DESIGN OF HANDWRITING RECOGNITION SYSTEM FOR KANNADA CHARACTERS USING TENSORFLOW

[1]Asha K, [2]Kiran Kumar Gowda R, [3]Diksha M M, [4]Soniya B S
[1]Assistant Professor Dept. of IS&E, GMIT, Davanagere, INDIA
ashak@gmit.ac.in
[2]Student [4GM18IS025] Dept. of IS&E, GMIT, Davanagere, INDIA
kirangowdareigns1@gmail.com
[3]Student [4GM18IS014], Dept. of IS&E, GMIT, Davanagere,INDIA
dikshamm2000@gmail.com
[4]Student [4GM18IS052]
Dept. of IS&E, GMIT, Davanagere, INDIA
soniyabs98642@gmail.com

Abstract— Most of the literatures and other historical data are in the form of handwritten documents. Such documents need to be digitized and preserved for future generation to transfer the knowledge about our culture and history. Development of Handwriting recognition system plays a vital role in digitizing the handwritten document. This paper provides technical details for the design of Handwriting recognition system for Kannada. The steps required for the development of handwriting recognition engine are discussed using Tensorflow - An open source machine learning framework.

Keywords— Tensorflow, handwriting recognition, Offline handwriting recognition, Kannada ocr, Pattern recognition, deep learning.

I. INTRODUCTION

Handwritten character recognition has received greater attention in academic and research fields. In this regard much research has been done for international languages like English, but least work is done for South Indian languages because of hardware limitation to accept large character set of Indian scripts. Hence, handwriting recognition system for South Indian languages is in demand. Majority of the people in South India speak one of the four major Dravidian languages: Telugu, Tamil, Kannada and Malayalam.

Kannada is the official language of the southern Indian state of Karnataka. Kannada is a Dravidian language spoken by about 44 million people in the Indian states of Karnataka, Andhra Pradesh, Tamil Nadu and Maharashtra. The Kannada alphabets were developed from the Kadamba and Chalukya scripts, descendents of Brahmi which were used between the 5th and 7th centuries AD. The Kannada character set considered for recognition in this paper has 16 vowels and 35 consonants as shown in Figure 1.

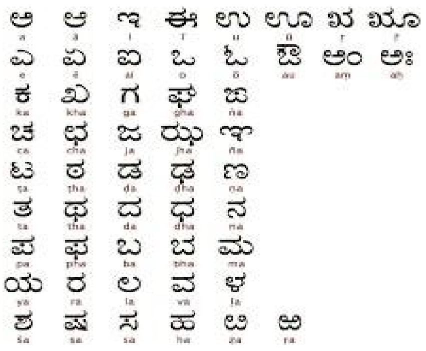
Figure 1 Basic Kannada Character Set

II. HANDWRITING RECOGNITION SYSTEM

Generalized steps involved in handwriting recognition system are data collection, pre-processing, feature extraction and classification as shown in Figure 1.

Step 1: Data acquisition

This is the first step in handwriting recognition system. In this step data is in the form of image consisting of handwritten information.

Step 2: Pre processing

The accuracy of any handwriting recognition system can be improved by applying proper pre-processing techniques. There are many pre-processing techniques like Binarization, noise elimination, slant correction, size normalization, smoothing or thinning, re-sampling etc. These pre-processing techniques are system dependent and basically pre-processing techniques are used to present the clear input to the recognition system by removing noise or distortion present in the input.

Binarization

This step is necessary because the captured /scanned image may be having data/information in different colours. The idea behind binarization is to separate the foreground and background information by setting the background to white.

Noise removal

The accuracy of handwriting recognition system much depends on the clarity of the input image. Most of the time additional noise may be present in the input image either because of the quality of capturing device or aging of image or external dust. This additional noise is called impulse noise or salt and pepper noise, by applying filtering techniques like median filter or morphological filter or Dvadasham (Dodeca) Edge Filter (DEF) [1]. Basically in median filtering, the value of each pixel is replaced by the median of pixel values in the specified dimension of neighbourhood.

Slant correction

Slant correction is a process of detecting and correcting the skew by estimating the angle at which an input image is rotated during the data acquisition. Some of the techniques for slant correction are Projection profile method, run length based technique, Hough transform, extrema method, generalized chain code estimator etc [2].

Size normalization

This technique is applied to bring uneven sized character present in the input to a predefined size to make it even with respect to all the characters present in the input. Size normalization can be done by comparing the input stroke border frame with assumed fixed size frame.

Smoothing or thinning

Thinning is applied to have characters with single pixel thickness by removing the flickers which may exist in handwritten data because of handwritten style and the hardware used. Thinning can be done by modifying each pixel value with mean value of k-neighbours and the angle subtended at position from each end.

Step 3: Feature extraction

After normalizing the data, by applying pre-processing techniques, the most challenging part in recognition phase is segmentation. Segmentation is not required for isolated handwritten character recognition but it is very important in case of document or word recognition. Handwritten document recognition system involves line segmentation which separates each line of text followed by word segmentation which separates each word in a line [3] as shown in Figure 2, Figure 3 and Figure 4.

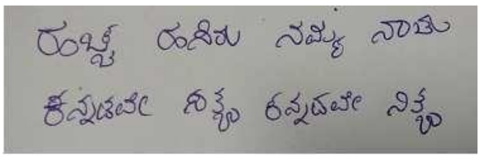

Figure 2.Before Segmentation

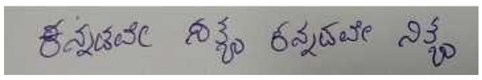

Figure 3.Line Segmentation

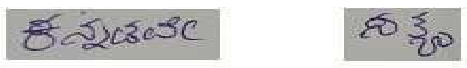

Figure 4.Word Segmentation

It is very important to identify and extract unique and correct features from character set of a particular language to maximize the recognition rate with the least amount of elements. Features used to represent a character are language dependent hence the method that gives better result for a particular script cannot be applied for other scripts. The character set of South Indian languages contains wide variety of structural features like loops, crossings, headline, straight line, dots etc. and statistical features like zoning, projection and profiles.

Step 4: Classification

Classification is the last and the most important step which carry out some form of comparison between a given unknown handwriting pattern to reference handwriting patterns to assign one of the references to the unknown one.

III. RELATED WORK

Arwa Mohammed Taqi et al., [4] presented a paper "The Impact of Multi-optimizers and Data Augmentation on TensorFlow Convolutional Neural Network Performance". Four different optimizers were used with the TF-CNN, Adagrad, ProximalAdagrad, Adam, and RMSProp to achieve accurate classification. The classification accuracy using Adam optimizer was 95.8%, and it reaches 100% when using RMSProp optimizer.

Reena Dhakad and Dinesh Soni [6] proposed Devanagari Digit Recognition by using Artificial Neural Network. The Support Vector Machine (SVM) and k-NN categoryifiers were used with one-against-rest class model. Neural network classifier has obtained highest result accuracy of 93.21%.

Anirudh Ganesh et al., [6] proposed Deep Learning Approach for Recognition of Handwritten Kannada Numerals. Authors implemented the LeNet model for Convolutional Neural Networks and models for the Deep Belief Network and explored the performance of Convolutional Neural Networks, and Deep Belief Networks in the classification of Handwritten Kannada numerals as obtained from the Chars74 k dataset. They have obtained an accuracy of around 97.76 percent using CNNs and a similar accuracy of 98.14 percent in case of DBNs and conclude that the convolutional neural networks converge faster than deep belief networks.

Keerthi Prasad et al., [7] proposed online Kannada handwritten character recognition for mobile devices. The proposed system was implemented on mobile devices for Kannada vowels and consonants using Principal component analysis (PCA) and dynamic time wrapping (DTW) techniques and they obtained an accuracy of 88% for PCA and 64% for DTW approach.

S.A Angadi and Sharanabasavaraj H.Angadi [8] proposed a method that uses structural features and Support Vector Machine (SVM) classifier for recognition of handwritten Kannada characters. They obtained recognition accuracy of 89.84% and 85.14% for handwritten Kannada vowels and consonants.

Anitha Mary M.O et al., [9] proposed a method which uses the combination of Chain Code Histogram and Differential Chain Code Histogram based features for recognition of isolated basic Malayalam characters. They obtained an accuracy of 92.75% using neural network classifier. Authors [10] have also proposed a novel method for the isolated Malayalam character recognition based on the combination of global and local features. Global features include moment invariants and projection features and gradient features of the characters are considered as local features. Proposed method achieves an accuracy of 96.16% recognition using a two layer feed forward neural network as a classifier.

IV. TENSOR FLOW

TensorFlow is a powerful data flow oriented machine learning library created the Brain Team of Google and made open source in 2015. It is designed to be easy to use and widely applicable on both numeric and neural network oriented problems as well as other domains. It can be thought of as a programming system in which you represent computations as graphs. Nodes in the graph represent math operations, and the edges represent multidimensional data arrays (tensors) communicated between them.

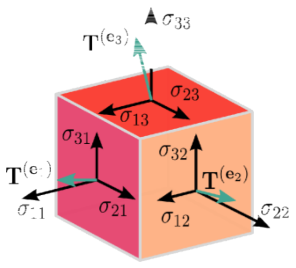

Figure 4.Tensor

Tensors are higher dimensional arrays, as represented in Figure 5, which are used in computer programming to represent a multitude of data in the form of numbers. There are other n-d array libraries available on the internet like Numpy but tensorflow stands apart from them as it offers methods to create tensor functions and automatically compute derivatives.

Generalized TensorFlow model to create handwriting recognition engine is shown in Figure 5.

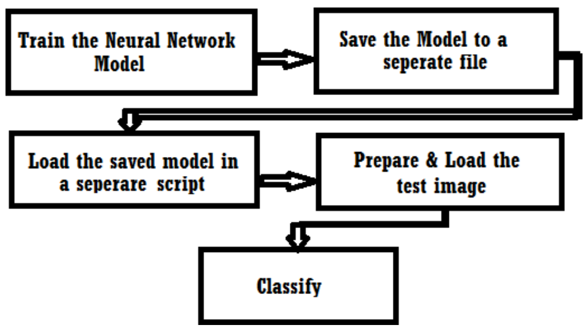
Train the Neural Network Model
Save the Model to a seperate file
Load the saved model in a seperare script
Prepare & Load the test image
Classify

Figure 5.Tensorflow Model

Training Neural Network: Collect and load the dataset for training with any of the Deep learning architectures such as convolutional neural networks, deep belief networks and recurrent neural networks. There are many dataset available online for research purpose such as MNIST, Chars74K and many more.

For example, training the CNN with MNIST dataset is shown below. We know that CNN consists of different layers namely input layer, convolutional layer, pooling layer, dense layer and output layer. Configuration of these layers using tensorflow is as below.

# **Imports**

import numpy as np import tensorflow as tf

# **Input Layer**

input_layer = tf.reshape(features["x"], [-1, 28, 28, 1])

The parameters are [batch_size, image_height, image_width, channels]. MNIST dataset is composed of monochrome 28x28 pixel images, so the desired shape for our input layer is [batch_size, 28, 28, 1].

Convolutional Layer #1 conv1 = tf.layers.conv2d(inputs=input_layer,

filters=32, kernel_size=[5, 5], padding="same", activation=tf.nn.relu)

In our first convolutional layer, we want to apply 32 5x5 filters to the input layer, with a ReLU activation function.

Pooling Layer #1

pool1 =

tf.layers.max_pooling2d(inputs=conv1, pool_size=[2, 2], strides=2)

Construct a layer that performs max pooling with a 2x2 filter and stride of 2.

Convolutional Layer #2 and Pooling Layer #2

conv2 = tf.layers.conv2d(inputs=pool1,

filters=64, kernel_size=[5, 5], padding="same", activation=tf.nn.relu)

pool2=tf.layers.max_pooling2d(inputs=conv2 , pool_size=[2, 2], strides=2)

Convolutional layer #2 has a shape of [batch_size, 14, 14, 64], the same height and width as pool , and 64 channels for the 64 filters applied.

Dense Layer

pool2_flat = tf.reshape(pool2, [-1, 7 * 7 * 64])

dense = tf.layers.dense(inputs=pool2_flat, units=1024, activation=tf.nn.relu)

dropout = tf.layers.dropout (inputs=dense, rate=0.4, training=mode

== tf.estimator.ModeKeys.TRAIN)

Next, we want to add a dense layer (with 1,024 neurons and ReLU activation) to our CNN to perform classification on the features extracted by the convolution/pooling layers.

Logits Layer

logits = tf.layers.dense(inputs=dropout, units=10)

The final layer in our neural network is the logits layer, which will return the raw values for our predictions. Our final output tensor of the CNN, logits, has shape [batch_size, 10].

Generate Predictions & Probabilities

The predicted class for each example: a digit from 0–9.

The probabilities for each possible target class for each example: the probability that the example is a 0, is a 1, is a 2, etc.

Tensorflow has two functions for prediction & probabilities. tf.argmax(input=logits, axis=1)

tf.nn.softmax(logits, name="softmax_tensor")

Calculate Loss

We need to define a loss function that measures how closely the model's predictions match the target classes.

# Calculate Loss (for both TRAIN and EVAL modes) loss=
tf.losses.sparse_softmax_cross_entropy(labels =labels, logits=logits)

Configure the Training Op

Let's configure our model to optimize the loss value of previous section during training. We'll use a learning rate of and stochastic gradient descent as the optimization
algorithm.

if mode == tf.estimator.ModeKeys.TRAIN: optimizer=
tf.train.GradientDescentOptimizer(learning_rate=0.001) train_op = optimizer.minimize(
loss=loss,
global_step=tf.train.get_global_step())
return
tf.estimator.EstimatorSpec(mode=mode, loss=loss, train_op=train_op)

Training the CNN MNIST Classifier

# Load training and eval data
mnist =
tf.contrib.learn.datasets.load_dataset("mnist") train_data = mnist.train.images # Returns np.array train_labels =
np.asarray(mnist.train.labels,
dtype=np.int32)

eval_data = mnist.test.images # Returns np.array eval_labels =
np.asarray(mnist.test.labels, dtype=np.int32)

Now we're ready to train our model, which we can do by creating train_input_fn and calling train() on mnist_classifier.

# Train the model
train_input_fn =
tf.estimator.inputs.numpy_input_fn(x={"x": train_data},
y=train_labels, batch_size=100, num_epochs=None, shuffle=True)
mnist_classifier.train(
input_fn=train_input_fn, steps=20000, hooks=[logging_hook])

After training the selected model save it to a separate file and load the same on to new python script for evaluation with test input.

V. CONCLUSION

Handwriting recognition system for international languages like English is commercially available and much work need to be carried out for South Indian languages like Kannada, Tamil, Telugu and Malayalam. This paper has explored the design for the implementation of handwriting recognition system using tensorflow which is very useful for the research in the field of handwriting recognition of Indian scripts.

REFERENCES

[1] Naveen R Chanukotimath, Feroz Khan, Keerthi Prasad G, Imran Khan, Deepak D J, Nasreen Taj M B, "Dvadasham (Dodeca) Edge Filter for Impulse Noise, Gaussian Noise, Quantum Noise Reduction in Images", Compusoft, An International Journal of advanced computer technology, July 2013, Volume-II.

[2] Nazia Makkar and Sukhjit Singh, "A Brief tour to various skew detection and correction techniques", IJSETT, 2012, pp 54-58.

[3] A. Sushma and Veena G.S "Kannada handwritten word conversion to electronic textual format using HMM model" International conference on CSISS 2016, pp 330-335.

[4] Arwa Mohammed Taqi, Ahmed Awad, Fadwa Al-Azzo and Mariofanna Milanova, "The Impact of Multi-optimizers and Data Augmentation on TensorFlow Convolutional Neural Network Performance", In Proc. IEEE Conference on Multimedia Information Processing and Retrieval, 2018.

[5] Reena Dhakad and Dinesh Soni. "Devanagari Digit Recognition by using Artificial Neural Network", In Proc. ICECDS-2017.

[6] Anirudh Ganesh, Ashwin R. Jadhav, K.A. Cibi Pragadeesh, "Deep Learning Approach for Recognition of Handwritten Kannada Numerals", Springer International Publishing AG 2018, Proceedings of the Eighth International Conference on Soft Computing and Pattern Recognition.

[7] Keerthi Prasad, Imran Khan, Naveen R Chanukotimath and Firoz Khan. "On-line Handwritten Character Recognition System for Kannada using Principal Component Analysis approach", In Proc. WICT-12, Trivendram, India.

[8] S.A Angadi and Sharanabasavaraj H.Angadi. “Structural Features for Recognition of Hand Written Kannada Character based on SVM”. International Journal of Computer Science, Engineering and Information Technology, Vol. 5, No. 2, April 2015.

[9] Anitha Mary M.O. Chacko, Dhanya P.M, “Combining Classifiers for Offline Malayalam Character Recognition”, Emerging ICT for bridging the future, Vol. 2, Springer International Publishing, Switzerland 2015.

[10]Anitha Mary M.O. Chacko, Dhanya P.M. “A differential chain code histogram based approach for offline Malayalam character recognition”, International conference on communication and computing, pp. 134-139, 2014.