Optical and Quantum Electronics (2023) 55:976
https://doi.org/10.1007/s11082-023-05211-y

Optical handwritten character recognition for Tamil language using CNN-VGG-16 model with RF classifier

N. Pughazendi[1] · M. HariKrishnan[1] · Rashmita Khilar[2] · L. Sharmila[3]

Received: 13 March 2023 / Accepted: 20 July 2023 / Published online: 3 September 2023

Abstract
In this world of modern data, it is so difficult to recognize handwritten characters for Tamil as many people have different styles of writing, so some of the letters are very difficult to understand and only a few can understand them. So, to overcome this issue, we built an algorithm in which the system could recognize the character and return the output. As it is difficult to understand letters manually for all their text, there is a need for some automatic method. The only intention of character recognition is that it wants to create a high-quality, accurate result that has the important points while considering the outlined input source image. Mostly, natural language processing and machine learning face the same problem with text recognition. The main goal of automatic character recognition is to create a high degree of accuracy as best as a human can do. Character recognition is the process of filtering the required information from the input-trained source to output the most useful content. This paper proposes a CNN-VGG16-RF model (convolution neural network-VGGNet-random forest) which employs an effective method to pick out the correct output. Experimental tests for our model were carried out to evaluate text quality, and the Tamil language dataset from the HP Tamil Lab website was used to compare our model to some other models; our model was found to be more effective in solving the handwritten recognition problem. In this model, we are going to propose Tamil vowels such as 12 letters only for the training and testing process.

Keywords Handwritten character recognition (HCR) · Tamil · CNN-VGG16-RF (convolution neural network-VGGNet-random forest) · Transfer learning · Recognize handwritten character (RHC) · HP Tamil Lab website · Handwritten Tamil character recognition (HTCR) · Support vector machine (SVM)

1 Introduction

Tamil is one of the classical languages and also one of the languages of the Dravidian language family in the world. Spoken by Tamil people in the state of Tamil Nadu, India, Sri Lanka, Malaysia, Singapore, and some other countries. As of now, it is the eighteenth most spoken language, with over 80 million native speakers worldwide. It is one of the official languages in India, Singapore, Malaysia, and Sri Lanka. The oldest Tamil writing is

Extended author information available on the last page of the article historical evidence in inscriptions and potsherds from the fifth century BCE. The language was distinguished by analyses of grammatical and lexical changes based on three time periods: Olden Tamil (from 450 BCE to 700 CE), Middle Tamil (700–1600), and Modern Tamil (after 1600). The Tamil writing method evolved based on the Brahmi script. The shape of the letters changed enormously over time, eventually stabilizing when printing was introduced in the sixteenth century CE. A total of 247 characters, which include vowels and consonants $(12+18+1+(12\times 18))$. And also, Tamil has borrowed 5 consonants from Sanskrit, which would produce another 60 compound characters when combined with vowels. The total Tamil language has 307 characters.

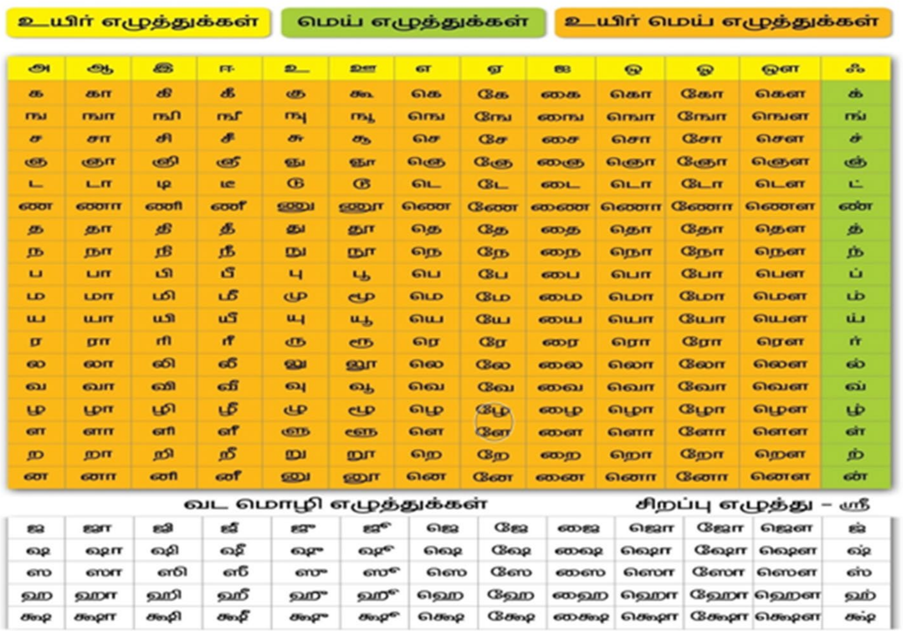
The handwritten text aims to recognize the output Tamil characters from the input text image for a total of 247 Tamil characters. The reason for recognizing the Tamil offline characters is that everyone has a different writing style and variation for every character, so it can be difficult for the machine to identify the input handwritten character. For that, we use the concept named Convolution Neural Network (CNN) to recognize the Tamil characters. Why does CNN mean it is a part of a deep neural network and is mostly used for image and text classification and has a higher accuracy level? The CNN can be broadly classified into 3 major stages, such as (1) input, (2) feature extraction, and (3) classification for classifying the Tamil handwritten text.

Our proposed method is CNN-VGG16-RF, which is a combination of both convolution neural networks and random forest. Where CNN is used in Feature Extraction whereas RF is used for Classification of image dataset. Here VGG16 is a type of CNN that is consider for best algorithm for image classification and is very easy to use with transfer learning. The Transfer Learning is the knowledge of already trained machine is gather knowledge from previous task and applied to different but similar problem. We combined these two methods to solve the problem, first reducing execution time which is apply the concept of transfer learning technique and then increasing recognition accuracy by means of CNN-VGG16-RF technique. For that we are using the Tamil language dataset from HP Lab India website for our proposed method.

It has been state that based on the various surveys we come to know that only few Tamil dataset is available because of the Tamil language character as different style, size and orientation angle is difficult and vary for each person. And only few classification methods are available when compared to other language classification method. From the above few Tamil classification technique, we cannot be able to achieved the highest accuracy and the execution time is also high when compared to other Tamil HCR technique. So that our proposed method is concentrate on high accuracy and reducing the execution time is achieved by our proposed CNN-VGG16 with RF classifier (Fig. 1).

2 Literature survey

- Kavitha and Srimati (2019) have proposed the state of the art in CNN in recognizing handwritten Tamil characters in offline mode based on traditional approaches. The challenging part of HCR is the variations in the writing styles of a person can vary. The approach used for HCR was traditional machine learning for long time. When a new character image is given as input, the system able to recognize it accurately before an offline HCR system is first trained with the set of characters (as scanned images). This work is an endeavor to benchmark for offline HTCR using deep learning techniques as a benchmark. The drawback is that most of the errors were due to writing ambiguities



உயிர் எழுத்துக்கள் | மெய் எழுத்துக்கள் | உயிர் மெய் எழுத்துக்கள்

அ	ஆ	இ	ஈ	உ	ஊ	எ	ஏ	ஐ	ஒ	ஓ	ஔ	ஃ
க	கா	கி	கீ	கு	கூ	கெ	கே	கை	கொ	கோ	கௌ	க்
ங	ஙா	ஙி	ஙீ	ஙு	ஙூ	ஙெ	ஙே	ஙை	ஙொ	ஙோ	ஙௌ	ங்
ச	சா	சி	சீ	சு	சூ	செ	சே	சை	சொ	சோ	சௌ	ச்
ஞ	ஞா	ஞி	ஞீ	ஞு	ஞூ	ஞெ	ஞே	ஞை	ஞொ	ஞோ	ஞௌ	ஞ்
ட	டா	டி	டீ	டு	டூ	டெ	டே	டை	டொ	டோ	டௌ	ட்
ண	ணா	ணி	ணீ	ணு	ணூ	ணெ	ணே	ணை	ணொ	ணோ	ணௌ	ண்
த	தா	தி	தீ	து	தூ	தெ	தே	தை	தொ	தோ	தௌ	த்
ந	நா	நி	நீ	நு	நூ	நெ	நே	நை	நொ	நோ	நௌ	ந்
ப	பா	பி	பீ	பு	பூ	பெ	பே	பை	பொ	போ	பௌ	ப்
ம	மா	மி	மீ	மு	மூ	மெ	மே	மை	மொ	மோ	மௌ	ம்
ய	யா	யி	யீ	யு	யூ	யெ	யே	யை	யொ	யோ	யௌ	ய்
ர	ரா	ரி	ரீ	ரு	ரூ	ரெ	ரே	ரை	ரொ	ரோ	ரௌ	ர்
ல	லா	லி	லீ	லு	லூ	லெ	லே	லை	லொ	லோ	லௌ	ல்
வ	வா	வி	வீ	வு	வூ	வெ	வே	வை	வொ	வோ	வௌ	வ்
ழ	ழா	ழி	ழீ	ழு	ழூ	ழெ	ழே	ழை	ழொ	ழோ	ழௌ	ழ்
ள	ளா	ளி	ளீ	ளு	ளூ	ளெ	ளே	ளை	ளொ	ளோ	ளௌ	ள்
ற	றா	றி	றீ	று	றூ	றெ	றே	றை	றொ	றோ	றௌ	ற்
ன	னா	னி	னீ	னு	னூ	னெ	னே	னை	னொ	னோ	னௌ	ன்

வட மொழி எழுத்துக்கள் சிறப்பு எழுத்து – ஸ்ரீ

ஜ	ஜா	ஜி	ஜீ	ஜு	ஜூ	ஜெ	ஜே	ஜை	ஜொ	ஜோ	ஜௌ	ஜ்
ஷ	ஷா	ஷி	ஷீ	ஷு	ஷூ	ஷெ	ஷே	ஷை	ஷொ	ஷோ	ஷௌ	ஷ்
ஸ	ஸா	ஸி	ஸீ	ஸு	ஸூ	ஸெ	ஸே	ஸை	ஸொ	ஸோ	ஸௌ	ஸ்
ஹ	ஹா	ஹி	ஹீ	ஹு	ஹூ	ஹெ	ஹே	ஹை	ஹொ	ஹோ	ஹௌ	ஹ்
க்ஷ	க்ஷா	க்ஷி	க்ஷீ	க்ஷு	க்ஷூ	க்ஷெ	க்ஷே	க்ஷை	க்ஷொ	க்ஷோ	க்ஷௌ	க்ஷ்

Fig. 1 Tamil character letter

between similar characters. For an HCR, we use the dataset developed by HP Labs India.

- Shams et al. (2020) have proposed the Arabic handwritten script recognition in the domain of handwriting recognition. The algorithms focused on integrating two classifiers; CNN and SVM, for offline Arabic handwriting recognition (OAHR), on which the dropout technique was applied. A CNN is used for feature extraction and SVM functions as a recognizer. From this model, we have found that it both automatically extracts features from the raw images and performs classification. Additionally, we protected our model against over-fitting due to the powerful performance of dropouts. The training and test sets were taken from the HACDB and IFN/ENIT databases.
- Antony Robert Raj and Abirami (2019) has proposed that handwritten character recognition is a tough task due to the presence of slant, variation in large, discontinuities, and the character's freestyle. This paper deals with an algorithm named "hierarchical classification based on SVM" is used for predicting the character from its character features using a divide-and-conquer procedure. The key features of this paper are the strip tree-based hierarchical formation, the Z-ordering algorithm, and finally the representation of the PM-Quad tree. The three different types of feature combinations are implemented in MATLAB. This algorithm was developed only for recognizing the characters that have a curvy nature and is capable of addressing all Tamil characters.
- Babitha Lincy and Gayathri (2020) have proposed that, unlike other languages, the Tamil language is more complex to recognize. Thus, the research work develops a novel Tamil HCR approach, the FC layer and weights are fine-tuned by a new Self Adaptive Lion Algorithm (SALA) that is a conceptual improvement of the standard Lion Algorithm (LA). We introduced a CNN model for recognizing the Tamil characters with an optimally configured neural network. This in turn introduces a new Self Adaptive Lion

Algorithm (SALA), which is the advanced version of the Lion algorithm. The proposed model accuracy is better than DCNN, RNN, LA, and EHO-NN, respectively.

- Gupta and Bag (2021) have proposed that handwritten numeral recognition is much more complex than printed one because of the different writing styles. In this work, we have developed a multilingual handwritten digit from an independent numeral recognition system, which is independent of fusion, and every single numeric digit has only 10 classes corresponding to each one. Exhaustive experiments are done with a numeral dataset of eight Indic and non-Indic scripts. The proposed system is the first step in developing a script-independent handwritten numeral recognition system using a fusion-free approach for eight languages.
- Ahlawat et al. (2020) have proposed that training an optical character recognition (OCR) system based on these prerequisites is a challenging task. Research in the handwriting recognition field is focused on deep learning techniques and has achieved breakthrough performance in the last few years. CNN the most suitable approach for solving handwriting recognition problems. We focus on evaluating various SGD optimization algorithms in improving the performance of handwritten digit recognition. It is observed that deep neural architectures are more advantageous than shallow neural architectures. The MNIST database is used in this work. Thus, the proposed CNN architecture achieves accuracy even better than that of ensemble architectures, along with reduced operational complexity and cost.
- Vinotheni et al. (2021) have proposed a Convolutional Neural Network (CNN) designed to demonstrate a high capability of object recognition in image data. This paper proposed a deep neural network model based on the use of convolutional neural networks to classify the handwritten images from the MNIST dataset. The experimental results achieved a high accuracy score with one hidden layer and 10 epochs. The increase in the number of epochs increases the complexity and does not guarantee an enhancement in the model performance. The model performance, especially for complex and large datasets, has a great effect on the depth architecture.
- Salameh and Surakhi (2020) have proposed that the recognition of Tamil handwritten characters is the leading research area for improving inaccuracy. This work presents modified convolution neural network (M-CNN) architecture to achieve a faster convergence rate and also to get the highest recognition accuracy. Systematic experiments on an isolated handwritten Tamil character dataset collected from various schools by ourselves. Our dataset comprises 54,600 images of 156 distinct classes of Tamil isolated characters, which are identically visual or written similarly by most people. The CNN in a neural system network is a multilayer feedforward that automatically extracts the discriminative features multiple times from the high dimensionality of input data and also can learn high-dimensional complexes with nonlinear mapping.
- Prabavathi et al. (2021) have proposed that Prehistoric Tamil character recognition is a technical challenge compared to other languages for the similarity and complexity of characters. The research challenge in recognizing Tamil characters is mainly because of the characters consisting of the number of holes, loops, and curves. To overcome the issues behind Tamil character recognition, our proposed model will work well. This system takes a challenge to recognize prehistoric Tamil characters using a deep neural network. The proposed work uses the simplex method for optimization of the deep neural networks to achieve accuracy.
- Suriya et al. (2020) have proposed a system that is capable of recognizing characters in a variety of challenging conditions using the Convolutional Neural Network. It use Tamil character dataset developed by HP Labs India. The major drawback, as

in the case of any HCR problem, is the writing styles of the individual at different times and among various people. In this project, I have projected various aspects of each phase of the offline Tamil character recognition process. As a result, among the proposed algorithms, it has been found that different CNN models produce different results, and the best one yields the highest recognition accuracy.

- Bhardwaj (2020) has proposed the process where the machine detects and recognizes the characters from a text image and converts that processed data into a code that is understood by the machine. In this paper, a Devnagari Character Dataset a new dataset for Devnagari script characters is used. The proposed model explored the difficulty in the classification of characters in the Devnagari Dataset. By Deep CNN, we use the dropout and dataset increment approach to improve test accuracy. By implementing these techniques in Deep CNN, we achieved a test accuracy of nearly 98% percent. The proposed architecture scored the highest test accuracy of 98.13%. The experimental results suggested that Deep CNNs with added Dropout layer and Dataset increment technique can result in more test accuracy even for a diverse and challenging dataset.
- Danthuluri et al. (2021) has proposed deep learning algorithms for character recognition. This paper suggests a Convolutional Neural Network-based character recognition method. Train your dataset and classify it to see the results. The EMNIST dataset provides approximately 132,000 images of 47 people that can be recognized as learning. Convolution neural networks were used to train EMNIST datasets for high accuracy. The input image is pre-processed, standardized, standardized, and provided with a categorizer to predict characters.
- Preetha (2020) have proposed the application of pattern recognition through the image for Handwriting recognition. There are several approaches, among these strategies, the very best accuracy is achieved from Convolutional Neural Network and also the least accuracy is achieved from the Slope and Slant Correction methodology. A high accuracy will achieve when trained the image in CNN and only the drawback is that training time is too long when large image sample are included. Handwriting recognition is extremely difficult as a result of all the people have totally different handwriting and it becomes a lot of advanced to notice once this area unit compared to a different.
- Ram Kumar (2020) have proposed the set of Tamil Handwritten Characters as input, train the CNN algorithm to acknowledge the pattern and convert the recognized characters to a Printed document. It differs from traditional approach of Tamil Handwritten Character Recognition (THCR) in extracting the features by the methods. This work is for digitalizing offline THCR using deep learning technique. This digital text can be stored in a document file, and also a training and test accuracy of 93% is obtained by this CNN model which is better than other classification technique like Clustering and groupwise classification, SVM, Component labelling method and ANN.
- Deepa (2019) have proposed the recognize handwritten characters for Tamil alphabets using multilayer Convolutional Neural Networks without feature extraction. In the proposed system, each character is resized into required pixels, which is directly subjected to training. A neural network based off line HCR system without feature extraction has been introduced in this paper for classifying and recognizing the Tamil alphabets. The pixel values for the segmentation stage have been directly used for training the neural network and have been found to yield to be highest accuracy of 90.19%.

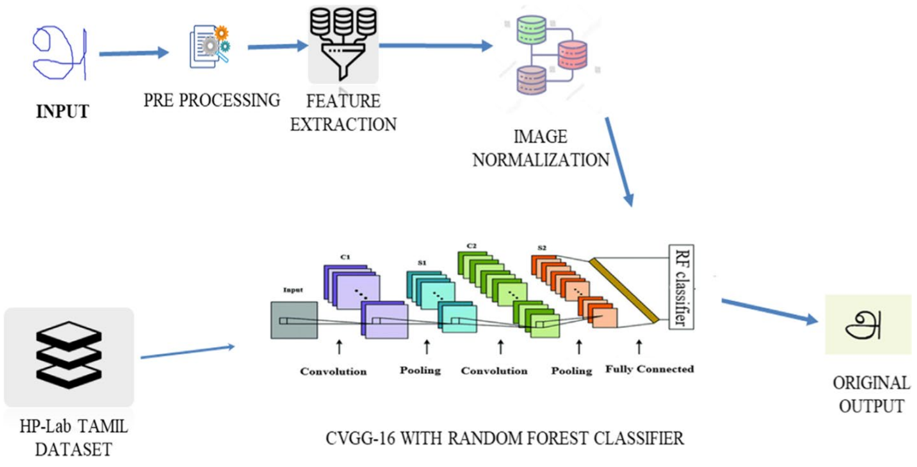


Fig. 2 CNN-VGG16-RF architecture

Before Denoising Image
After Denoising Image

Fig. 3 Pre-processing

3 The proposed model

The proposed recognition process involves the following stages (Fig. 2).

3.1 Pre-processing

Pre-processing is a sequence of operations, which increases the accuracy of the image. For the hand-written character recognition process, the following pre-processing techniques are followed, image binarization, normalization, and noise reduction (Fig. 3).

3.2 Feature extraction

It is the process of extracting the features from the input image for the classification process. It is an important step for achieving high recognition accuracy. Each character contains its own features such that each of the features is done in this phase (Fig. 4).

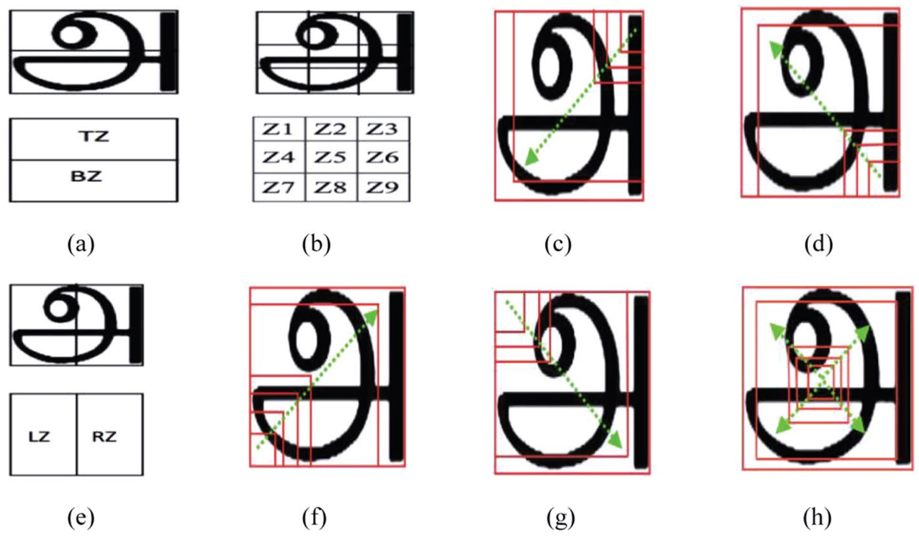


Fig. 4 Feature extraction

3.3 Image normalisation

Normalization is a process that changes the range of pixel intensity values. It is usually called contrast stretching or histogram stretching. From the input image, the normalization is administered by removing the background pixels, and therefore the character alone is provided because it is within the images. The value of the pixel is more than the value of '0' for the region where the character 'A' is written, and all other regions have pixel value '0' after normalizing (Fig. 5).

3.3.1 VGG-16 model

A pre-trained model like VGG16 is also called the "top" model. Once the VGG16 model has been imported, excluding the "top" of the model, we can use any one of the 2 Transfer Learning approaches.

1. Feature extraction approach

Before Normalization
After Normalization

Fig. 5 Image normalization

In this approach, we create a new dataset for our input images from the pre-trained model architecture. We'll import the Convolutional and Pooling layers but leave the fully-connected layer.

The feature extraction is carried out in the following manner-

1. Download the pre-trained VGG16 model. Ensure that the fully-connected layer—is not included.
2. Pass the image data through the pre-trained model to extract convolved visual features

The output feature stack will be 3D, and for it to be used for prediction by other machine learning classifiers.

1. From this point, we have two options:

Stand-alone extractor: We can use the pre-trained VGG16 model layers to extract image features once. Let you create a new dataset that doesn't require image processing.

Bootstrap extractor: Write your code for the FC layer, and integrate it with your VGG16 model layers. You are creating custom code in your top model into your VGG16 model layers. Initialize this FC layer with random no. of weights, which will update via the backpropagation method.

2. Fine-tuning approach

In this approach, we use Fine-Tuning. It allows a portion of our VGG16 model layers. From the previous approach, we used the pre-trained layer to extract features. We passed our HP Tamil Lab dataset through the convolutional layers and weights, outputting the transformed visual features. Fine-tuning a Pre-trained VGG16 Model layer entails:

1. Bootstrapping a new top portion of the model
2. Freezing pre-trained VGG16 convolutional layers
3. Un-freezing the last few pre-trained VGG16 layers training.

In this method, we use the concept of the Feature extraction approach for image classification under this using bootstrap extractor.

VGG-16 architecture with random forest classifier using transfer learning

Transfer learning

It is a process where a model trained on one problem is used in some way on a second related problem. In deep learning, TL is a technique where by a neural network model is first trained on a problem similar to the problem that is being solved.

The learning process during transfer learning is:

- *Fast*—Normal CNN will take days or even weeks to train, but you can cut short the process with TL.
- *Accurate*—Transfer learning model performs 20% better than a custom-made model.

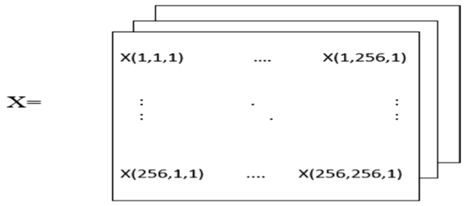
- *Needs less training data*—Being trained on a large dataset, the model can already detect specific features and need less training data to further improve the model.

Transfer learning on image data

To demonstrate transfer learning here, I've chosen a large dataset of the multiclass classifier which can be found here official page of Hp Lab Tamil Page. This data consists of 156 classes of Tamil Letter, which as more than 80,000 images (Fig. 6).

Our input image is in the form of the pixel value. The images in the dataset are fixed-size 256×256 RGB. This means that each color channel i.e. (R, G, B) contains each image in 256×256-pixel values. we can represent each input as a TensorFlow with dimensions (256,256,3) and label it as x. Each pixel value is between 0 and 225 for the scalar matrix.

We can represent a pixel value as x (i, j, k), where i is the value for the first-dimension j for a second, and k for the third (which recall is the channel). The first two dimensions represent the location of the pixel value, and the third dimension is the channel the pixel value belongs. The image as a tensor looks like this:



Our output ŷ will be a vector of probabilities for each of the 11 classes for the given image.

$$\hat{y} = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ - \\ - \\ \hat{y}_{99} \end{pmatrix} \tag{1}$$

Let's say we input an image x and the model believes the image belongs to class 1 with a probability of 3%, class 2 with a probability of 5%, class 4 with a probability of 7%, class

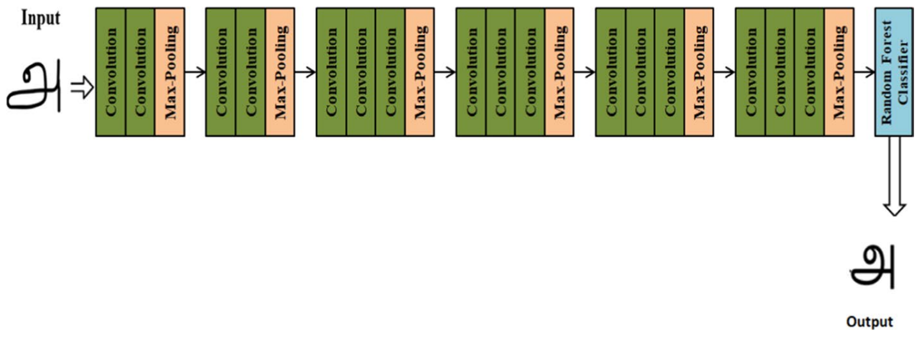


Fig. 6 CNN-VGG16-RF classifier

9 with a probability 2%, class 10 with probability 77%, and all other classes with probability 0%. In this situation, our output would look like the

$$\hat{y} = \begin{pmatrix} 0.03 \\ 0.05 \\ 0 \\ 0.07 \\ 0.02 \\ 0.77 \end{pmatrix} \tag{2}$$

Notice that the sum of the elements in ŷ is equal to 1. We use a softmax function at the end of our ConvNet to ensure this property. To compute accuracy, we use to create a vector of the top 5 classes in decreasing order of probability.

$$c = \begin{pmatrix} \boldsymbol{c_1} \\ \boldsymbol{c_2} \\ \boldsymbol{c_3} \\ \boldsymbol{c_4} \\ \boldsymbol{c_5} \end{pmatrix} \tag{3}$$

Using our model equation, we get:

$$c = \begin{pmatrix} 10 \\ 2 \\ 1 \\ 4 \\ 9 \end{pmatrix} \tag{4}$$

Let's say the image was an image of a letter 'அ'. The picture of the letter 'அ' is the 4th class. The picture of the other letter 'ஆ' is the 11th class. Since the image has two ground truth classes, we get:

$$c = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{pmatrix} \tag{5}$$

Using our example, we get for our ground truth labels:

$$c = \begin{pmatrix} 4 \\ 11 \end{pmatrix} \tag{6}$$

Let's $d\left(c_i c_k\right) = 0$, if $c_i = c_k$ and 1 otherwise. To calculate the error of the algorithm, we use:

$$e = \frac{1}{n} \sum_k \min_i d\left(ci, c_k\right) \tag{7}$$

For our example, the error is

$$e = \frac{1}{\mathrm{n}} \left(\min_i d\left(ci, c_k\right) + \min_i d\left(ci, c_k\right) \right) \tag{8}$$

$$e = \frac{1}{2}(0 + 0)$$
$$\mathrm{e} = 0 \tag{9}$$

Since the ground truth labels were in our top 5, we get an error of 0. We just disassemble the input and output for our model. Optimistically, it makes sense. So far, we've only considered the input and output for one example. Consider, m training examples that we wanted to input into the model as a batch? It's not too bad, we are just going to add another dimension.

$$\mathrm{X} = \left[\mathrm{x}^{(1)}, \mathrm{x}^{(2)}, \ldots \mathrm{x}^{(\mathrm{m})}\right] \tag{10}$$

So, we can think of X as a tensor with dimensions (m,256,256,3), where m is the number of examples in our batch. The superscript denotes which example it is in the batch for training. So $x^{(1)}$ will be the 1st training example from the batch. Similarly for output:

$$\mathrm{Y} = \left[\mathrm{y}^{(1)}, \mathrm{y}^{(2)}, \ldots \mathrm{y}^{(\mathrm{m})}\right] \tag{11}$$

Y is a tensor with dimensions (11, m), and we use the same superscript notation as above.

3.3.1.1 Convolutional layer It is the first layer, which extracts the various features from the input images. In this layer, various mathematical operation is performed between the input image and a filter of a particular size M × M by the convolution. By sliding the input image over the filter, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (M × M). The output is termed the Feature map which gives us detail about the image such as the corners and edges. Later, this feature map is fed to alternative layers to be told many alternative options of the input image.

$$\mathrm{a}^{[1]} = \mathrm{g}\left(\mathrm{W}^{[1]}\mathrm{a}^{[0]} + \mathrm{b}^{[1]}\right) \tag{12}$$

$a^{[1]}$ with dimension (256,256,64).

We add the same bias for each filter, so $b^{[1]}$ has dimensions (6,1).

A convolution is applied to×6 times using 6 different filters $W^{[1]}_{c(1)}, W^{[2]}_{c(2)}, \ldots, W^{[6]}_{c(6)}$. Reduce the dimensions to make it easier to visualize. x is a tensor with dimensions (256,256,3). Let's instead make it (5,5,3). $W^{[1]}_{c(1)}$ has dimensions (16,16,3,64). Let's make it (3,3,3,6). We won't change the dimensions for b (Kavitha and Srimati 2019). Finally, for $Z^{[1]}_{(1)}$ we'll change its dimensions from (256,256,64) to (3,3,6).

Bias ($\boldsymbol{b}^{[1]}$)

Using Convolution function,

$$\mathrm{z}^{[1]}_{(\mathrm{i,j,k})} = \left(\mathrm{x} * \mathrm{W}^{[1]}_{\mathrm{c}}\right)(\mathrm{i,j,k}) + \mathrm{b}^{[1]}_{(\mathrm{k},1)} \tag{13}$$

Which become,

$$\mathrm{z}^{[1]}_{(\mathrm{i,j,k})} = \sum_{l=1}^{3}\sum_{\mathrm{m}=1}^{3}\sum_{\mathrm{n}=1}^{3} \mathrm{x}(\mathrm{i}+\mathrm{l}-1, \mathrm{j}+\mathrm{m}-1, \mathrm{n})W^{[1]}_{c(l,m,n,k)} + \mathrm{b}^{[1]}_{(\mathrm{k},1)} \tag{14}$$

Same padding

Let's Define the transformation as $\mathbf{S}^{[0]} = \mathbf{h_p}(\mathbf{x})$, where **p** is equal to number of borders of 0, we place around **x**.

x is padded with one of the borders of **0**'s.

$$\mathrm{p} = \frac{\mathrm{f}-1}{2} \tag{15}$$

Relu operation

If our RELU function **g**, is gn, then

$$\mathrm{a}^{[1]} = \mathrm{g}\left(\mathrm{z}^{[1]}\right) \tag{16}$$

g is applied elementwise to every element in $z^{[1]}$. As we would expect, $a^{[1]}$ (3,3,6) and share the same dimension as $z^{[1]}$.

3.3.1.2 Pooling layer A Pooling Layer is come after by a Convolution Layer. The fundamental aim of this layer is to reduce the size of the convolution feature map to reduce computational costs. It is performed by decreasing the connections between layers. Depending on the various method used, there are several types of Pooling operations. In VGG16 we use, Max Pooling, the largest value is taken from the feature map. This Layer usually serves as a bridge between the Convolution Layer and the FC Layer.

The dimension of the input $a^{[2]}$ are (256,256,64) and the dimension of the output $m^{[2]}$ are (128,128,64).

Let's make $m^{[2]}$ have dimension (6,6,6) and $m^{[2]}$ have dimension (3,3,6).

Like the Convolution operation, we are sliding over our input and performing a pooling operation. If we were to compute the $m^{[2]}_{(i,j,k)}$, it would look like this:

$$\mathrm{m}^{[2]}_{(\mathrm{i,j,k})} = \max_{\mathrm{i*s} \le = \mathrm{l} < i*s+f, j*s \le l < j*s+f} a^{[2]}_{(l,m,k)} \tag{17}$$

Let's n_L represent the height and width of the input, and n_{L+1} represent height and width of our output. To find height and width (**f**) of the window we need to use max pooling, we use

$$\mathrm{n}_{\mathrm{L}+1} = \left\lfloor \frac{\mathrm{n_L} + 2\mathrm{p} - \mathrm{f}}{\mathrm{s}} + 1 \right\rfloor \tag{18}$$

Solving for **f**

$$\mathrm{f} = \mathrm{n_L} + 2\mathrm{p} - \mathrm{s}\left(\mathrm{n}_{\mathrm{L}+1} - 1\right) \tag{19}$$

where **p** is equal to the padding. Since we aren't using padding, we set **p = 0,** and plugged in $n_L = 2$, s = 2 and $n_{L+1} = 3$ we get,

$$\begin{aligned} \mathrm{f} &= 6 + 2 * 0 - 2(3-1) \\ \mathrm{f} &= 2 \end{aligned} \tag{20}$$

3.3.1.3 Fully connection layer This layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. It is usually placed before the output layer and forms the last few layers of a CNN Architecture. In this, the input image from the previous layers is flattened and fed to the FC layer. In this stage, the classification process begins to take place.

3.3.1.4 Random forest classifier It is a Supervised Machine Learning Algorithm that is extensively used in Classification and Regression problems (Geetha and Arunachalam 2022; Sri Vidhya and Karthi 2022). It provides solutions to complex problems which utilizes ensemble learning technique. Random Forest is a classifier that contains some decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. The important feature of the Random Forest Algorithm is that, can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It executes superior outcome for classification problems and consists of many decision trees. The 'forest' generated by the random forest algorithmic program is trained through bagging or bootstrap aggregating. The random forest algorithmic program is associate extension of the bagging methodology. Feature randomness, additionally called feature bagging or random subspace methods. In this article, we will see how to build a Random Forest Classifier using the Scikit-Learn library of Python programming language, and to do this, we use the HP Tamil LAB dataset.

4 Experiment setup

4.1 Dataset

The Dataset used for our model is from HP Labs India, this dataset consists of 156 different Tamil characters (hpl-Tamil-iso-char) written by native Tamil writers from various cities of Southern India using HP Laptop or desktops. The dataset contains approximately 200 samples for each class (with very few classes having around 300 samples) with a total of 82,928 samples and is freely available (Fig. 7). The entire Tamil character set can be represented with these 156 unique characters (Fig. 1).

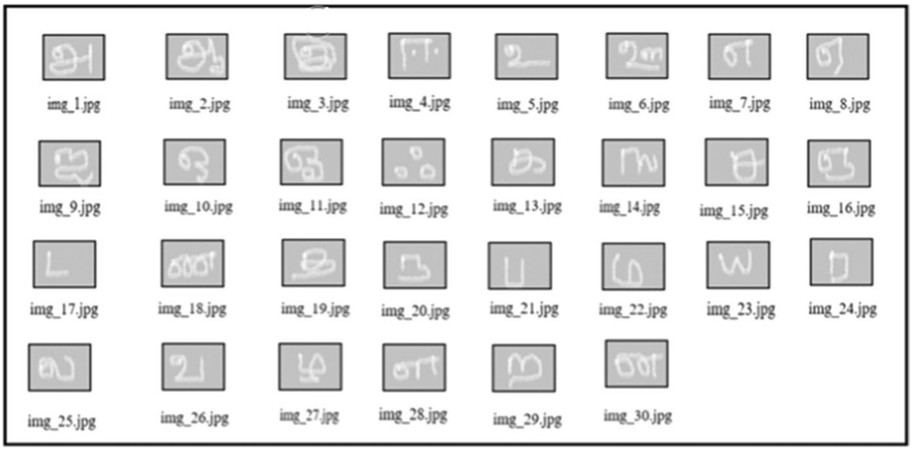


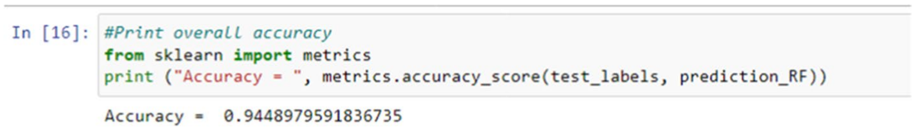
Fig. 7 Sample images from HP-Tamil lab dataset

4.2 Training and testing process

In this experimental setup, Tamil vowel characters i.e., uyireluttu such as only 11 classes are used, where each class for training set has around 340 samples and testing set for each 11 classes around 180 samples. A total of 3740 images were used in training and a total of 1980 images were used in testing phase. A total of 5720 images are used for run time. For training and testing the neural networks, Keras was used as a deep learning framework with the TensorFlow library. A necessary module should be installed for each phase of execution of the coding. The input data are acquired from the HP Tamil Lab database to carry out training and testing models. This dataset is composed of 80,000 training images. The samples are normalized with a fixed image size of 256 × 256. The entire training and testing were performed on a Windows 11 Pro, 64-bit Operating system- × 64 based processor, and the processor is AMD Ryzen 3 3200G with Radeon Vega Graphics 3.60 GHz and 16 GB ram is used. The Software which is used is Latest version of Anaconda3 in that we use the Jupiter Notebook for execution of the program. A Python Language is used for developing the program Module.

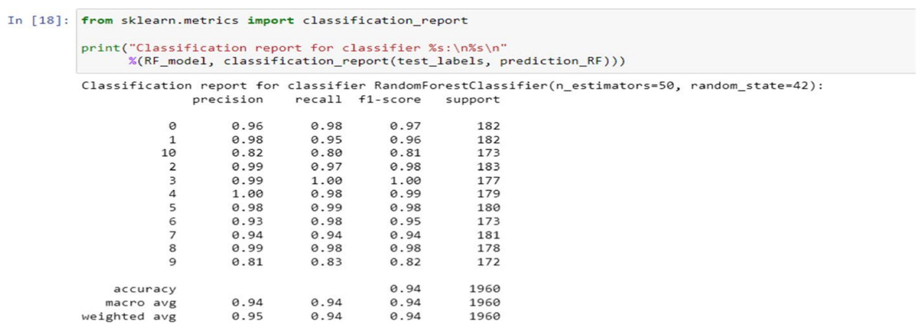
5 Result and discussion

The numerical results obtained from the experimentation of the proposed CNN-VGG16-RF approach are discussed in this section. Figure 8 depicts the prediction accuracy of the proposed CNN-VGG16-RF model. This is compared with existing approaches like CNN, SVM, MCNN, and RCNN respectively. The major disadvantage with CNN is that the training time is high when the parameter is large. If the class label is more than two means it is not getting an accurate prediction result. In the existing paper, they deal only with Lenet-5 or Alex net model for Tamil language recognition. The same CNN model was only used throughout the process with some modifications in method and a few Tamil characters were only classified in a previous paper. So, in our proposed method, we use the concept of the VGG-16 type of CNN model for feature extraction and we use the Random Classifier to classify the Tamil character. The VGG16 uses to handle large datasets efficiently using the transfer learning method with using the classifier namely such as Random Forest classifier to classify the Tamil character. Based on our proposed method we come to know that from above Fig. 8 it gives a better accuracy when compared to other HCR technique. And one more is that in our proposed model we use to identify the external source of Tamil Handwritten Character to identify the input Tamil character which is not been used in previous prescribe Tamil handwritten character recognition techniques using any of the methods.


```
In [16]: #Print overall accuracy
         from sklearn import metrics
         print ("Accuracy = ", metrics.accuracy_score(test_labels, prediction_RF))

         Accuracy =  0.9448979591836735
```

Fig. 8 Model accuracy



```
In [18]: from sklearn.metrics import classification_report

         print("Classification report for classifier %s:\n%s\n"
               %(RF_model, classification_report(test_labels, prediction_RF)))
```

```
Classification report for classifier RandomForestClassifier(n_estimators=50, random_state=42):
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       182
           1       0.98      0.95      0.96       182
          10       0.82      0.80      0.81       173
           2       0.99      0.97      0.98       183
           3       0.99      1.00      1.00       177
           4       1.00      0.98      0.99       179
           5       0.98      0.99      0.98       180
           6       0.93      0.98      0.95       173
           7       0.94      0.94      0.94       181
           8       0.99      0.98      0.98       178
           9       0.81      0.83      0.82       172

    accuracy                           0.94      1960
   macro avg       0.94      0.94      0.94      1960
weighted avg       0.95      0.94      0.94      1960
```

Fig. 9 Precision and recall value

5.1 Experimental result on /HP-lab tamil dataset

The HP Lab Tamil Dataset is applied to our model CNN-VGG16-RF and the accuracy is shown in Fig. 8.

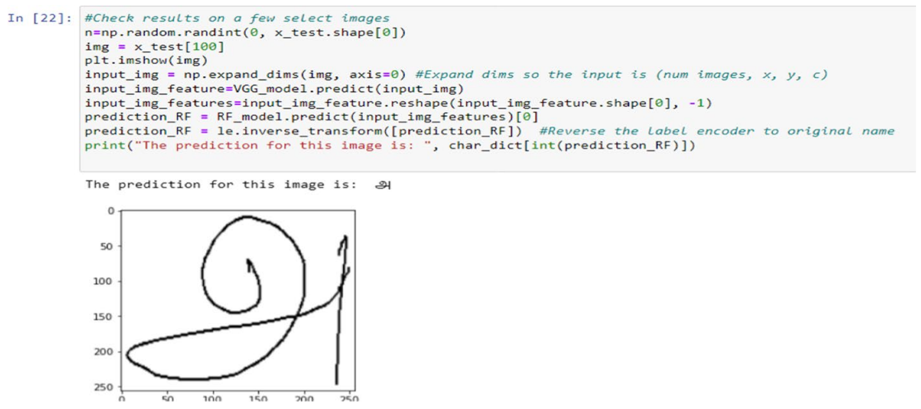
The Confusion matrix for our model is

See Fig. 9.

6 Precision and recall value:

The output with random index value is for our model is:

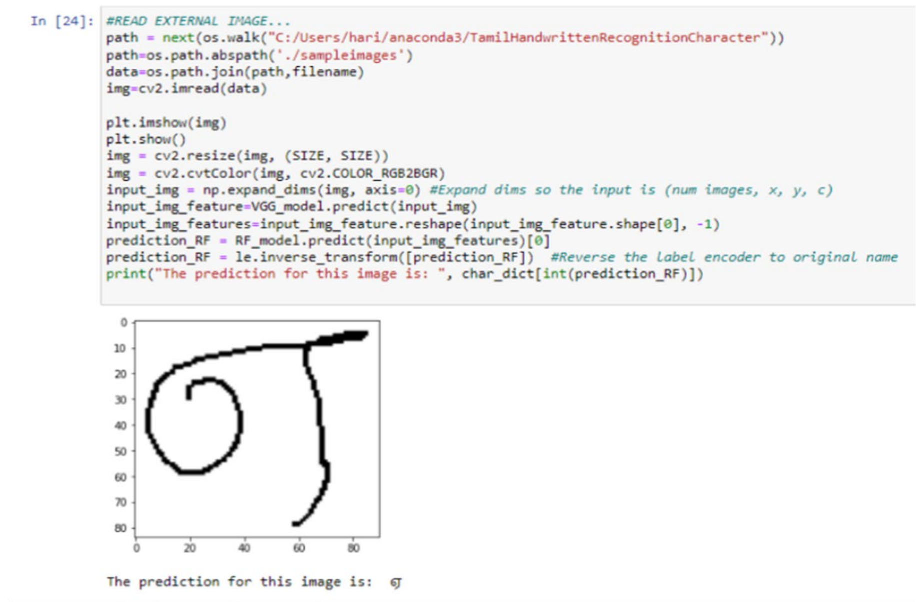
See Fig. 10.



```
In [22]: #Check results on a few select images
         n=np.random.randint(0, x_test.shape[0])
         img = x_test[100]
         plt.imshow(img)
         input_img = np.expand_dims(img, axis=0) #Expand dims so the input is (num images, x, y, c)
         input_img_feature=VGG_model.predict(input_img)
         input_img_features=input_img_feature.reshape(input_img_feature.shape[0], -1)
         prediction_RF = RF_model.predict(input_img_features)[0]
         prediction_RF = le.inverse_transform([prediction_RF])  #Reverse the label encoder to original name
         print("The prediction for this image is: ", char_dict[int(prediction_RF)])
```

```
The prediction for this image is:  அ
```

Fig. 10 Output on random index value



```
In [24]: #READ EXTERNAL IMAGE...
         path = next(os.walk("C:/Users/hari/anaconda3/TamilHandwrittenRecognitionCharacter"))
         path=os.path.abspath('./sampleimages')
         data=os.path.join(path,filename)
         img=cv2.imread(data)

         plt.imshow(img)
         plt.show()
         img = cv2.resize(img, (SIZE, SIZE))
         img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
         input_img = np.expand_dims(img, axis=0) #Expand dims so the input is (num images, x, y, c)
         input_img_feature=VGG_model.predict(input_img)
         input_img_features=input_img_feature.reshape(input_img_feature.shape[0], -1)
         prediction_RF = RF_model.predict(input_img_features)[0]
         prediction_RF = le.inverse_transform([prediction_RF])  #Reverse the label encoder to original name
         print("The prediction for this image is: ", char_dict[int(prediction_RF)])
```

```
The prediction for this image is:  ஏ
```

Fig. 11 Output predicted on external images

By using the output with the External image source location, the predicted value for the input is given below:

See Fig. 11.

7 Conclusion and future work

Thus, the handwritten character recognition is a process of identifying the Tamil Vowel Character from the input, to find out the correct output using our proposed model. The paper proposed a CNN-VGG16-RF model (Convolution Neural Network-VGG16 with Random Forest) which employs an effective method to train the input handwritten image and identify the Tamil letter. The experiment for our model is done with the help of the HP lab Tamil Dataset. Our proposed model achieved an accuracy of 94.4% which is far better than other prescribe models. In the Proposed model we use only 11 Tamil characters to recognize. But, In the Future, we discuss the remaining Tamil character using the same model in a detailed manner.

Authors' contributions HM contributed on Conceptualization of proposed system and implementation. PN contributed on Mathematical Conceptualization and overall supervision. RK contributed on the Data Collection. SL contributed on the review of existing works.

Funding No funding is received.

Data availability This declaration has no repositories of data and materials.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical approval Not applicable.

References

Ahlawat, S., et al.: Improved handwritten digit recognition using convolutional neural networks (CNN). Sensors **20**(12), 1–18 (2020). https://doi.org/10.3390/s20123344

Ali, A.A.A., Mallaiah, S.: Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout. J. King Saud Univ. Comput. Inf. Sci. **34**(6), 3294–3300 (2022)

Anjul, A.T., et al.: Review of offline handwritten text recognition in south Indian languages. Malaya J. Matematik **9**(1), 751–756 (2021). https://doi.org/10.26637/MJM0901/0132

Antony Robert Raj, M., Abirami, S.: Structural Representation-Based of-Line Tamil Handwritten Character Recognition. Springer (2019). https://doi.org/10.1007/s00500-019-03978-5

Babitha Lincy, R., Gayathri, R.: Optimally Configured CNN for Tamil Handwritten Character Recognition by Improved Lion Optimization Model. Springer (2020). https://doi.org/10.1007/s11042-020-09771-z

Bhardwaj, A.: Handwritten Devanagari character recognition using deep learning-convolutional neural network (CNN) model. PalArch J. Archaeol. Egypt Egyptol. **17**(6), 7965–7984 (2020)

Danthuluri, S., et al.: Character recognition using deep learning algorithm. Turk. J. Comput. Math. Educ. **12**(14), 1165–1174 (2021)

Deepa, M.: Tamil handwritten text recognition using convolutional neural networks. Int. J. Eng. Sci. Comput. **9**(3), 20986–20988 (2019)

Eswaran, P.M., et al.: Recognizing Tamil palm-leaf manuscript characters using hybridized human perception based features. ICTACT J. Image Video Process. **44**, 2432–2440 (2021). https://doi.org/10.21917/ijivp.2021.0346

Geetha, C., Arunachalam, A.R.: Prediction of parameters of liver tumor using feature extraction and supervised function. Meas. Sensors **22**, 100386 (2022). https://doi.org/10.1016/j.measen.2022.100386

Guha, R., et al.: A hybrid swarm and gravitation-based feature selection algorithm for handwritten Indic script classification problem. Complex Intell. Syst. **7**, 823–839 (2021). https://doi.org/10.1007/s40747-020-00237-1

Gupta, D., Bag, S.: CNN-based multilingual handwritten numeral recognition: a fusion-free approach. Expert Syst. Appl. **165**, 113784 (2021). https://doi.org/10.1016/j.eswa.2020.113784

Hamdan, Y.B., Sathesh, A.: Construction of statistical SVM based recognition model for handwritten character recognition. J. Inf. Technol. Digit. World **3**(2), 92–107 (2021)

Karthikeyan, S.: Automatic detection and recognition of Tamil shop name in outdoor signboard. Int. J. Mod. Agric. **9**, 1208–1215 (2020)

Kavitha, B.R., Srimati, C.: Benchmarking on offline handwritten Tamil character recognition using convolutional neural networks. J. King Saud Univ. Comput. Inf. Sci. **34**, 1183–1190 (2019). https://doi.org/10.1016/j.jksuci.2019.06.004

Mohamed Sathik M, Spurgeon Ratheash R (2020) Optimal character segmentation for touching characters in Tamil language palm leaf manuscripts using hoover method. Int. J. Innov. Technol. Explor. Eng. **9**(6)

Mohamed Sathik, M., Spurgen Ratheash, R.: Text line segmentation in Tamil language palm leaf manuscripts—a novel approach. J. Tianjin Univ. Sci. Technol. **54**(4), 297–304 (2021)

Prabavathi, R., et al.: Prehistoric stone image Tamil character recognition using optimized DNN using Zernike moments and simplex method. Turkish J. Comput. Math. Educ. **12**(11), 5983–5591 (2021)

Preetha, S.: Machine learning for handwriting recognition. Int. J. Comput. **38**(1), 93–101 (2020)

Ram Kumar, S.: Digitalization of Tamil handwritten characters recognition using convolutional neural networks. IJARIIE **6**, 2395–4396 (2020)

Salameh, A.W., Surakhi, O.M.: An optimized convolutional neural network for handwritten digital recognition classification. J. Theor. Appl. Inf. Technol. **98**(21), 3494–3503 (2020)

Samantha Naidu, D.J., Rafi, M.: Handwritten character recognition using convolutional neural networks. Int. J. Comput. Sci. Mob. Comput. **10**(8), 41–45 (2021)

Senthil, T., et al.: An efficient CNN model with squirrel optimizer for handwritten digit recognition. Int. J. Adv. Technol. Eng. Explor. **8**(78), 2394–7454 (2021)

Shafana, M.S., et al.: An effective feature set for enhancing printed Tamil character recognition (2021). https://doi.org/10.4038/jnsfsr.v49i2.9466

Shams, M., et al.: Arabic handwritten character recognition based on CNN and SVM. Int. J. Adv. Comput. Sci. Appl. **11**, 144–149 (2020)

Sridhar, S., et al.: Character recognition using deep learning algorithm. Turkish J. Comput. Math. Educ. **12**(14), 1165–1174 (2021)

Sridharan, M., et al.: Recognition of font and Tamil letter in images using deep learning. Appl. Comput. Sci. **17**(2), 90–99 (2021). https://doi.org/10.23743/acs-2021-15

Sri Vidhya, S.R., Karthi, A.: Soft Sensor data based autonomous detection of aneurysm impacted coronary illness using machine learning algorithms. Meas. Sensors **23**, 100404 (2022). https://doi.org/10.1016/j.measen.2022.100404

Subadivya, S., et al.: Tamil-Brahmi script character recognition system using deep learning technique. Int. J. Comput. Sci. Mob. Comput. **9**(6), 114–119 (2020)

Suriya, S., et al.: Intelligent character recognition system using CNN. EAI Endors. Trans. Cloud Syst. **88**, 604–613 (2020). https://doi.org/10.4108/eai.16-10-2020.166659

Thilagavathi, G.: Tamil handwritten character recognition using artificial neural network. Int. J. Sci. Technol. Res. **8**(12), 1611–1616 (2019)

Vinotheni, C., et al.: Modified Convolutional Neural Network of Tamil Character Recognition. Springer (2021)

Authors and Affiliations

N. Pughazendi[1] · M. HariKrishnan[1] · Rashmita Khilar[2] · L. Sharmila[3]

✉ N. Pughazendi
pughazendi@gmail.com

M. HariKrishnan
harik1595@gmail.com

Rashmita Khilar
rashmita.khilar@gmail.com

L. Sharmila
shar.hariharan@gmail.com

[1] Department of CSE, Panimalar Engineering College, Chennai, Tamilnadu, India

[2] Department of CSE, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, Tamilnadu, India

[3] Department of IT, Agni College of Technology, Chennai, Tamilnadu, India