Design and Development of a 2D-Convolution CNN model for Recognition of Handwritten Gurumukhi and Devanagari Characters

Indu Chhabra *chhabra_i@rediffmail.com*
Professor,
Department of Computer Science and Applications,
Panjab University, Chandigarh

Sushil Kumar Narang *susheelnarang@yahoo.com*
Research Scholar,
Department of Computer Science and Applications,
Panjab University, Chandigarh

Abstract

Owing to the innumerable appearances due to different writers, their writing styles, technical environment differences and noise, the handwritten character recognition has always been one of the most challenging task in pattern recognition. The emergence of deep learning has provided a new direction to break the limits of decades old traditional methods. There exist many scripts in the world which are being used by millions of people. Handwritten character recognition studies of several of these scripts are found in the literature. Different hand-crafted feature sets have been used in these recognition studies. Feature based approaches derive important properties from the test patterns and employ them in a more sophisticated classification model. Feature extraction using Zernike moment and Polar harmonic transformation techniques was also performed and a moderate classification accuracy was also achieved. The problems faced while using these techniques led us to use CNN based recognition approach which is capable of learning the feature vector from the training character image samples in an unsupervised manner in the sense that no hand-crafting is employed to determine the feature vector. This paper presents a deep learning paradigm using a Convolution Neural Network (CNN) which is implemented for handwritten Gurumukhi and devanagari character recognition (HGDCR). In the present experiment, the training of a 34-layer CNN for a 35 class self-generated handwritten Gurumukhi and 60 class (50 alphabet and 10 digits) handwritten Devanagari character dataset was performed on a GPU (Graphic Processing Unit) machine. The experiment resulted with an average recognition accuracy of more than 92% in case of Handwritten Gurumukhi Character dataset and 97.25% in case of Handwritten Devanagari Character dataset. It was also concluded that the training and classification through our network design performed about 10 times faster than on a moderately fast CPU. The advantage of this framework is proved by the experimental results.

Keywords: Handwritten Character Recognition, Neural Network, Deep Learning, Convolution Neural Network.

1. INTRODUCTION

Traditionally handwritten character recognition [1] techniques were based upon template and feature based approaches. Feature based approaches followed the principle of deriving the important features from the test sample character images and feeding them to a sophisticated classification model. Both spatial domain (e.g. Hu and Zernike moment [2, 3] based) and transform domain approaches (e.g. Wavelet [4] and Polar harmonic transform [5, 6, 7] based) were deployed. Features extracted using these approaches were used to create and train a classification model usually built using artificial neural networks [8]. But it is also established that

manually extracted features require prior knowledge of the language and these are not particularly robust to the diversity and complexity of handwriting.

Recently the advent of deep learning has led to the development of Convolutional neural network (CNN) [9-11] systems are the absolute most persuasive developments in the field of computer vision and pattern recognition. As a new feature extraction method, deep learning has made achievements in text mining. The major difference between deep learning [10, 11] and conventional methods is that deep learning automatically learns features from the supplied data, instead of adopting handcrafted features including millions of parameters. At present, deep learning feature representation includes autoencoder, restricted Boltzmann model, deep belief network [9, 10], convolutional neural network and recurrent neural network, etc.

The year 2012 was the principal year that neural nets developed to noticeable quality as Alex Krizhevsky utilized them to win ImageNet competition during that year. Ever from that point forward, a large group of organizations has been utilizing profound learning at the center of their administrations. Facebook utilizes neural nets for their programmed labeling calculations, Google for their photograph seeking and Amazon for their product suggestions.

Deep learning algorithms try to learn high-level features from data. This is a very distinctive part of Deep Learning and a major step ahead of traditional Machine Learning. Among all deep learning approaches, CNN is one of the most popular model and has been providing the state-of-the-art performance on object recognition. CNN approach has been designed to imitate human visual processing, and it has highly optimized structures to process 2D images. Further, CNN can effectively learn the extraction and abstraction of 2D features.

A Convolutional Neural Network (CNN) is made of at one or more convolutional layers and then is taken after by at least one or more completely associated layers as in a standard multilayer neural system. The engineering of a CNN is intended to exploit the 2D structure of an information picture. This is accomplished with nearby associations and tied weights which are carried further by some type of pooling which brings about interpretation invariant highlights. Another advantage of CNNs is that they are simpler to train and have numerous fewer parameters than completely associated systems with a similar number of shrouded units.

Convolutional Neural Networks (ConvNets or CNN's) [12, 13] are a class of Neural Networks that have demonstrated extremely compelling in application zones, for example, picture acknowledgment and characterization. ConvNets have been fruitful in distinguishing faces, objects and movement signs separated from controlling vision in robots and self-driving autos.

2. LITERATURE STUDY

In the field of pattern recognition, the curse of dimensionality [14] means that that the learning complexity grows exponentially with linear increase in the dimensionality of the data. The traditional approach of dealing with this curse has been through reducing the dimensionality, the process which is also called as feature extraction. This process of feature extraction has been quite challenging and highly application dependent. Besides, the incomplete or erroneous features may limit the performance of the classification process.

The early work on the automatic recognition of characters has been concentrated either upon well printed text or upon small set of well distinguished hand written text or symbols. Successful but, constrained algorithms had been implemented mostly for Latin characters and numerals. Besides some studies on Japanese, Chinese, Hebrew, Indian, Cyrillic, Greek and Arabic characters and numerals in both printed and handwritten cases are also done. The early work on the automatic recognition of characters has been concentrated either upon well printed text or upon small set of well distinguished hand written text or symbols. Successful but, constrained algorithms had been implemented mostly for Latin characters and numerals. Besides some studies on Japanese,

Chinese, Hebrew, Indian, Cyrillic, Greek and Arabic characters and numerals in both printed and handwritten cases are also done.

Initially G.S. Lehal and C. Singh [15] suggested “two sets of features”. Primary and Secondary feature sets were developed. The features used in Primary Feature Set were Number of junctions with the headline, Presence of sidebar, Presence of a loop and No Loop formed with headline. The Secondary feature set consisted of Number of endpoints and their location, Number of junctions and their location, Horizontal Projection Count, Right Profile depth, Left Profile Upper Depth, Left Profile Lower Depth, Left and Right Profile Direction Code and Aspect Ratio. A recognition rate of 91.6% was achieved and the average processing time was 4 millisecond for each character using these features. Later on in 2009, G.S. Lehal suggested OCR using Multiple Classifiers [15] and received an accuracy rate of 99.59%.

S. Arora, D. Bhattacharjee, M. Nasipuri, D.K. Basu and M.Kundu proposed a scheme for offline Handwritten Devnagari Character Recognition [16], which used different feature extraction methodologies and recognition algorithms. First the character was preprocessed and features namely: Chain code histogram and moment invariant features were extracted and fed to Multilayer Perceptrons as a preliminary recognition step. Finally the results of both MLP’s were combined using weighted majority scheme. It was observed that the proposed system achieved recognition rates of 98.03% for top 5 results and 89.46% for top 1 result.

P. Kasza presented a pseudo-Zernike feature descriptor [17] based recognition technique for accurate identification of printed and handwritten Chinese characters.

K.C. Leung and C.H. Leung also proposed a ‘‘critical region analysis’’ [18] technique which highlighted the critical regions that distinguish one character from another similar character. The critical regions were identified automatically based on the output of the Fisher’s discriminant. Additional features were extracted from these regions and contributed to the recognition process. By incorporating this technique into the character recognition system, a record high recognition rate of 99.53% on the ETL-9B database was obtained.

The emergence of deep learning which focusses on capturing both spatial and temporal dependencies in the patterns, has paved the way for the development of much more efficient pattern recognition models. Convolutional Neural Networks [19] which use deep learning approach, are a multi-layer neural networks predominantly designed for use on two-dimensional data, such as images.

A deep learning approach using CNN was purposed for recognizing the handwritten Farsi/Arabic digits [20] by fusing the recognition results of a number of Convolutional Neural Networks with gradient descent training algorithm. Their results revealed a very high accuracy classifier outperforming most of the previous systems showing 99.17% in recognition rate.

A model based on densely-connected belief nets [21] that have many hidden layers and which uses a fast, and greedy algorithm, was suggested and was proved that this model gives better digit classification than the best discriminative learning algorithms available at that time. And then performance evaluation for CNN and DBN [22] on the MNIST database was conducted and it was found that the classification accuracy rate of CNN and DBN on the MNIST database was 99.28% and 98.12% respectively.

The convolutional neural networks (CNN) offered contemporary end-to-end methodology for handwritten Chinese character recognition (HCCR) [23] with very promising results in recent years. A streamlined version of GoogleNet with deep architecture generated new state of the art recognition accuracy of 96.35%. For offline Arabic handwriting recognition [24], results proved that the new design based-SVM of the CNN classifier architecture with dropout performs significantly more efficiently than CNN based-SVM model without dropout and the standard CNN classifier.

3. CNN PARADIGM FOR HGDCR

The proposed framework has been exhibited in Fig. 1. The framework majorly consists of four stages: First, the sample dataset preparation (includes pre-processing and normalization), CNN model creation and compilation, training the model (including model accuracy/error report preparation) and finally testing.

- Dataset Formulation: Since the training of CNN requires a large number of training samples, the sample generation is important to provide enough number of samples to fully train the CNN model. Dataset is prepared systematically by labelling all the input images. The preprocessing and normalization of the dataset is performed using image resizing and histogram equalization techniques.
- Defining CNN Architecture: The network structure of the CNN models is designed according to the properties of handwritten characters and several training tricks are also employed for model compilation and further causing better training.
- CNN Model Training/Testing: Training of the model is prepared using various parameter settings and optimizers are changed for further fine tuning the model. See Table 1 for training details.
- CNN Model Prediction: Testing of the model is done by providing a character image and predicting the class of the supplied character.

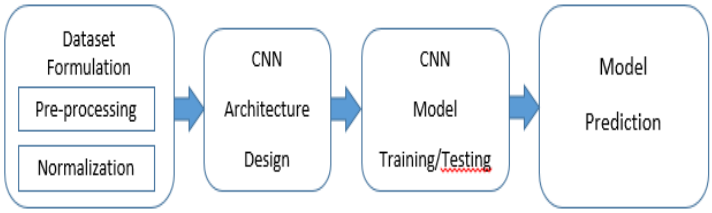

FIGURE 1: The Proposed Framework.

Layer(type)	Output Shape	Param #
Image_array (Conv2D)	(None,75,75,16)	800
Batch_Normalization_1 (Batch)	(None,75,75,16)	64
Conv2D_1(Conv2D)	(None,75,75,16)	12560
Batch_Normalization_2 (Batch)	(None,75,75,16)	64
Activation_1(Activation)	(None,75,75,16)	0
Average_pooling2D_1(Average	(None,38,38,16)	0
Dropout_1(Dropout)	(None,38,38,16)	0
Conv2D_2(Conv2D)	(None,38,38,32)	12832
Batch_Normalization_3 (Batch)	(None,38,38,32)	128
Conv2D_3(Conv2D)	(None,38,38,32)	25632
Batch_Normalization_4 (Batch)	(None,38,38,32)	128
Activation_2(Activation)	(None,38,38,32)	0
Average_pooling2D_2(Average	(None,19,19,32)	0
Dropout_2(Dropout)	(None,19,19,32)	0
Conv2D_4(Conv2D)	(None,19,19,64)	18496
Batch_Normalization_5 (Batch)	(None,19,19,64)	256
Conv2D_5(Conv2D)	(None,19,19,64)	36928
Batch_Normalization_6 (Batch)	(None,19,19,64)	256
Activation_3(Activation)	(None,19,19,64)	0
Average_pooling2D_3(Average	(None,10,10,64)	0
Dropout_3 (Dropout)	(None,10,10,64)	0
Conv2D_6(Conv2D)	(None,10,10,128)	73856
Batch_Normalization_7 (Batch)	(None,10,10,128)	512
Conv2D_7(Conv2D)	(None,10,10,128)	147584
Batch_Normalization_8 (Batch)	(None,10,10,128)	512
Activation_4(Activation)	(None,10,10,128)	0

Average_pooling2D_4(Average	(None,5,5,128)	0
Dropout_4 (Dropout)	(None,5,5,128)	0
Conv2D_8(Conv2D)	(None,5,5,256)	295168
Batch_Normalization_9 (Batch)	(None,5,5,256)	1024
Conv2D_9(Conv2D)	(None,5,5,35)	80675
global_average_pooling2D_1	(None,35)	0
Predictions (Activation)	(None,35)	0

TABLE 1: Training Model Details for Gurmukhi & Devanagari character recognition.

The following part contains details about the proposed framework.

3.1 Dataset Formulation

Due to the paucity of the standard Gurumukhi [25] dataset, we have collected the sample handwritten character images from the writings of persons of different age groups. Then isolated character images were extracted from the scanned document image and stored in their respective classes. And for devnagari character dataset, we have used a generic comprehensive devnagari numeral and character database developed by V. J. Dongre and V. H. Mankar [26]. The images from both the datasets have been shown in Figure 3 and Figure 4. Further the individual images were labelled according to their class names. Since all the character images present in the dataset were of different sizes, the same were preprocessed by introducing image resizing i.e. resized all the images to the size [75,75]. Following that, all the images were converted from rgb to grayscale before being fed to input layers. Also as found in traditional methods, the normalization is important for reducing the variance of the characters, resulting in reducing the classification difficulty. Therefore normalization was performed using histogram equalization which enhances the image by changing the range of pixel intensity values. The resulting equalized image possesses approximately linear cumulative distribution function for every pixel proximity. Since in the future, the training process might be rerun more than one time before it is actually evaluated, it is useful that we don't have to go through all the images on every run and read every image, preprocess and normalize it, make labels for every image, split it into train/test for training and evaluating the model, encode the labels from text to numeric. That is why we store the results of each process in the form of NumPy binary files. NumPy files contain all images data converted in the form of multi-dimensional arrays.

FIGURE 3: Gurmukhi Handwritten Dataset.

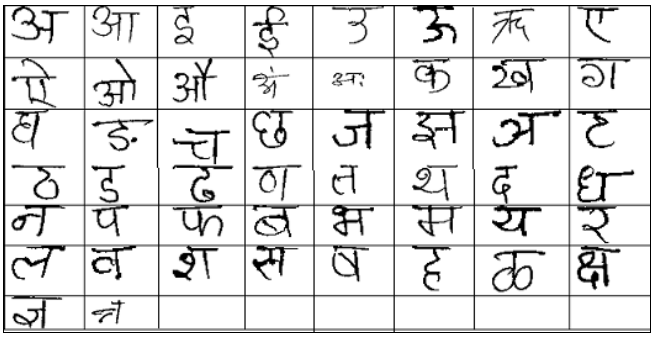

FIGURE 4: Devanagari Handwritten Dataset.

As deep learning models take images separately, and labels separately that is why we have to separate them out. The whole dataset in the form of the files was loaded and split into corresponding train and test data (80% for train and 20% for test in our case). Resulting files were stored in numpy(npy) format as described in Table 2.

Data File	Meaning
X_train	Training Data
X_test	Test Data
Y_train	Corresponding labels for train data
Y_test	Corresponding labels for test data
Y_train_encoded	Train data labels converted into numeric
Y_test_encoded	Test data labels converted into numeric

TABLE 2: Data File Details.

3.2 HGDCR CNN Model

The architecture forms a 2D sequential convolutional Neural Network model which is linear stack of 33 layers out of which ten convolution layers were deployed. The size of first two convolution filters (Total: 16) is 7 x 7 in first two layers, 5 x 5 in next subsequent two layers (Total: 32) and 3 x 3 in the rest of six consecutive following layers (64 filters in the layer no 5 and 6, 128 filters in the layer 7 & 8 and 256 filters in the last two convolution layers). As the image reduces in size after every pooling layer, our filters are increasing to learn as much information as possible.

3.2.1 Each time after convolution layer, the output is fed to batch normalization. A major glitch in sequence model is that as output of each layer is dependent on each layer, this steers to correlation between the layers. Therefore the changes made in any one layer will definitely affect all the following layers. This is seen as Internal Covariate Shift [27]. Batch normalization [27] addresses this problem by normalizing the output of each layer in mini-batches, which decreases the effect of changes in one layer in by both mean and variance reference. This also leads to allowing us to use higher learning rate to train our model. The normalized output is then fed to activation layer.

3.2.2 In order to get better rate of approximation, the choice of an activation function [28] plays an important role. As the study shows that Sigmoids and tanh functions should be sometimes avoided because of vanishing gradient problem, 'relu' – rectified linear unit function was chosen. It's just $R(x) = \max(0,x)$ i.e if $x < 0$, $R(x) = 0$ and if $x >= 0$, $R(x) = x$. Hence it is apparent from the mathematical form of this function that it is very simple and efficient. ReLu is nonlinear in nature. Using a sigmoid or tanh set off almost all neurons to fire in an analog way, which indicates that almost all activations are processed to describe the output of a network. In other words the activation becomes dense which is a costly affair. Ideally a few neurons in the network should not activate so that activations could be sparse and efficient.

ReLU function provides this benefit. If we construct a CNN and randomly initialize the weights, almost 50% of the network then will yield 0 activation because of the characteristic of ReLU which gives output 0 for negative input values. This means a fewer neurons are firing and the network is lighter. It is also evident from the mathematical forms of all activation functions that ReLU is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations compelling it a better choice while designing CNNs.

3.2.3 After ReLU layers, we applied pooling layers [29] which is also referred to as down sampling layer. When we get a high activation value and we establish a particular feature in the original input volume, its exact location is not as significant as its relative location with respect to the other features. This becomes the prime reason behind using this layer. This layer radically reduces the spatial dimension of the input volume. This performs two main functions. The first is that the amount of weights is reduced by almost 75% thereby reducing the computation cost. The second is that it helps in controlling over fitting. Over fitting might occur when the model is so fine-tuned to the training examples that it is not able to generalize well for the validation and test sets. The available options to pooling layer were max pooling or average pooling. As the choice of the pooling method depends on the structure of the image dataset therefore, we experimented with both pooling methods on our dataset. Comparative classification performances with both pooling methods are depicted in the Table 3.

Method	Training Error %	Accuracy %
Average Pooling	32.0684%	92.2857%
Max Pooling	33.1124%	91.7568%

TABLE 3: Comparative Classification Performances using different Pooling techniques.

3.2.4 We did make use of drop out layer [30] after the average pooling layer whose output was fed to drop out layer. The function of this layer is to drop out a random set of activations by setting them to zero. By doing so, it forces the network to be redundant and establishes that the network is able to provide the right classification even if some of the activations are dropped out. It also makes sure that the network doesn't get over fitted to the training data and thus assists alleviate the over fitting problem. We have set 50% of randomly chosen nodes to zero to prevent overfitting for each iteration so that network may not start using only a few filters. It enforces network to utilize all network paths as if one is set to zero, network tries to learn non-zero path.

3.2.5 Global average pooling layer [19] has been introduced before the final activation layer instead of fully connected layer. GAP layer takes the average of each feature vector for each corresponding category of the classification process. The subsequent feature vector is supplied into softmax layer. Also the GAP layer does not require any parameter optimization therefore there is less probability of over fitting. This structure bridges the convolutional structure with traditional neural network classifiers. Nevertheless, the fully connected layers being prone to over fitting, hinder the generalization ability of the whole network.

In our Gurumukhi handwritten character recognition, we had 35 different mutually exclusive classes, hence, the dimension of the output class is 35. For Devanagari character recognition we had 60 different mutually exclusive classes so the dimensionality of the output class has been set to 60. Ideally, the best prediction is if the probability comes out to be 1.0 for a single output node, and probability of rest of the output nodes comes out zero. We have used Softmax function as output layer which roughly work like Max layer as well as it is differentiable to train by gradient descent. Finally, a softmax layer trained on our Gurumukhi dataset will output a separate probability for each of the thirty five characters, and the probabilities will all add up to 1. The output of this softmax classifier is in the form of an array of probabilities for each class. The highest probability in this array is the class that is predicted.

3.3 Experiment

We performed this experiment on Intel Core I-7 7th Generation, 16GB RAM, nVidia GTX 1060 6GB Graphics Card, machine. The code was run on linux which was having Anaconda for python, Keras 2.1.5 as frontend model and Tensorflow r1.7 as backend. Keras is a high-level API for neural networks. It is written in Python and capable of running on top of TensorFlow. Our gurmukhi dataset contained hundred handwritten character images, out of which 80 images were used for testing and 20 were used for testing purposes.

3.4 Results

In order to compile and evaluate the quality of our neural network model, we preferred to use cross-entropy loss function which is also known as log loss. The principle is that as the predicted probability approaches 1, log loss slowly decreases whereas when the predicted probability decreases, the log loss increases rapidly. Once the computed output gets closer to 0.0 or 1.0, the chances of gradient getting smaller and smaller are diminished in case of cross entropy and the training does not stall out. Finally, compilation process was tested using different optimizers (Adam (Adaptive Moment estimation), RMSprop (root mean squared propagation) and SGD (Stochastic gradient descent) and nadam (Nesterov Adam) [31] out of which 'nadam' was found more efficient for our classification process as it provided better convergence rate guarantee than all other. As this deep learning model was expected to take hours, there were chances of losing a lot of work if the run was stopped unexpectedly or due to any other fault, check points have been introduced in the CNN model. A snapshot of the state of the system was fixed to be taken in case of system failure. The training data consisted of 707,475 total params (weights) out of which 706,003 were found to be trainable and rest 1472 were non-trainable. Non-trainable parameters means the number of weights which were chosen to keep constant when training i.e. the algorithm didn't update these weights during training. The confusion matrix after as well as without normalization was displayed. The classification summary was displayed showing precision, F1-score, recall and support values for each individual character. The f1-score represents the harmonic mean of precision and recall. The scores corresponding to every class tells the accuracy of the classifier in classifying the data points in that particular class compared to all other classes. The support is derived from the number of the samples with the true response that lie in that class. A glimpse of classification summary report is shown in Fig using pre-labelled input images from the dataset.

Gurumukhi Character Classification Summary				**Devanagari Character Classification Summary**			
Alphabet Labels (as per phonetic sound)	**Precision**	**Recall**	**F1-Score**	**Alphabet Labels (as per phonetic sound)**	**Precision**	**Recall**	**F1-Score**
Bhubba	0.90	0.95	0.93	A	0.95	1.00	0.98
Bubba	0.83	0.95	0.88	Aa	0.95	1.00	0.98
Chhuc-hha	0.86	0.95	0.90	Aae	0.90	0.95	0.93
Chucha	0.90	0.90	0.90	Ae	1.00	1.00	1.00
Dhuda	1.00	1.00	1.00	Ang	1.00	0.85	0.92
Dhudda	1.00	1.00	1.00	Bada-Ae	1.00	1.00	1.00
Duda	0.87	1.00	0.93	Bada-O	0.95	1.00	0.98
Dudda	1.00	0.75	0.86	Bada-oo	1.00	1.00	1.00
Ghugga	0.91	1.00	0.95	Ch	1.00	1.00	1.00
Jhujja	0.95	0.95	0.95	Chh	1.00	1.00	1.00
Jujja	1.00	0.95	0.97	Chota-O	1.00	0.95	0.97
Lulla	1.00	0.85	0.92	Chota-oo	0.95	1.00	0.98
Mumma	1.00	0.95	0.92	D	1.00	1.00	1.00
Nannhha	1.00	0.75	0.97	D-dawaat	0.95	0.95	0.95
Nunna	0.88	0.90	0.81	Dh	0.95	0.95	0.95
Pupha	0.82	0.85	0.86	Dd	0.95	0.90	0.92
Puppa	0.89	0.95	0.87	Dn	0.95	0.90	0.92
Rahrha	1.00	1.00	1.00	E	0.86	0.95	0.90

Rara	1.00	0.85	0.92	Ee	1.00	1.00	1.00
Tainka	0.90	0.95	0.93	G	1.00	1.00	1.00
Thhuthha	0.73	0.95	0.83	Gh	1.00	0.95	0.97
Thutha	0.94	0.85	0.89	Gy	1.00	0.95	0.97
Tutta	0.89	0.85	0.87	H	1.00	1.00	1.00
Ungga	1.00	0.90	0.95	J	1.00	1.00	1.00
Vava	1.00	0.95	0.97	Jh	0.95	0.90	0.92
Yaiyya	0.94	0.85	0.89	K	1.00	1.00	1.00
Yanza	0.95	1.00	0.98	Kh	1.00	0.90	0.95
Eerhi	0.87	0.90	0.95	Ksh	0.90	0.95	0.93
Erha	0.95	1.00	0.98	L	1.00	0.85	0.92
Gugga	0.87	1.00	0.93	Nn	0.91	1.00	0.95
Haha	0.95	1.00	0.98	Nh-veena	0.91	1.00	0.95
Khukha	0.90	0.95	0.93	P	1.00	1.00	1.00
Kukka	0.90	0.95	0.93	Ph	1.00	1.00	1.00
Sussa	0.80	0.80	0.80	RRishi	0.91	1.00	0.95
Urha	0.91	1.00	0.95	S	1.00	1.00	1.00
Avg/Total	**0.93**	**0.92**	**0.92**	Shh	0.95	1.00	0.98
				T-Tamatar	0.91	1.00	0.95
				Thh	1.00	0.95	0.97
				V	0.95	0.95	0.95
				Y	0.95	1.00	0.98
				b	1.00	0.95	0.97
				Bh	1.00	0.95	0.97
				La	0.95	1.00	0.98
				m	1.00	1.00	1.00
				n	0.91	1.00	0.95
				r	1.00	0.95	0.97
				sh	1.00	1.00	1.00
				t	0.95	1.00	0.98
				th	1.00	0.85	0.92
				trra	1.00	0.90	0.95
				Avg/Total	**0.95**	**1.00**	**0.98**

FIGURE 5: Classification summary of All Gurmukhi and Devnagari Characters.

The model resulted in an average accuracy of more than 92% and 97% in case Gurumukhi and Devanagari Characters respectively as shown in Figure 5. In order to improve the recognition accuracy, Confusion matrix is generated from large amount of data of both scripts. The confusion matrix showed how many times one character is confused with the other character in output with given Input. It is reasonable that the confusion between characters occurs basically due to similar shape characters. A lot of algorithmic changes were done while fine tuning the model. The resultant model produced the highest accuracy rate while minimizing the error. As a result this was chosen to be the base model for this recognition process.

4. CONCLUSION

This research work deals with the recognition of Handwritten Gurmukhi and Devanagari Characters using deep learning algorithms by implementing convolution neural networks. Proposed research work delivers more efficient and accurate results than any other existing machine learning systems. As a part of future work, recognition accuracy needs to be tested by augmenting our datasets by auto generating character images by performing different alterations on existing datasets. Our experimental result shows that the recognition rate is reduced a bit due to false detection of few characters during testing. In spite of that the overall recognition accuracy is acceptable and conforms to somewhat human like recognition. The recognition accuracy also needs to be tested by combining CNN with recurrent neural networks using LSTM or GRU like deep learning algorithms also and deriving the probability of a single deep learning model for

recognizing characters from multiple scripts instead of creating different models for different scripts.

5. REFERENCES

[1] R. Plamondon and S. N. Srihari, “On-Line and off-line handwritten recognition: A comprehensive survey”, IEEE Trans on PAMI, Vol.22, pp.62-84, 2000.

[2] C. Kan and M. D. Srinath, “Invariant character recognition with Zernike and orthogonal Fourier–Mellin moments,” Pattern recognition, 35(1), pp.143-154, 2002.

[3] S.O. Belkasim, M. Shridhar and M. Ahmadi, "Pattern recognition with moment invariants: A comparative study and new results,” Pattern recognition, 24(12), pp.1117-1138, 1991.

[4] A. Majumdar and A. Bhattacharya, “A Comparative Study in Wavelets, Curvelets and Contourlets as Feature Sets for Pattern Recognition,” International Arab Journal of Information Technology (IAJIT), 6(1), 2009.

[5] P. T. Yap, X. Jiang and A. C. Kot, “Two-dimensional polar harmonic transforms for invariant image representation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(7), pp.1259-1270, 2010.

[6] T. V. Hoang and S. Tabbone, “Generic polar harmonic transforms for invariant image representation,” Image and Vision Computing, 32(8), pp.497-509, 2014.

[7] I. Chhabra and S. Narang, “Applying Polar Harmonic Transform for feature extraction from a two-dimensional Handwritten Character Image”, in Proc. of Twenty First International Conference of FIM on interdisciplinary Mathematics, Statistics and Computational Techniques IMSCT-2012-FIM XXI, Panjab University, Chandigarh, Dec. 2012.

[8] C. Singh and I. Chhabra, “Recognition System: A Knowledge Based Neural Approach”, in Proc. of International Conference on Speech and Language Technology, New Delhi, Nov. 2004.

[9] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition,” in Proc. of the IEEE, 86(11), 1998, pp.2278-2324.

[10] I. Arel, D. C. Rose and T. P. Karnowski, “Deep machine learning-a new frontier in artificial intelligence research,” IEEE computational intelligence magazine, 5(4), pp.13-18, 2010.

[11] J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural networks, 61, pp.85-117, 2015.

[12] L. Chen, S. Wang, W. Fan, J. Sun and S. Naoi, “Beyond human recognition: A CNN-based framework for handwritten character recognition,” In Proc. IEEE 3rd Asian Conference on Pattern Recognition (ACPR), 2015, pp. 695-699.

[13] D. S. Maitra, U. Bhattacharya and S. K. Parui, “CNN based common approach to handwritten character recognition of multiple scripts,” In Proc. IEEE 13th International Conference on Document Analysis and Recognition (ICDAR), 2015, pp.1021-1025.

[14] R. B. Marimont and M. B. Shapiro, “Nearest neighbour searches and the curse of dimensionality,” IMA Journal of Applied Mathematics, 24(1), pp.59-70, 1979.

[15] G. S. Lehal and C. Singh, “A Gurmukhi script recognition system,” In Proc. IEEE 15th International Conference on Pattern Recognition, 2000, Vol. 2, pp.557-560.

[16] S. Arora, D. Bhattacharjee, M. Nasipuri, D. K. Basu, M. Kundu and L. Malik, “Study of different features on handwritten Devnagari character,” in Proc. IEEE 2nd International

Conference on Emerging Trends in Engineering and Technology (ICETET), 2009, pp.929-933.

[17] P. Kasza, "Pseudo-Zernike Moments for Feature Extraction and Chinese Character Recognition," In Proc. IEEE second International Conference on Computer Automation and Engineering, 2009.

[18] K. C. Leung and C. H. Leung, "Recognition of handwritten Chinese characters by critical region analysis," Pattern Recognition, 43(3), pp.949-961, 2010.

[19] M. Lin, Q. Chen and S. Yan, "Network in network," arXiv preprint arXiv:1312.4400, 2013.

[20] S. S. Ahranjany, F. Razzazi and M. H. Ghassemian, "A very high accuracy handwritten character recognition system for Farsi/Arabic digits using Convolutional Neural Networks," In Proc. IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010, pp. 1585-1592.

[21] M. Wu and L. Chen, "Image recognition based on deep learning," In Proc. IEEE Chinese Automation Congress (CAC), 2015, pp. 542-546.

[22] G. E. Hinton, S. Osindero and Y. W. Teh, "A fast learning algorithm for deep belief nets," Neural computation, 18(7), pp.1527-1554, 2006.

[23] Z. Zhong, L. Jin and Z. Xie, "High performance offline handwritten chinese character recognition using googlenet and directional feature maps," In Proc. IEEE 13th International Conference on Document Analysis and Recognition (ICDAR), 2015, pp. 846-850.

[24] M. Elleuch, R. Maalej and M. Kherallah, "A new design based-SVM of the CNN classifier architecture with dropout for offline Arabic handwritten recognition," Procedia Computer Science, 80, pp.1712-1723, 2016.

[25] I. Chhabra and C. Singh, "Integration of Structural and Statistical Information for the Recognition of Gurmukhi Script," in Proc. International Conference on Applied Computing, Algarve, Portugal, Feb 2005.

[26] V. J. Dongre and V. H. Mankar, "Development of comprehensive devnagari numeral and character database for offline handwritten character recognition," Applied Computational Intelligence and Soft Computing, pp.29, 2012.

[27] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in Proc. 32nd International Conference on Machine Learning (ICML'15), pp. 448-456, 2015.

[28] H. N. Mhaskar and C. A. Micchelli, "How to choose an activation function," In Advances in Neural Information Processing Systems, pp. 319-326, 1994.

[29] D. Scherer, A. Müller and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," In Proc. Artificial Neural Networks–ICANN, 2010, pp. 92-101.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," The Journal of Machine Learning Research, 15(1), pp.1929-1958, 2014.

[31] N. Ketkar, "Introduction to keras," In Deep Learning with Python, Apress, Berkeley, CA, 2017, pp. 97-111.