Handwritten Character Recognition of Kannada Language Using Convolutional Neural Networks and Transfer Learning

Parikshith H¹, Naga Rajath S M², Shwetha D³, Sindhu C M⁴, Ravi P⁵

¹,²,³,⁴,⁵Department of Computer Science and Engineering, Vidyavardhaka College of Engineering, Mysuru, Karnataka, India

E-mail: parikshithh@gmail.com

Abstract: Recognition of Kannada Handwritten Characters in recent times is one among the active research fields of study. It is one of the challenging topics in the pattern recognition field due vocabulary in large scale, complicated structural hierarchy and different people have different handwriting styles. In this study, we put forward a technique for recognition of characters in kannada which are handwritten based on convolutional neural networks and transfer learning. We have first pre-processed each character to remove noise, cropped each character image and resize the images. After the pre-processing the enhanced pixel values in the image helps in training of the neural network for an efficient classification of characters to their respective classes. We have considered vowels, consonants and numerals of Kannada language which are handwritten. Even though some of the characters in kannada language have similarities in structures our model is capable of classifying the character to their correct respective class. We have also used transfer learning to further improve the model so that any new style of handwritten character can be correctly predicted. Our method is efficient in neural network approach and it achieves competitive performance on the rest of the other traditional recognition methods in the literature.

Keywords: Convolutional Neural Networks, Handwritten Character Recognition, Transfer Learning, Deep learning, Handwritten Kannada Characters

1. Introduction

Handwritten Character Recognition (HCR) is the competency to recognize the character obtained from the image source and interpret the characters. The main challenge is to recognise the character because every person has a different handwriting. Sometimes a person writes a character differently than before. Handwritten texts can be overlapped, and pressure points make differences. There are many ways in which a single character can be written.

The focus of this work is to recognize the handwritten Kannada character image uploaded by users and provide output as a character in digital format. It consists of many stages where the first stage is pre-processing of data which includes removing unwanted spaces and noise from
the images, padding and resizing and converting the image to grey scale value. The data set is stored as training and testing data and are fed as input to the neural network as input. Convolutional Neural Network (CNN) model is trained using training data and validation is performed with testing data. Accuracy is improved using Transfer Learning with trained models as input and the output obtained is used for prediction. The UI allows the user to upload the image and the trained model is uploaded in the backend to obtain output in digital format.

2. Related Work

Neural network and TensorFlow techniques are used to Handwritten Character Recognition (HCR). It has a detailed procedure for recognising characters. SoftMax Regression technique is used for classification and character recognition. Diagonal and direction feature extraction methods yielded better results. Feed forward model which is trained by back-propagation algorithm is used. Normalization improved the accuracy of recognition [1].

English characters are recognised and classified using feed forward neural networks. The samples include digits, capital letters, small letters and alphanumeric character. Better results were produced when the model was applied on individual sets rather than all sets combined together [2].

Classification of cursive English script is done by two major phases segmentation and classification. Optical Character Recognition (OCR) which is used to recognize the character and to convert the image format to equivalent machine-readable format. For segmentation the projection methods such as horizontal and vertical projections are used [3].

The approach used to classify Devanagari (Hindi) is Histogram of Oriented Gradients (HOG). The whole process is broken down to segmentation and pre-processing, feature extraction and classification and recognition. Individual characters are classified using ANN. Feature extraction is performed by partitioning the image into six parts. An accuracy of 97.06% is obtained in recognising the Devanagari characters [4].

Characters are recognised using transfer learning and it is recognised even in the poor-quality image of the text. It was done by a pre-trained deep neural network using Inception V3. This model performs better with noisy character images also. It was trained on English Text samples and obtained an accuracy of 90.6% for good quality images and 78% for poor quality images.[5]

Two-stage convolutional neural network is implemented on recognise the Chinese characters which has up to a rotation angle of ±45°. It was tested on the dataset ICDAR-2013 and obtained an accuracy of 97.38%. It has 7 input channels. The characters are transformed to predict the angle of rotation represented as θ so that the characters are transformed by θ degree to obtain the correct sample. Data augmentation is applied for huge handwriting styles.[6]

Recognition of Kannada characters are performed using two techniques. Tesseract tool has an image with a character in different combinations and sent for pre-processing where the image
is converted to grey scale, denoising technique is used to remove noise and erosion is performed to reduce the thickness of the letter. CNN requires the words to be broken into each character before pre-processing where the counter of letter is formed to obtain x and y coordinates and broken down into each letter and stored in respective folders. Tesseract tool obtained an accuracy of 86% and CNN obtained an accuracy of 87%.[7]

Cursive handwritten text recognition is performed in the language Urdu. The dataset used is UNHD. There are two steps involved in the character recognition namely CNN for feature extraction and bi-directional Long-Short term memory technique for the classification. CNN uses seven layers and is followed by a B-LSTM layer. An accuracy of more than 83% overall is obtained.[8]

Character recognition is performed for Bangla language. The dataset used is Boise State Bangla Handwriting Dataset. It involved segmentation of characters and x and y coordinates marking the length and height of the character is obtained. Features are classified as zonal, pattern and gradient features. Support Vector Machine (SVM) is used for classification purposes. Accuracy is improved by One Versus One (OVO) technique. 96.42% of accuracy is obtained.[9]

Gurmukhi is an Indic language used in Punjab. It has 35 letters and 10 numerals. The dataset was collected from people who wrote the language. The grey scale conversion of the image is done before skeletonization where letters are given proper structure and normalised. The image undergoes segmentation to segregate the letters and the features are extracted. An accuracy of 98.06% is obtained from this.[10]

This paper has compared different techniques used to categorise and recognise Kannada characters and a review is made. The pre-processing is done by converting the image into a binary image, the skews are detected and corrected with an angle of θ. Segmentation is done for a line, word and characters using various techniques. Features are extracted and characters are classified and reviewed using different methods. Accuracy is improved by fusion and classifier methods.[11]

Character classification is done on the basic Kannada characters. This paper has compared the results obtained by two different classifiers namely SVM and K-NN with an accuracy obtained as 93.73% and 91.24% respectively. It is concluded that SVM has higher accuracy for basic characters. It can be further extended for all the Kannada characters. For feature analysis and computing techniques such as Crack code and Fourier descriptors are used.[12]

3. Mathematical Model

The loss function used is the Sparse categorical cross entropy which is actually used in the determination of the performance of the CNN model in this work. The categorical cross entropy formula is given as shown below (1).
This categorical cross entropy formula is based on integers, so there is no need for the conversion of targets into categorical format. The formula parameters in this case are ‘c’ which is to iterate over all classes, m is the total number of classes, \( t_{o,c} \) is the binary indicator used to indicate whether c is in the correct class, \( p_c \) is the predicted probability. The accuracy is used to evaluate the prediction, it is given by the formula as below (2).

\[
CCE(p, t) = -\sum_{c=1}^{m} t_{o,c} \log(p_{o,c})
\]  

(1)

Accuracy = \[
\frac{TP + TN}{TP + FN + FP + TN}
\]  

(2)

The accuracy formula consists of a ratio of sum of true negatives and true positives to the sum of true negatives, false positives and false negatives and true positives.

### 4. Proposed Methodology

#### 4.1 Dataset

The dataset used for this work is the Chars74K dataset. It consists of handwritten kannada character images of all vowels in which there are 16 characters and consonants in which there are 34 characters in total there are 50-character classes. The dataset also consists of handwritten kannada digits from 0 to 9. The dataset consists of images of handwritten characters which have different handwriting styles.

![Kannada Characters and Numbers](image.png)

**Figure 4.1.** Kannada Characters and Numbers

The dataset consisted of around 25 images in each class. This number of images in each class is quite less in number for the developing a model using convolutional neural networks, hence there was a need for more number of images in each character class. The below Figure 4.1 shows all the vowels, consonants and digits in digital form that are present in the dataset.
4.2 Model Architecture

This work aims at recognition of handwritten kannada characters precisely and recognised characters are obtained as output in the digital form. The flow of the development of the model involves data pre-processing which involves pre-processing techniques, then the dataset is divided into training and testing data for training and validation purposes. The Figure 4.2 provides the flowchart of system architecture.

After the dataset division in the ratio of 80:20 by shuffling, 80% of data is used for training purposes of the CNN model and rest 20% testing data was used for validation of the trained CNN model. Then the trained model is used for the transfer learning to improve the accuracy so that the model is able to recognize the characters which might have a bit different style of handwriting because transfer learning is used when there is sparse data. Then we have
developed a user interface for the user to upload a handwritten character image and get the prediction of the character in the image.

4.3 Data Preprocessing

We have used some of the data pre-processing techniques like cropping of images, removal of noise, contrast enhancement, padding, resizing and gray-scale conversion.

- Cropping - The cropping of images is done to extract a rectangular region of interest around the image, so that extra background spaces are removed from the image and create a bounding box like so only the character is given attention. The cropping is done by taking the difference between the pixel of the image and the background starting pixel.
- Noise Removal - This is used to remove the extra unwanted noise in the image, we have used gaussian blur to remove the unwanted noises.
- Contrast Enhancement - This is used to enhance the brightness between the character image and the background.
- Padding - This is used to introduce new pixels around the edges of the character image, so there will be a uniform boundary around the character for using advanced filtering techniques and it would also help in accuracy as all images that are uploaded for recognition undergoes pre-processing and helps in accurate recognition.
- Resizing - As the deep neural network models expect all the images to be of the same dimensions, we have resized the images to 208x208 pixels.
- Gray-scale Conversion - This is used to convert RGB values which are 24bits into grayscale values which is of 8bits. So this reduces the processing time as there are only 8bits.

One of the samples of the pre-processed image is as shown in the figure 4.3.

![Preprocessed image](image.png)

Figure 4.3. Preprocessed sample of a character

4.4 Image Data Augmentation

It is the technique used to artificially increase the dataset when there is sparse image data. This is helpful when only a few data samples are there for each image class. In deep learning, there will be the problem of overfitting due to lack of data samples. So, we also had few image samples in each class so we have used image augmentation to increase the data samples which were around 25 in each class to around 350 per class. This is done using the
ImageDataGenerator function in the keras library. The arguments used in the ImageDataGenerator function are as follows.

**Rotation_range**: we have used a rotation range of $15^\circ$ so that the images are randomly rotated within this range so we can get different handwriting style rotations.

**Height_shift_range**: we have used to shift the image pixels in times of height to obtain different dimensions of images so that we obtain the characters in terms of height variability to different handwriting styles.

**Shear_range**: we have used to control the shear intensity of the images and used a 0.2 as the range for it.

**Zoom_range**: we have used the zoom range by 0.1 range so that the images are zoomed in this range so that we get images with different dimensional zooms.

**Fill_mode**: We have used fill mode as nearest as our is character specific so it has to be nearest to the original images.

### 4.5 Implemented Models

The CNN models were implemented to check the accuracy obtained from the machine learning models obtained after the training and validation set is complete. We used the ReLU activation function present in the pytorch functional library to appropriately adjust weights in the neural network and to get correct prediction of results. We first implemented a small convnet with a simple stack consisting of convolution layers which are 3 in number with a ReLU activation and which is followed by a layer of max-pooling. This then has two fully connected layers. In the end we are left with fifty units and hence use softmax activation since this is a multi-class classification. Each convolution block creates a convolution kernel which is convolved with the input layer over a single spatial followed by a nonlinear activation function. The pooling layer down scales the input in both spatial dimensions. The number of filters, filter size and downscale pooling size for 1st, 2nd and 3rd layers are [32, (3,3), (2,2)], [32, (3,3), (2,2)] and [64, (3,3), (2,2)] respectively. After flattening the image i.e converting our 3D feature maps to 1D feature vectors we apply the first fully connected layer which has 64 units and activation function.

After the initial model, we decided to implement a deeper convnet. The stack for each block is as follows - we first applied zero padding i.e., adding rows and columns of zeros at the top, bottom, left and right side of an image. Then convolutions and activation functions are followed by the Batch Normalization layer. This way the activations of the previous layer at each batch are normalized i.e. mean activation is almost 0 and the activation standard deviation close to 1. After this we applied a max-pooling layer. This architecture has 5 such blocks. This is then followed by two fully connected layers. We also used dropout to avoid overfitting. In the end we are left with fifty units and hence use softmax activation since this is a multi-class classification. Kernel size for zero padding, filter size, downscale size in each block is the same.
and is (1,1), (3,3) and (2,2) respectively. The number of filters for 1st, 2nd and 3rd layers are 128, 256 and 512 respectively. After flattening the image i.e. converting our 3D feature maps to 1D feature vectors we apply the first fully connected layer which has 64 units and activation function. Finally, we apply relu activation to the last unit to classify into respective class labels. For this CNN model we have achieved the training accuracy of 0.89 and validation accuracy of 0.8692. And for the same model with the same layers of deep network we have trained it for kannada handwritten numerals for which we have achieved a validation accuracy of 0.9670.

The parameters of a neural network are typically the weights of the connections. In this case, during the training stage the learning of the parameters takes place. So, the algorithm itself (and the input data) tunes these parameters. The hyper parameters are typically the learning rate, the batch size or the number of epochs. The model summary is as follows in Table 1.

| Layer (type) | Output Shape   | Param         |
|--------------|----------------|---------------|
| Conv2d-1     | [-1,64,96,96]  | 1,664         |
| Conv2d-2     | [-1,128,44,44] | 204,928       |
| Conv2d-3     | [-1,256,18,18] | 819,456       |
| Linear-4     | [-1,512]       | 10,617,344    |
| Linear-5     | [-1,256]       | 131,328       |
| Linear-6     | [-1,128]       | 32,896        |
| Linear-7     | [-1,50]        | 6,450         |
| Total params: 11,814,066 |
| Trainable params: 11,814,066 |

4.6 The Transfer Learning Model

For our transfer learning implementation, we have used the VGG-16 which is a convolutional neural network model. Fully the layers are connected with activation function ReLU, shape = (n_inputs, 256), Dropout with 0.2 chance of dropping, the output is fully connected with log softmax, shape = (256, n_classes). To the model when the extra layers are added, they are set to be trainable by default (require_grad=True). For the VGG-16, the last fully connected layer is only changed. The first fully connected 5 layers and all of the weights that are in the convolutional layers are not trainable. The final outputs from the VGG-16 network are log probabilities for each of the 50 classes in our kannada character dataset.
Figure 4.4. VGG-16 Model Architecture

The above Figure 4.4 shows the architecture for the VGG16 model. It is significant to observe that input dimensions for this model were 100x100 instead of 208x208 as shown in the figure. The architecture for this model is as follows: Each block is stacked with convolution layers and at the end of every block a pooling layer to reduce spatial dimension is applied. The output dimensions are increased by a factor of 2 every block. It should also be noted that zero padding is applied before every convolution layer in each block since it is not shown in the figure. The last three fully connected layers were removed as discussed above. Just as a mere experiment, we also decided to implement the VGG16 model architecture from scratch including the fully connected layer to see how it performed. The accuracy after the transfer learning we have achieved is for training we got an accuracy of 0.919 and for validation we have got accuracy of 0.9306.

5. Experiment and Result

We evaluate our models based on the accuracy and loss. We try to minimize the loss function in all the models to achieve better accuracy. Overall, we get a whole simulated model to recognize the handwritten characters. We ran tests for the machine learning model built using ReLU, softmax activation function and the transfer learning model and obtained results for the following measures on validation samples as shown in the table below.

| Kannada Characters                  | ML Model          | Accuracy | Loss    |
|------------------------------------|-------------------|----------|---------|
| Kannada Alphabets (Vowels and Consonants) | CNN Model        | 0.8692   | 0.4992  |
|                                    | Transfer Learning Model | 0.9306 | 0.1591  |
| Kannada Numerals                   | CNN Model        | 0.9670   | 0.1374  |
The graph is plotted for accuracy and loss shown by each model. The accuracy v/s epoch graph has accuracy values on y-axis and epoch values on x-axis. Similarly, for loss v/s epoch graph, loss is plotted on y-axis and epoch on x-axis. It provides an overview of the model behaviour for each epoch run on the model and their accuracy rate.

5.1 Graph Obtained on Training of CNN Model

As shown in Figure 5.1 we can observe that the training accuracy is maintained almost constant between 0.87 and 0.89 with a slight variation. In training loss, we can see a constant decrease.

5.2 Graph Obtained on Validation of CNN Model

As shown in Figure 5.2 we can observe that the validation accuracy increases steadily with slight variation in the beginning and we achieve accuracy of 0.8692. The validation loss decreases as the number of epoch’s increases but with variations.
5.3 Graph Obtained on Training of Pre-trained Model using Transfer Learning

As shown in Figure 5.3 the training accuracy increases and achieves accuracy of 0.9167 for 20 epochs. The training loss decreases, and we can see a constant decrease.

5.4 Graph Obtained on Validation of Pre-trained Model using Transfer Learning

As shown in the above Figure 5.4 the validation accuracy increases at a constant level and achieves an accuracy of 0.9306 for 20 epochs, while the validation loss also decreases as the epochs increase.

We have developed a user interface so that the user can upload the handwritten character image for prediction. The user will get the predicted character output in digital form and also a
similarity graph of the character predicted as some characters will have some similarity in the strokes.

The Figure 5.5 below shows the user interface and the predicted character output along with the similarity graph.

Figure 5.5. User interface of the predicted character along with similarity graph

6. Conclusion

Handwritten kannada character recognition is one of the laborious tasks since there are numerous variations in the styles of writing of every person and also relies on the complication of the languages that are handwritten which needs to be recognized. In this paper we have explored the two main methods used to recognize character from the images consisting of handwritten characters i.e Convolutional Neural Network (CNN) and transfer learning. Before training the CNN model, various pre-processing techniques are applied which includes cropping, padding, denoising and resizing of the image. Since we had a small dataset, the technique called image data augmentation is used to increase the samples in the dataset. Then the augmented images are fed for training CNN model which has both fully connected and convolutional layers and learns the mapping of character image to the respective character label while training. We achieved 86.92% validation accuracy after training the developed CNN model. To develop a good handwritten character recognition system, we had to increase and fine tune the accuracy obtained on training the CNN model. So, we applied a fine-tuning method called transfer learning in which the obtained CNN model is fed and we used VGG16 as the pre-trained network. After fine tuning we achieved a validation accuracy of 93.06% which was much better than the accuracy obtained on training CNN. And we have achieved a validation accuracy of 96.7% for the handwritten kannada numerals recognition.
References

[1] Megha Agarwal, Shalika, Vinam Tomar and Priyanka Gupta, “Handwritten Character Recognition using Neural Network and TensorFlow”, *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 2019.

[2] Shyla Afroge, Boshir Ahmed and Firoz Mahmud, “Optical character recognition using back propagation neural network”, *2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)*, Rajshahi, Bangladesh, 2016.

[3] Pritam Dhande and Reena Kharat, “Recognition of cursive English handwritten characters”, *2017 International Conference on Trends in Electronics and Informatics (ITEI)*, Tirunelveli, India, 2017.

[4] Nikita Singh, “An Efficient Approach for Handwritten Devanagari Character Recognition based on Artificial Neural Network”, *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida, India, 2018.

[5] Tan Chiang Wei, U.U.Sheikh and Ab Al-Hadi Ab Rahman, “Improved Optical Character Recognition with Deep Neural Network”, *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*, Batu Ferringhi, Malaysia, 2018.

[6] Zhe Li, Lianwen Jin, Songxuan Lai, “Rotation-Free Online Handwritten Chinese Character Recognition using Two-Stage Convolutional Neural Network”, *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Niagara Falls, NY, USA, 2018.

[7] Roshan Fernandes, Anisha P Rodrigues, “Kannada Handwritten Script Recognition using Machine Learning Techniques”, *2019 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, Manipal, India, 2019.

[8] Shahbaz Hassan, Ayesha Irfan, Ali Mirza, Imran Siddiqi, “Cursive Handwritten Text Recognition Using Bi-Directional LSTM: A Case Study on Urdu Handwriting”, *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*, Istanbul, Turkey, Turkey, 2019.

[9] Nishatul Majid, Elisa H. Barney Smith, “Introducing the Boise State Bangla Handwriting Dataset and an Efficient Offline Recognizer of Isolated Bangla Characters”, *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Niagara Falls, NY, USA, 2018.

[10] Manoj Kumar Mahto, Karamjit Bhatia, R. K. Sharma, “Combined horizontal and vertical projection feature extraction technique for Gurmukhi handwritten character recognition”, *2015 International Conference on Advances in Computer Engineering and Applications*, Ghaziabad, India, 2015.
[11] K Indira, S Sethu Selvi, “Kannada Character Recognition System: A Review”, *InterJRI Science and Technology, Vol. 1, Issue 2*, July 2009.

[12] Ganapatsingh G Rajput, Rajeshwari Horakeri, “Handwritten Kannada Vowel Character Recognition Using Crack Codes and Fourier Descriptors”, December 2011.