest are IoT, embedded, and machine learning. She can be contacted at email: maddiralasupraja4@gmail.com.