Handwritten Devanagari Character Recognition

Kaustav Maity

July 2020

Abstract

This project is about building a system that can recognize handwritten devanagari characters using Machine Learning and Deep Learning techniques.

1 Introduction

Machine Learning, Deep Learning and Artificial Intelligence has empowered the current society with automated and intelligent systems that has helped to serve human purposes.Over the last few decades human beings are evolving more faster than ever before because of the digital revolution. Mobile Phones, Laptops and other digital devices has became as our part of daily life. And because of internet we have access to enormous amount of data around the world. So, how Machine Learning, Deep Learning comes into picture? In our mobile phone we have Finger Print scanner, Face Recognition system that serves the security purpose. Search Engines gives us useful results, people can see auto generated subtitles while watching videos, search through voice search, translation of language, recommender systems all these things rely on Machine learning. This is also one of the most emerging field. There are lot of things to discovered from this community. This project is also a use case of Machine Learning and Deep Learning techniques where I have built a system that can recognize Handwritten Devanagari characters.

2 About Devanagari

Devanagari is Indian subcontinent based language which was developed in 1st to 4th century and was in regular use by the 7th century. Devanagari script composed of 47 primary characters including 14 vowels and 33 consonants,is the fourth most widely adopted writing system in the world,being used for over 120 languages.Devanagari is a compound of 'deva' and 'nagari'. Deva means 'Heavenly or Divine' and is also one of the terms for deity in Hinduism. Nagari comes from the word 'Nagaram' which means 'city' so, all together 'Devanagari' means City of Divinity. Devanagari is part of the Brahmic Family of scripts of India, Nepal, Tibet and South east Asia. Some of the earliest epigraphical evidence attesting to the developing Sanskrit Nagari script in ancient India, in a form similar to Devanagari , is from the 1st to 4th century CE inscriptions discovered in Gujrat. 'Bhagavat Gita' is one of the Novel Scripture that was originally written in Devanagari.

3 Devanagari Characters

3.1 Letters

Occlusives

	Voiceless plosives				Voiced plosives				Nasals	
	unaspirated		aspirated		unaspirated		aspirated			
Velar	क	ka	ख	kha	ग	ga	घ	gha	ङ	ṅa
Palatal	च	ca	छ	cha	ज	ja	झ	jha	ञ	ña
Retroflex	ट	ṭa	ठ	ṭha	ड	ḍa	ढ	ḍha	ण	ṇa
Dental	त	ta	थ	tha	द	da	ध	dha	न	na
Labial	प	pa	फ	pha	ब	ba	भ	bha	म	ma

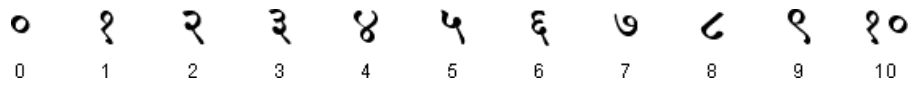
Sonorants and fricatives

	Palatal		Retroflex		Dental		Labial	
Sonorants	य	ya	र	ra	ल	la	व	va
Sibilants	श	śa	ष	ṣa	स	sa		

Other letters

ह ha ळ ḷa

3.2 Numerals



4 Dataset

Two datasets are used for this particular project. First dataset is used for training and testing and the second dataset is used only for test purposes manually, like how the system is performing in a completely different dataset.

4.1 Description

The First dataset contains images of handwritten devanagari characters of 33 consonants, 3 special characters and 10 numerals. It does not include any vowels all together there are 46 classes.There are 1700 examples of each characters. Each image of this dataset has size 32*32*3 are gray scale image with 3 channels.The background of the images are black and the characters are written in black. Labels for the images are like 'character_1_ka','charcter_2_kha','digit_0','digit_9' etc. The dataset can be found here.

As I mentioned earlier the Second dataset is used for completely testing purpose to see how the algorithm is doing. Unfortunately this dataset has a completely different set of labels, because of that I have not measured accuracy of the system in this dataset. But I have checked manually how my algorithm is working on a completely unknown dataset. We will see the interesting results as we go along. This dataset contains vowels,consonants,numeral and some special characters. We have discarded all the vowels for our purpose. This dataset contains almost 12000 examples in total. Each images have the size of 28*28*3 are gray scale image with 3 channels. But the background here is white and the characters written inside are in black. The images are being transformed with black background and white face by doing a binary not operation as a step of data preprocessing and also the images are being resized to fit the algorithm. The dataset can be found here.

4.2 Training Examples

Figure 1: Images from the training dataset

5 Model Building

We have used Logistic Regression and Random Forest Classifier Machine Learning models and used Artificial Neural Network and Convolution Neural Network as our Deep Learning model. We will see how the models have performed.

5.1 Building our Machine Learning Models

Here we have basically used two machine learning algorithm for this dataset. Logistic Regression and Random Forest Classifier. We have partitioned the total 78200 examples of the first dataset into 80% and 20% for training and testing purpose respectively.

We achieved almost 73% test accuracy in Logistic Regression and 90% test accuracy in Random Forest Classifier. Here is a pictorial representation of performance.

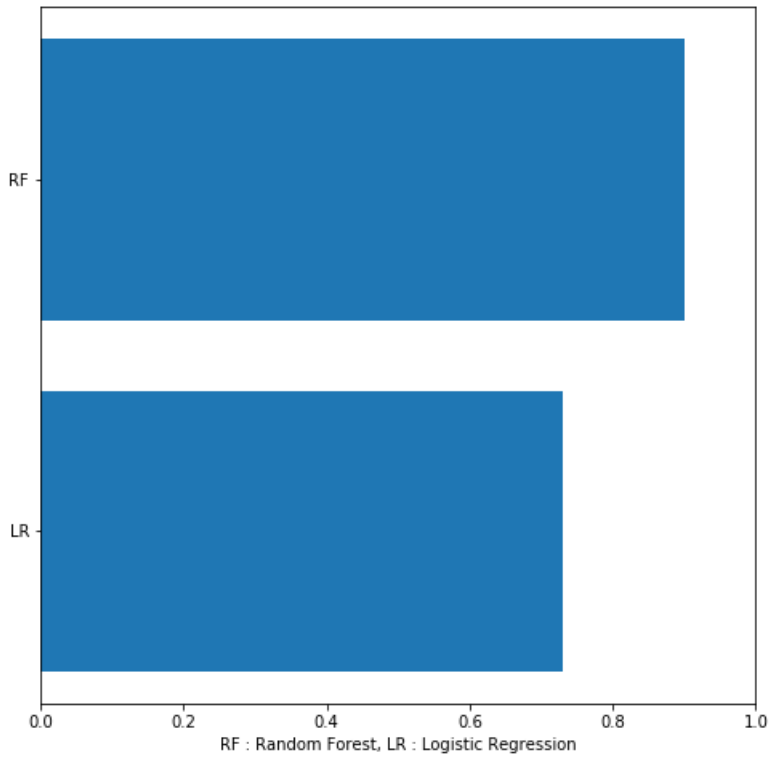


Figure 2: Horizontal Bar Plot of the accuracies

5.2 Building Deep Learning Models

We have built Artificial Neural Network(ANN) and Convolution Neural Network(CNN) models for classifying the handwritten character images. Again here we have partitioned the first dataset into 80% training and 20% testing dataset. And further we have partitioned the train data set into 90% train and 10% validation set.

5.2.1 Building ANN model

Our Artificial Neural Network(ANN) model consists of one input, three hidden and one output layer. First of all we have flatten the pixel values into a vector. There are total $32 \cdot 32 \cdot 3 = 3072$ pixels in the image so, there are 3072 units in the input layer. In the first hidden layer there are 512 units connected with a fully connected layer. We applied Batch Normalization on the units of the first hidden layer. In the second hidden layer there are 256 units. Batch Normalization applied on the units of the second hidden layer and then we have randomly dropped some units of the second layer with probability 0.2. Third hidden layer has 128 hidden units and a Batch Normalization layer is added after that and randomly some cells are dropped with probability 0.3. ReLu activation function is used for each hidden layer. In the output layer there are 46 softmax units as we have 46 classes all total. Output layer gives us some probability for an example to belong a particular class.

Sparse Categorical Cross Entropy is used as the loss function here and Adam optimizer is used to minimize the loss function. The model is trained for 20 epochs.

The Cross Entropy loss function is given by,

$$L_n = -\sum_{i=1}^{k} y_i^{(n)} \ln y_i^{*(n)}$$

$$L = \frac{1}{N} \sum_{n=1}^{N} L_n$$

ANN architecture

Here is a pictorial representation of our ANN model.

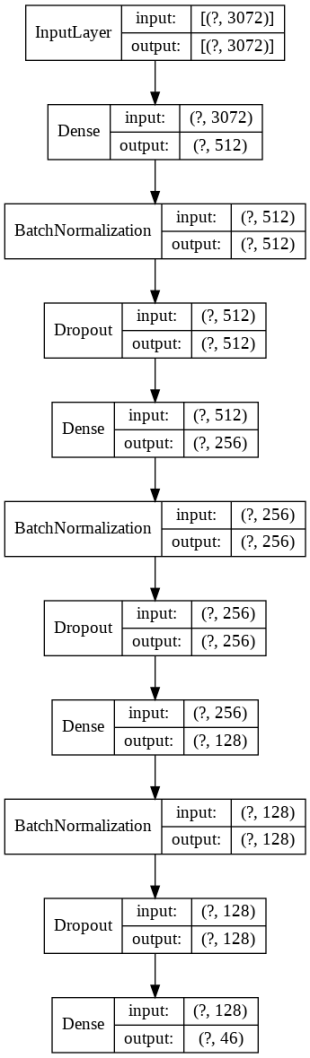


Figure 3: ANN architecture

Learning Curves

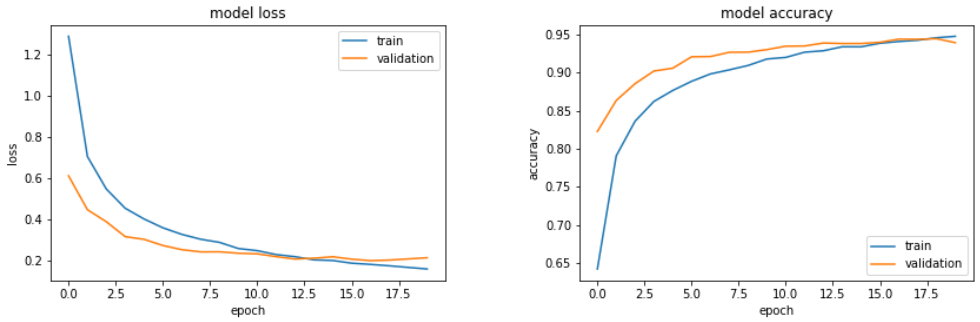


Figure 4: Learning Curves of ANN model

Result

After 20 epochs the model has achieved train loss 0.1621 and validation accuracy 0.2166 with train accuracy 0.9476 and validation accuracy 0.9393. In the test dataset the model has achieved loss 0.2141 and test accuracy 0.9445.

So from the above figures we can see that the model achieves optimum accuracy at 15 epochs. After that we can intuitively say it will overfit.

5.2.2 Building CNN Model

Our CNN model has 3 convolution layer followed by one flatten layer, 2 dense layers and one final output layer. In first convolution layer there are 32 filters of kenel size 3*3. A pooling layer is added after the convolution layer of size 2*2. After that there is second convolution layer of size 3*3 with 64 filters, a pooling layer of size 2*2 and dropout layer with 0.2 probability. Then there is third convolution layer of size 5*5 with 128 filters, a pooling layer of size 2*2 and dropout layer with drop probability 0.4. The output we get from the third convolution layer is being flattened as a vector. After that we added two dense layer following by a Batch Normalization and dropout layer. In first dense layer there are 128 hidden units and in the second dense layer there are 64 hidden units. No units are dropped in first dense layer and in the second dense layer the units are dropped with probability 0.3. ReLu

activation function is used on every layer in this model. In the output layer there are 46 softmax units as before.

CNN Architecture

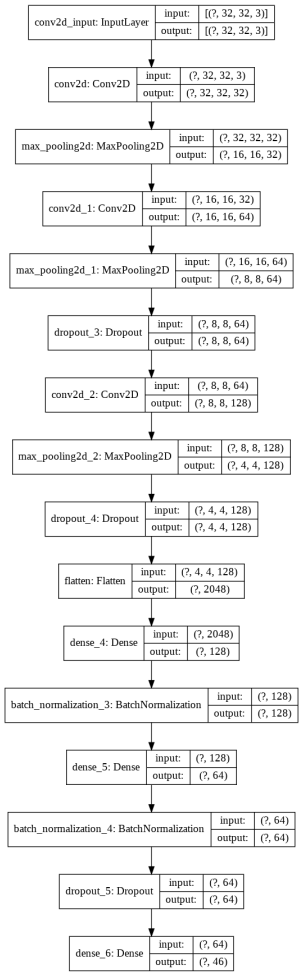


Figure 5: CNN model architecture

As before here also Sparse Categorical Cross Entropy is used as loss function and Adam optimizer is used to minimize the loss function. Model is trained for 20 epochs.

Learning Curves

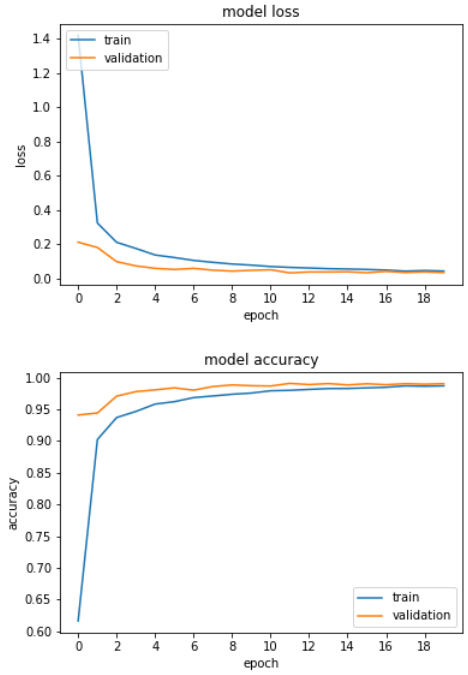


Figure 6: Learning Curves of CNN model

5.2.3 Further Analysis with CNN model

We have seen that how CNN has outperformed over the remaining algorithms. One simple reason behind that is CNN can extract features from an image by using different set of filters.

Now in the test dataset the test accuracy 0.99. Hence there are still 1% images which are being misclassified. All total 170 images are misclassified in the test dataset. When we deep dive into the reason behind this we found that the images which are being misclassified are actually very unclear. It is not possible for a human to tell what the image is actually. This is happening because some of the characters in Devanagari looks almost similar. In the next section we will see the images that are being misclassified.

Misclassified Test Images

Figure 7: Misclassified Images

If we see carefully at the misclassified images we could clearly see that the misclassification is happening due similar appearance of some of the characters. For example 'character_4_gha' and 'character_17_tha' looks similar so, 'character_4_gha' is misclassified as 'character_17_tha' in some cases or vice-versa.

Fixing the issue in Application

We can see that the images which are misclassified are actually very unclear and it is hard for a human also to recognize these images so, it is almost impossible to categorize these images for a machine. But we can fix this issue when we will use our algorithm in further application. Ultimately we will use this algorithm to read devanagari scripts.So there we can make use of additional information to predict an image of particular character. We will then use a RNN to predict, the algorithm will take the image of a character and surrounding characters(context) to predict that particular character. We can make use of bidirectional RNN also.In this way my might get an almost error free system.

5.2.4 Prediction in images of Second Dataset

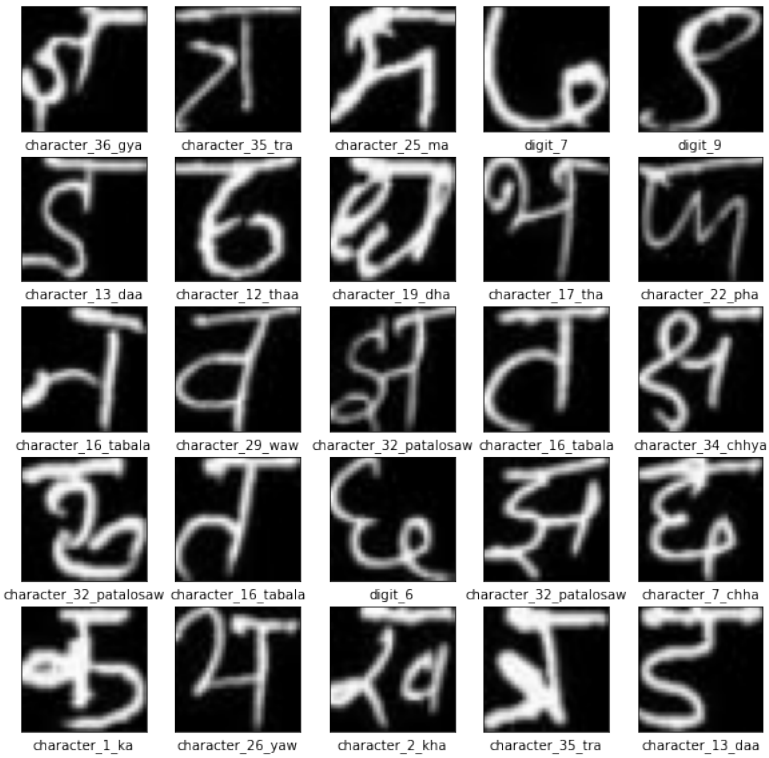


Figure 8: Predictions in Second dataset

If we see carefully the images and compare with the devanagari character we will find that our algorithm is doing very well in this dataset also. So, we have successfully built a system that can recognize devanagari characters properly.

6 Further Works & Applications

Though this is a small effort but we have to think about the Bigger picture. When we talk about Machine Learning and Deep Learning techniques we should always keep in mind that our work should add some values to the society. A lot of further works to be done. Our objective is to first make a system that can read devanagari script. That will be helpful for Image to text translation. We can use it then for further applications eg. Text to Speech translation, Language translation. That will be helpful for those people who does not devanagari but still want to learn, for blind people, for those who does not know to pronounce devanagari words.

Script Reading

Here we will have a full script to read. First we will have to find the portions that includes a text. We can do this by a object detection algorithm. Then we have to segment the word into characters. We can use sliding window algorithm for this. Finally we will use CNN then output of CNN will go through RNN to recognize the word properly.

Text to Speech

Here first we have to recognize text then we have convert it to the audio equivalent. We can do this by a Encoder Decoder model. Encoder will recognize the text and Decoder will transform it to audio equivalent.

Language Translation

As before here also we can do this task by an Encoder Decoder model. Encoder will recognize the text and Decoder will translate it in another language.

References

[1] Breiman,L. (2001). Random Forest. *Machine Learning, 45, 5–32(2001)*

[2] J.S. Cramer (2002). The Origins of Logistic Regression. *TI 2002-119/4*

[3] Z. Boger and H. Guterman, "Knowledge extraction from artificial neural network models," *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, Orlando, FL, USA, 1997, pp. 3030-3035 vol.4

[4] O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks.

[5] A.Sherstinsky (2018). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network