Journal of Information Systems Engineering and Management
2025, 10(10s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Modi Script Handwritten Character Classification using Drop-out Induced Incremental Learning Approach

[1]Chaitali Chandankhede, [2]Rajneeshkaur Sachdeo, [3]Urmila Shrawankar, [4]Nisha Wankhade, [5]Girish Talmale
[1]*Dr. Vishwanath Karad MIT World Peace University, Pune, India*
[2]*MIT-ADT University, Pune, India*
[3]*Ramdeobaba University (RBU), Nagpur, India*
[4]*Yeshwantrao Chavan College of Engineering, Nagpur*
[5]*G H Raisoni College of Engineering Nagpur, India*
[1]*https://orcid.org/0000-0002-1291-6636*, [2]*https://orcid.org/0009-0002-5387-5671*,
[3]*https://orcid.org/0000-0003-4523-9501*, [4]*https://orcid.org/0000-0002-8964-5459*,
[5]*https://orcid.org/0000-0002-0180-5228*

ARTICLE INFO

Received: 08 Nov 2024

Revised: 26 Dec 2024

Accepted: 10 Jan 2025

ABSTRACT

Most of genuine information based on healthcare, food habits and Ayurveda is imbibed in Modi script and very few people are remaining who understand Modi script reading and writing. This motivates us to work with Modi script. After going through the deep literature review, we found InceptionV3 and Residual framework (ResNet152) need to experiment on Modi script dataset. The main objective of the system is to develop an incremental learning model which starts identifying from individual characters with acceptable accuracy, which further trained and tested for words followed by sentences, group of sentences. We are the first one to experiment with whole Modi script character set which covers 360 class labels comprising 35 consonants and 10 vowels which was collected using different people. The individual characters are cut, labelled manually and pre-processed using Otsu, Savaula and To-zero binarization techniques. The model is further explored by adding InceptionResNetV2 framework for classification. Initially the model shows overfitting behaviour as discussed in section 5.2. To regularize the model, first experiment is done using augmented dataset which also not able to show satisfactory results and took un-conventional time for training. Further, drop-out approach is induced which shows good hyper-parameter tuning as shown in table 6. Due to drop-out layers, training is extended till 300 epochs to get better results. After training using drop-out technique, the model is showing proportionate increase in training and validation accuracy. At the same time, it is showing considerable consistent decrease in training and validation loss. This gives us intuition that the newly developed hybrid model helps to reduce overfitting and learns appropriately. The developed incremental model is tested on Modi words where all words are mostly classified correctly except the character “sa” which is in-correctly classified as banacha “na” sinch both the characters look similar.

Keywords: Deep learning, InceptionV3, Residual Network, Modi Script, To-zero, Sauvola, Otsu, Binarization, segmentation, classification

1. Introduction

Handwritten text recognition has a prominent role, among others, in business, healthcare or cultural heritage preservation. Slowly, while computer systems replace handwritten documents in day-to-day practices of today, still there exists very large body of documents that have been handwritten and should be digitized. This must be done, among others, to assure preservation of content as pen-and-paper documents have limited lifespan (due to normal ageing and degradation). Here, digitization must involve not only creation of images (which is required primarily for historical manuscripts, which are work of art in their own right, and preserving them as images is required), but also extraction of their actual content/text (needed for further processing/searching/storage). For the latter, text images are usually converted into a machine-encoded form, using Optical Character Recognition (OCR). Recently, a lot of work has been devoted to recognition of printed, as well as handwritten characters and still there is need to

focus due to rapid development of industries and academic fields, higher demand for recognition accuracy and recognition efficiency of HCR. The current deep learning methods still have room for further development.

2. III. State of the Art

2.1 After going through the rigorous literature following selected papers help to give direction to our research as discussed in table 1.

Table 1: Literature review highlights

Sr. No.	Paper Details	Techniques used	Major findings
1	Dave, N. (2015). Segmentation methods for handwritten character recognition. International journal of signal processing, image processing and pattern recognition, 8(4), 155-164.[25]	Pixel Counting and Histogram approach are discussed	Histogram approach more suitable for handwritten text
2	Husham, S., Mustapha, A., Mostafa, S. A., Al-Obaidi, M. K., Mohammed, M. A., Abdulmaged, A. I., & George, S. T. (2020). Comparative analysis between active contour and Otsu thresholding segmentation algorithms in segmenting brain tumor magnetic resonance imaging. Journal of Information Technology Management. [23]	Active Contour and Otsu thresholding methods are discussed on MRI images	Active Contour performs well over Otsu thresholding
3	Raynaud, G., Chane, C. S., Jacob, P., & Histace, A. (2019, February). Active Contour Segmentation based on Histograms and Dictionary Learning for Video Capsule Image Analysis. In VISIGRAPP.[24]	Active Contour based on Histogram and Dictionary Learning	Active Contour based on Histogram performs well on experime nted images
4	Jindal, U., Gupta, S., Jain, V., & Paprzycki, M. (2020). Offline Handwritten Gurumukhi Character Recognition System Using Deep Learning. In Advances in Bioinformatics, Multimedia, and Electronics Circuits and Signals (pp. 121-133). Springer, Singapore.[17]	CNN (Gurumukhi characters)	Training data accuracy is 98.32%, the testing data accuracy is 74.66%. on 35 characters
5	Mustafa, W. A., & Kader, M. M. M. A. (2018, June). Binarization of document images: A comprehensive review. In Journal of Physics: Conference Series (Vol. 1019, No. 1, p. 012023). IOP Publishing. [19]	7 binarization techniques used	Otsu and Gradient based threshold perform well
6	Guha, R., Das, N., Kundu, M., Nasipuri, M., & Santosh, K. C. (2020). Devnet: an efficient CNN architecture for handwritten devanagari character recognition. International Journal of Pattern Recognition and Artificial Intelligence, 34(12), 2052009. [20]	Different variants of CNN discussed	DevNet shows efficient though InceptionV3 and RestNet50 also gives effective results

7	Wang, Y., Xiao, W., & Li, S. (2021, April). Offline Handwritten Text Recognition Using Deep Learning: A Review. In Journal of Physics: Conference Series (Vol. 1848, No. 1, p. 012015). IOP Publishing.[21]	Deep learning concept for OHHR discussed	Deep learning techniques seems effective in the last decades
8	Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." Thirty-first AAAI conference on artificial intelligence. 2017.[22]	Inception-ResNet hybrid concept introduced	Inception-ResNet V2 seems effective
9	Li, Z., Wu, Q., Xiao, Y., Jin, M., & Lu, H. (2020). Deep Matching Network for Handwritten Chinese Character Recognition. Pattern Recognition, 107471. [18]	CNN with local SoftMax regression algorithm to reduce the computation and memory usage in training iteration. (Chinese characters)	Matching network has a very promising generalization ability.
10	Ram, S., Gupta, S., & Agarwal, B. (2018). Devanagri character recognition model using deep convolution neural network. Journal of Statistics and Management Systems, 21(4), 593-599.[1]	DL approach on devnagri script	Optimization of network by using best hyperparameters for the network
11	C. Chandankhede, “Offline MODI script character recognition using deep learning techniques,” 2023. Chandankhede, C., & Sachdeo, R. (2023). Offline MODI script character recognition using deep learning techniques. Multimedia Tools and Applications, 1-12 [2]	Deep learning model using resnet152 and IncetionV3 is developed on Modi dataset	Both algorithms perform good while training but shows overfitted behavior
12	C. Chandankhede and R. Sachdeo, “Character Recognition using MODI script : Facts , Challenges and its future,” no. 25389, pp. 25389–25395, 2020. [3]	Review paper on Modi script	Challenges, facts and future aspects of Modi lipi are discussed
13	Alom, M. Z., Sidike, P., Hasan, M., Taha, T. M., & Asari, V. K. (2018). Handwritten bangla character recognition using the state-of-the-art deep convolutional neural networks. Computational intelligence and neuroscience, 2018. (Hindawi journal)[4]	Use of dciscriminitive features with DCNN	DenseNet shows best performer in classifying Bangla digits, alphabets, and special characters.
14	Vaidya, R., Trivedi, D., Satra, S., & Pimpale, M. (2018, April). Handwritten character recognition using deep-learning. In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT) (pp. 772-775). IEEE.[5]	Developed image segmentation based HCR system	Simple HCR system is developed using English characters with use of python libraries
15	Jangid, M., & Srivastava, S. (2018). Handwritten devanagari character recognition using layer-wise training of deep convolutional neural networks and adaptive gradient methods. journal of imaging, 4(2), 41.[6]	layer-wise DCNN approach.	The best recognition accuracy and a quicker convergence rate were obtained by using a layer-wise DCNN approach.

3. Proposed Approach

As per [2], a deep learning approach is implemented using Resnet152 and InceptionV3. The data set collection, cutting, labelling and pre-processing of data is described in the paper [2]. Initially 10 modi characters are experimented using ResNet50 and InceptionV3 model. Later dataset is increased and experimented with ResNet152, InceptionV3. Again in further step the experimentation is done with combined features of Residual and Inception called InceptionResNetV2. Key functionality of Inception-ResNet is output of inception module is added to input from previous layer.

As per algorithms discussed in chapter 3, Inception V2 and V3 reduce computational cost by factorizing the filters. Though Inception V2 and V3 have same work theory, Inception V3 has improvement like it optimizes 7x7 filters and label smoothing. InceptionResNet is a hybrid framework inspired by both Inception and ResNet. InceptionResNetV1 has a computational cost that is like that of InceptionV3 and InceptionresNetV2 has computational cost similar to InceptionV4[14]. So we choose InceptionV3 and InceptionResnetV2 for further experiment.

After doing classification using these 3 algorithms, the model seems overfitted which was regularized using drop-out module. As shown in graphs after training phase without drop-out, Finally, we added the module for testing Modi handwritten words. As per figure 1 shows training behaviour at 20 epochs using ResNet152+Otsu, ResNet+Savaula and ResNet_Tozero. In figure 1, with increase in trailing accuracy, training loss is increasing gradually and though training loss is decreasing acceptably but validation loss is increasing exponentially which shows the overfitted behaviour. The similar behaviour of the model was observed for InceptionV3 and InceptionResNet algorithms as well. The quantitive analysis is shown in table 2. It results in poor detection of unknown characters while testing the model. So initially we experimented with augmented dataset but as dataset size was increased almost 4 times than existing data images, so training time was increased un-conventionally. Then the drop-out induced framework is designed and experimented which had shown the satisfactory behaviour as shown in graphs of figure 2. The final modified version of proposed framework is represented as in figure 3. The implementation is divided into two major modules classification model word detection module.

We have discussed the working model of our framework using algorithm discussed in section 4.

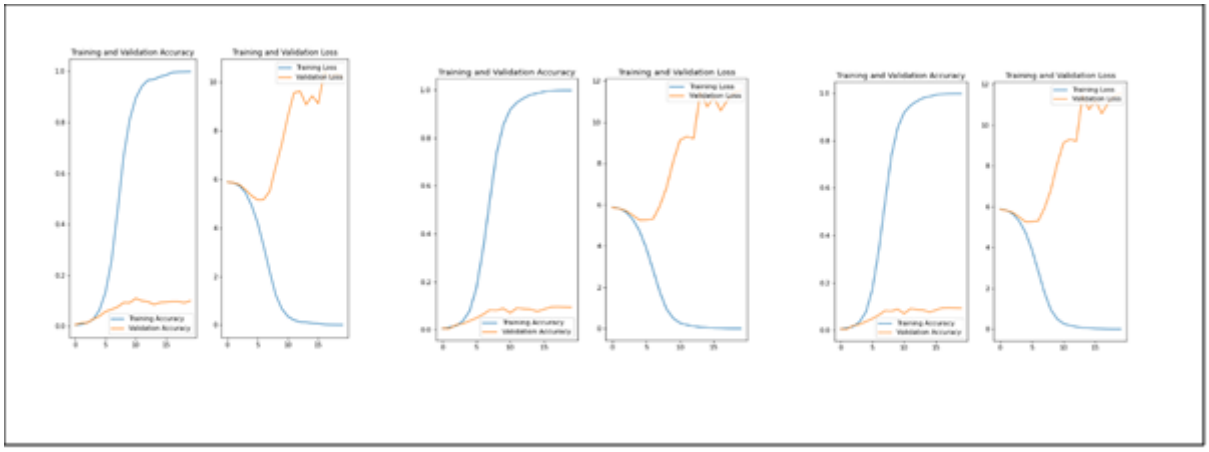

Figure 1: Training vs validation accuracy and training vs validation loss before drop-out for ResNet152 at 20 epochs

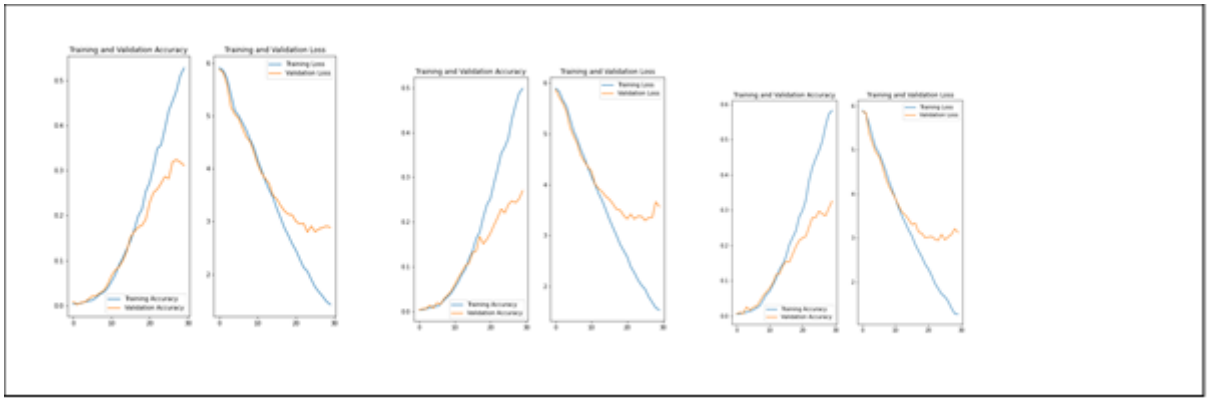

Figure 2: Training vs validation accuracy and training vs validation loss after drop-out for ResNet152 at 20 epochs

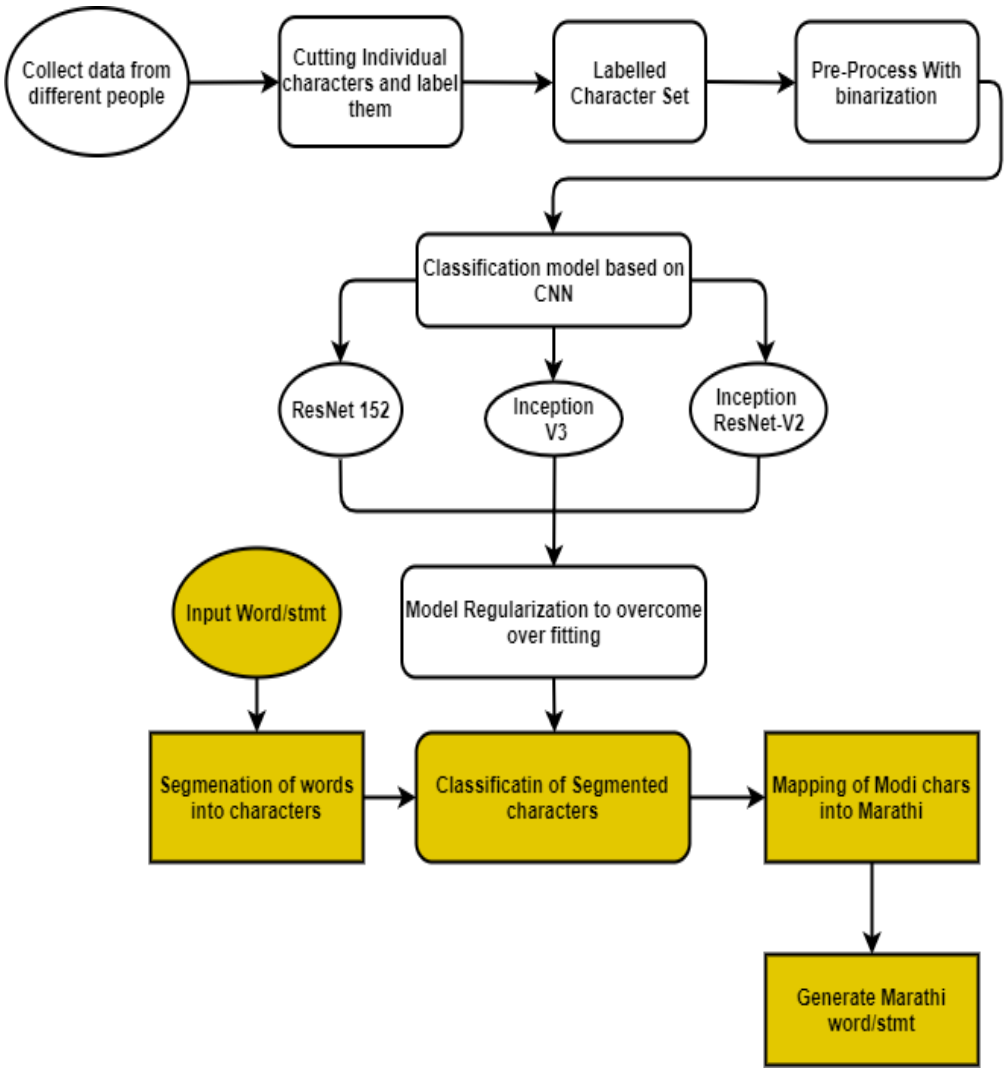

Figure 3: Modi barakhadi characters recognition framework

4. Implementation details

The whole framework is implemented as per following algorithm on python environment. Initially Input folder with 360 labelled subfolders is loaded for training. For training we have used total 7221 images and for testing we used total 1786 images. Labelling is di=one as per marathi character set. Then pre-processing of all images in training folder is done using 3 binarization techniques namely Otsu, Sauvola and Tozero. Next step is training the model using train set as per created mode discussed in step iv. The model is trained for regular interval of epochs 10 20, 30, 50, 100, 200, and 300 epochs. The trained model is tested using pre-processed

test set. We have trained the model till 300 epochs as due to drop-out techniques training is done gradually.

1. **Algorithm:** Image Classification using InceptionV3 / ResNet152 / InceptionResNetV2 algorithms
2. **Input:** Test images from test folder
3. **Output:** Prediction result in percentage of prediction accuracy
4. **Steps:**
    a. Arrange character images in folders. System will identify every folder name as classification label.
    b. Pre-process
        i. Pre-process images using 3 different thresholds: Otsu, Sauvola, Tozero
    c. Training
        i. Use pre-processed images from Pre-Process folder for every threshold.
        ii. Load every image from pre-process folder.
        iii. Create train and validation dataset.
        iv. Create Model as per algorithm:
            1. Image Size: 256 X 256
            2. Dense using Activation: Relu
            3. Add Extra Convolution 2D layers (Filters: 64 & 128).
            4. Add 3 dense layers (Units: 1024, 512 & 256).
            5. Add Dropout 0.3 and then 0.2.
        v. Compile model
            1. Optimizer: adam
            2. loss: SparseCategoricalCrossentropy
            3. metrics: accuracy
        vi. Fit Model
            1. Epoch Count: 10, 20, 30, 50, 100, 200, 300
            2. Validation split: 0.33.
        vii. Save Model
    d. Prediction
        i. Load test images
        ii. Pre-process images using 3 different thresholds: Otsu, Sauvola, Tozero
        iii. Load model
        iv. Predict images.
        v. Show result.

5. Experimental details and discussion

First experiment is done using un-processed images. Since the results of un-processed images wasnot satisfactory, we proceed with pre-processing of images.

5.1 RestNet152, InceptionV3 and InceptionResNetV2 results after pre-processing

Total training images are 7721 and 1786 images are used for testing to build all models. Model is built using combination of classification algorithms ResNet152, InceptionV3 and InceptionResNetV2 with binarized thresholding Otsu, Sauvola and Tozero. So performance evaluation of all 9 combinations are discussed in following sections.

5.1.1 ResNet152+Otsu, ResNet152+Sauvola Framework and ResNet152+To zero Framework

The three frameworks ResNet152+Otsu, ResNet152+Sauvola, ResNet152+To-zero frameworks are trained first for 10,15,20 and 30 epochs. As per observations, training accuracy is increasing impressively with very nominal change validation accuracy. Also, while training loss is decreasing but validation loss suddenly start falling. This shows the situation for overfitting. Comparative study performance of 3 algorithms discussed in Table 2. ResNet152+Tozero at 20 and 30 epochs performing better during training and testing as well.

Table 2: ResNet152 comparative test results before drop-out

Algorithm	No. of Epoch	Train_Acc	Avg_Test_Acc	Precision
ResNet152 + Otsu	10	96.00	87.90	0.75
	15	97.76	77.93	0.78
	20	99.84	91.60	0.81
	30	99.84	94.47	0.7939
ResNet152 + Sauvola	10	97.42	85.4	.59
	15	99.77	84.38	0.744
	20	99.81	85.40	0.75
	30	98.92	90.70	0.6567
ResNet152 + tozero	10	96.76	98.69	0.795
	15	96.45	97.06	0.74
	20	98.59	99.37	0.81
	30	98.90	99.73	0.8113

5.1.2 InceptionV3+Otsu, InceptionV3+Sauvola and InceptionV3+To zero Framework

The three frameworks InceptionV3+Otsu, InceptionV3+Sauvola, InceptionV3+To-zero frameworks are trained first for 10,15,20 and 30 epochs. As per observations, training accuracy is increasing impressively with very nominal change validation accuracy. Also while training loss is decreasing but validation loss suddenly start falling. This shows the situation for overfitting. Comparative study performance of 3 algorithms discussed in Table 3. InceptionV3+Tozero is performing good during training and testing as well.

Table 3: Comparative study of InceptionV3 Observations at different epochs

Algorithm	No. of Epoch	Train_Acc	Avg_Test_Acc	Precision
InceptionV3 + Otsu	10	99.82	95.03	0.79
	15	99.77	82.60	0.78
	20	99.84	92.75	0.81
	30	99.80	98.13	0.8113
InceptionV3 + Sauvola	10	99.82	83.06	0.67
	15	99.80	82.60	0.78
	20	99.80	83.74	0.73
	30	99.70	89.58	0.6517
InceptionV3 + tozero	10	98.92	99.68	0.80
	15	98.52	99.49	0.80
	20	98.90	99.69	0.80
	30	98.92	99.71	0.8029

5.1.3 InceptionResNetV2+Otsu, InceptionResNetV2+Sauvola and InceptionResNetV2+To zero Framework

The three frameworks InceptionV3+Otsu, InceptionV3+Sauvola, InceptionV3+To-zero frameworks are trained first for 10,15,20 and 30 epochs. As per observations, training accuracy is increasing impressively with very nominal change validation accuracy. Also, while training loss is decreasing but validation loss suddenly starts falling. This shows the situation for overfitting. Comparative study performance of 3 algorithms discussed in Table 4. InceptionV3+Tozero is performing pretty good during training and testing as well at 20 and 30 epochs.

Table 4: Comparative study InceptionResNetV2 Observations at different epochs

Algorithm	No. of Epoch	Train_Acc	Avg_Test_Acc	Precision
InceptionResNet-V2 + Otsu	10	91.17	83.96	0.77
	15	94.04	66.70	0.78
	20	98.75	90.68	0.7827
	30	98.59	91.78	0.789
InceptionResNet-V2 + Sauvola	10	93.04	76.71	0.64
	15	94.51	73.86	0.74
	20	98.78	83.58	0.78
	30	97.82	87.46	0.6825
InceptionResNet-V2 + To-zero	10	91.55	95.48	0.77
	15	86.65	92.80	0.78
	20	96.53	97.36	0.83
	30	98.49	98.84	0.824

After testing Modi words written by different set of people with the current build model by all above combination of algorithms, we are unable to get satisfactory results. So we decided to go regularization using Drop-out to minimize overfitting.

5.2 At 30 epochs performance with Drop-out and without drop-out

As we got good training and test results before applying drop-out till 30 epochs as shown in table 5, but model was unable to give satisfactory predition results for unknown word samples. Also there is huge difference between training and validation accuracy. Parallaly, training and validation loss is moving in opposite direction which indicates poor parameter tuning as shown in table 5. So we decided to treat the model using drop-out. But due to drop-out till 30 epochs we are unable to get good training results as shown in table 6. It pushes us to train the model with more number of epochs. After training the model step-wise we started getting the good training accuracy. From table 6, it is observed that, training accuracy is increasing stepwise in coordination with validation accuracy, this shows good hyper-parameter tuning while traing the model. Table 5 shows good training accuracy at 30 epochs but validation accuracy is increasing very slowly compared to validation accuracy shown in table 6 which is showing results after drop-out.

Table 5: Performance of all 3 algorithm Without Drop-out at 30 epochs

Algorithm	Without Drop-out			
	Tr-acc	Tr-loss	Val-acc	Val_loss
InceptionV3+otsu	0.998096	0.002919	0.040859	12.84438
InceptionV3+sauvola	0.997057	0.012719	0.047091	13.58673
InceptionV3+tozero	0.989268	0.025455	0.062327	10.98617
ResNet152+otsu	0.998442	0.002916	0.094183	14.78234
ResNet152+sauvola	0.989268	0.036535	0.078255	15.27467
ResNet152+tozero	0.989095	0.024092	0.09903	15.97871
IncResNetV2+otsu	0.985979	0.048614	0.246537	8.962371
IncResNetV2+sauvola	0.978189	0.063744	0.195291	10.13233
IncResNetV2+tozero	0.98494	0.051075	0.191828	11.77448

Table 6: Performance of all 3 algorithm with Drop-out at 30 epochs

Algorithm	With Drop-out			
	Tr-acc	Tr-loss	Val-acc	Val_loss
InceptionV3+otsu	0.287	2.219274	0.179363	3.476355
InceptionV3+sauvola	0.280595	2.270522	0.164127	3.620318
InceptionV3+tozero	0.249957	2.45479	0.171053	3.360918
ResNet152+otsu	0.527956	1.4283	0.310942	2.886939
ResNet152+sauvola	0.498702	1.530909	0.270083	3.566783
ResNet152+tozero	0.581963	1.266787	0.325485	3.122474
IncResNetV2+otsu	0.505799	1.512043	0.274931	3.079045
IncResNetV2+sauvola	0.715769	0.847796	0.467452	2.530933
IncResNetV2+tozero	0.75038	0.743861377	0.4972	2.456124306

5.3 Performance of training accuracy and loss at different epochs with different algorithm after Drop-out

This section discusses the results obtained after drop-out normalization. As we used drop-out techniques which makes the network sparse, so we need to experiment with more no. of epochs to get better results. Here we have done experiment with 30, 50, 100, 200 and 300 epochs as shown in section 5.3.1, 5.3.2 and 5.3.3. After training using drop-out technique, the model is showing proportionate increase in training and validation accuracy. At the same time, it is showing considerable consistent decrease in training and validation loss. This gives us intuition that the newly developed hybrid model helps to reduce overfitting and learns appropriately. The developed hybrid model will be tested on Modi words classification in the chapter 6 to implement incremental learning approach.

5.3.1 ResNet152 comparative test results after drop-out

This section discusses about the comparative performance of ResNet152 with Otsu, Sauvola and Tozero techniques at 10, 30,50,100 and 300 epochss respectively. Table 7, 8 and 9 shows the performance of training and test accuracy after drop-out using ResNet152 framework. From the results, it is observed that ResNet152+Tozero at 100 and 300 epochs gives good precision.

Table 7: ResNet152 + Otsu performance after drop-out

Algorithm	No. of Epoch	Train_Acc	Val_acc	Avg_Test_Acc	Precision
ResNet152 + Otsu	10	02.82	02.15	2.31	0.0392
	30	52.79	31.09	48.59	0.5145
	50	86.38	43.42	81.44	0.6601
	100	94.13	40.30	86.88	0.6853
	300	97.17	54.98	81.95	0.6052

Table 8: ResNet152 + Sauvola performance after drop-out

Algorithm	No. of Epoch	Train_Acc	Val_acc	Avg_Test_Acc	Precision
ResNet152 + Sauvola	10	05.02	04.64	4.49	0.0459
	30	49.87	27.00	50.77	0.3404
	50	84.32	40.65	77.20	0.6304
	100	94.59	43.63	90.25	0.6752
	300	97.04	36.22	87.39	0.6002

Table 9: ResNet152 + To-zero performance after drop-out

Algorithm	No. of Epoch	Train_Acc	Val_acc	Avg_Test_Acc	Precision
ResNet152+ Tozero	10	05.09	0.0685	11.50	0.0666
	30	58.19	32.54	76.39	0.5929
	50	81.99	30.96	88.51	0.6825
	100	94.82	47.09	96.24	0.8236
	300	96.37	45.08	97.38	0.7839

5.3.2 InceptionV3 comparative test results after drop-out

This section discusses about the comparative performance of InceptionV3 with To-zero, Sauvola and Tozero techniques at 10, 30,50,100 and 300 epochss respectively. Table 10, 11 and 12 shows the performance of training and test accuracy after drop-out using InceptionV3 framework. From the results, it is observed that InceptionV3+Tozero at 300 epochs gives better precision.

Table 10: InceptionV3 + Otsu Otsu performance after drop-out

Algorithm	No. of Epoch	Train_Acc	Val_acc	Avg_Test_Acc	Precision
InceptionV3 + Otsu	10	2.8	2.14	2.31	0.039
	30	28.7	17.93	29.54	0.245
	50	63.44	30.68	58.05	0.356
	100	88.31	37.67	77.67	0.5823
	300	97.21	32.75	91.10	0.7072

Table 11: InceptionV3 + Sauvola Otsu performance after drop-out

Algorithm	No. of Epoch	Train_Acc	Val_acc	Avg_Test_Acc	Precision
InceptionV3	10	2.7	2.28	4.52	0.0285

+ Sauvola	30	28.06	16.14	34.55	0.2021
	50	57.71	28.67	59.59	0.3992
	100	94.43	32.55	80.12	0.5055
	300	96.92	43.21	90.37	0.6859

Table 12: InceptionV3 + To-zero Otsu performance after drop-out

Algorithm	No. of Epoch	Train_Acc	Val_acc	Avg_Test_Acc	Precision
InceptionV3 + Tozero	10	2.38	2.0	7.04	0.0235
	30	24.99	17.10	45.09	0.2099
	50	62.09	27.98	77.94	0.5229
	100	94.23	43.07	93.63	0.6987
	300	97.92	49.72	98.14	0.8113

5.3.3 InceptionReNetV2 comparative test results after drop-out

This section discusses about the comparative performance of InceptionResNetV2 with Otsu, Sauvola and Tozero techniques at 10, 30,50,100 and 300 epochss respectively. Table 13, 14 and 15 shows the performance of training and test accuracy after drop-out using InceptionResNetV2 framework. From the results, it is observed that InceptionResNetV2+Otsu and InceptionResNetV2+Tozero at 300 epochs gives good precision.

Table 13: IncResNetV2+ Otsu Otsu performance after drop-out

Algorithm	No. of Epoch	Train_Acc	Val_acc	Avg_Test_Acc	Precision
IncResNetV2+ Otsu	10	07.89	09.62	9.31	0.1153
	30	50.58	27.49	42.92	0.4367
	50	85.33	47.85	80.15	0.7811
	100	94.13	53.05	84.81	0.7620
	300	97.99	57.62	94.38	0.8488

Table 14: IncResNetV2+ Sauvola Otsu performance after drop-out

Algorithm	No. of Epoch	Train_Acc	Val_acc	Avg_Test_Acc	Precision
IncResNetV2+ Sauvola	10	10.09	13.36	13.46	0.1472
	30	71.58	46.74	67.09	0.6433
	50	86.79	54.78	80.78	0.7318
	100	94.72	52.35	89.44	0.7648

	300	97.57	53.95	91.03	0.7726

Table 15: IncResNetV2+ To-zero Otsu performance after drop-out

Algorithm	No. of Epoch	Train_Acc	Val_acc	Avg_Test_Acc	Precision
IncResNetV2 + Tozero	10	10.94	12.81	27.79	0.1668
	30	75.04	49.72	88.37	0.7536
	50	87.22	52.15	95.05	0.8298
	100	93.80	52.71	95.98	0.8214
	300	97.56	55.47	98.16	0.8230

6. Modi word segmentation and prediction in marathi

As part of last module of the project, word are collected written by different people. Collected samples are segmented using active contour mehtod. The selected segmented individual words are classified using develeoped module. Figure 4 shows few word written in Modi.

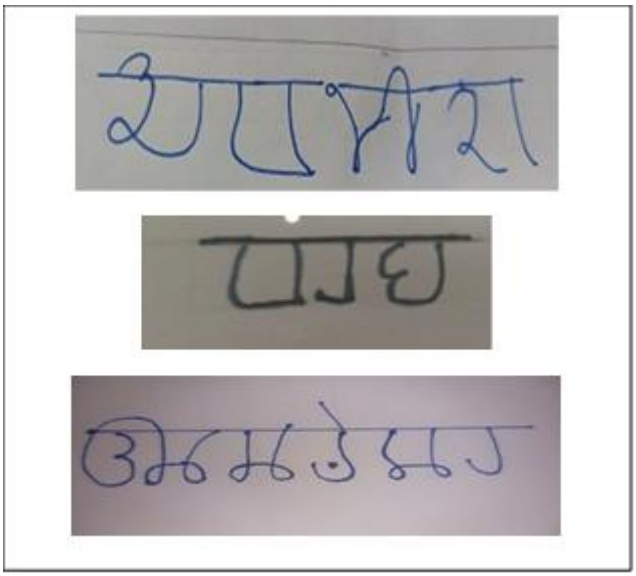

Figure 4: Un-processed sample Modi words

6.1 Word detection results

In this section, word detection in marathi is discussed using developed model for character recognition. In section 6.1, we have tested the few words namely “Varad”, “Desai” and “Vaibhav” using final hybrid model. Here all words are mostly classified correctly except the character “sa” which is in-correctly classified as banacha “na” sinch both the characters look similar as shown in table 16, 17, 18.

Table 16: Word “varad” is tested using InceptionResNetV2+Otsu algorithm at 300 epochs

File	Label	Accuracy
/content/drive/MyDrive/Modi Lipi/Dataset/Contours/Varad/Va.png	VA-varun	99.62
/content/drive/MyDrive/Modi Lipi/Dataset/Contours/Varad/Da.png	DA-daut	43.91
/content/drive/MyDrive/Modi Lipi/Dataset/Contours/Varad/Ra.png	RA-ravi	87.52

Table 17: Word “Desai” is testd using InceptionResNetV2+Otsu algorithm at 300 epochs

File	Label	Accuracy
/content/drive/MyDrive/Modi Lipi/Dataset/Contours/Desai/De.png	DE-daut	49.38
/content/drive/MyDrive/Modi Lipi/Dataset/Contours/Desai/Sa.png	NA-banacha	96.7
/content/drive/MyDrive/Modi Lipi/Dataset/Contours/Desai/ee.png	3 i-imarat	99.69

Table 18: Word “Vaibhav” is testd using InceptionResNetV2+Otsu algorithm at 300 epochs

File	Label	Accuracy
/content/drive/MyDrive/Modi Lipi/Dataset/Contours/Vaibhav/Va.png	VA-varun	51.27
/content/drive/MyDrive/Modi Lipi/Dataset/Contours/Vaibhav/Vai.png	VAI	22.72
/content/drive/MyDrive/Modi Lipi/Dataset/Contours/Vaibhav/Bha.png	BHA-bhatji	28.49

7. Conclusion and Future Scope

The developed model is tested with Modi lipi words to learn incrementally written by different set of people.

We have tried to developed an robust incremental learning model which is tested using Modi characters, and words written by different set of people. As of now for Modi lipi, people worked with only basic character set of consonants and vowels with deep learning approach, we are the first to work with whole Modi lipi barakhadi with 360 classes. Overall an interesting technical and non-technical journey to work with Modi character set. As part of future scope, we can work further on characters which are facing issue with classification with current model. Increase the size of dataset written by more people and more set of words.

The model can extend to test and upgrade as per need for testing handwritten Modi statements. As part of future scope, we can work further on characters which are facing issue with classification using current model.

The model can extend to test and upgrade as per need for testing handwritten Modi statements.

Acknowledgments

We are thankful to Bhartiya Itihas Sanshodhan Mandal, Pune (BISM) who had given chance to collect handwritten Modi data from different related people.

Authors contribution statement

Both the authors contributed equally to collect data, to design the system, to implement and analyse the system.

Data availability statement: Data used for the experiment is handwritten and created with help of writing by different people, friends.

References

[1] Ram, S., Gupta, S., & Agarwal, B. (2018). Devanagri character recognition model using deep convolution neural network. Journal of Statistics and Management Systems, 21(4), 593-599.

[2] C. Chandankhede, "Offline MODI script character recognition using deep learning techniques," 2023. Chandankhede, C., & Sachdeo, R. (2023). Offline MODI script character recognition using deep learning techniques. Multimedia Tools and Applications, 1-12.

[3] C. Chandankhede and R. Sachdeo, "Character Recognition using MODI script : Facts , Challenges and its future," TEST Engineering and Management, no. 25389, pp. 25389–25395, 2020.

[4] Alom, M. Z., Sidike, P., Hasan, M., Taha, T. M., & Asari, V. K. (2018). Handwritten bangla character recognition using the state-of-the-art deep convolutional neural networks. Computational intelligence and neuroscience, 2018. (Hindawi journal)

[5] Vaidya, R., Trivedi, D., Satra, S., & Pimpale, M. (2018, April). Handwritten character recognition using deep-learning. In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT) (pp. 772-775). IEEE.

[6] Jangid, M., & Srivastava, S. (2018). Handwritten devanagari character recognition using layer-wise training of deep convolutional neural networks and adaptive gradient methods. journal of imaging, 4(2), 41.

[7] Koyuncu, B., & Koyuncu, H. (2019). Handwritten Character Recognition by using Convolutional Deep Neural Network; Review.

[8] Zhai, S., Wu, H., Kumar, A., Cheng, Y., Lu, Y., Zhang, Z., & Feris, R. (2017). S3pool: Pooling with stochastic spatial sampling. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4970-4978).

[9] Wang, S. H., Tang, C., Sun, J., Yang, J., Huang, C., Phillips, P., & Zhang, Y. D. (2018). Multiple sclerosis identification by 14-layer convolutional neural network with batch normalization, dropout, and stochastic pooling. Frontiers in neuroscience, 12, 818.

[10] U. Pal, T. Wakabayashi, and F. Kimura, "Comparative study of Devnagari handwritten character recognition using different feature and classifiers," in Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, 2009, pp. 1111–1115. doi: 10.1109/ICDAR.2009.244.

[11] S. Chandure and V. Inamdar, "Handwritten MODI Character Recognition Using Transfer Learning with Discriminant Feature Analysis," IETE J. Res., 2021, doi: 10.1080/03772063.2021.1902867.

[12] Brust, C. A., Käding, C., & Denzler, J. (2020). Active and Incremental Learning with Weak Supervision. *KI-Künstliche Intelligenz*, 1-16.

[13] Ren, H., Wang, W., Qu, X., & Cai, Y. (2019). A new hybrid-parameter recurrent neural network for online handwritten chinese character recognition. Pattern Recognition Letters, 128, 400-406.

[14] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

[15] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[16] Gan, J., Wang, W., & Lu, K. (2020). Compressing the CNN architecture for in-air handwritten Chinese character recognition. Pattern Recognition Letters, 129, 190-197.

[17] Jindal, U., Gupta, S., Jain, V., & Paprzycki, M. (2020). Offline Handwritten Gurumukhi Character Recognition System Using Deep Learning. In *Advances in Bioinformatics, Multimedia, and Electronics*

Circuits and Signals (pp. 121-133). Springer, Singapore.

[18] Li, Z., Wu, Q., Xiao, Y., Jin, M., & Lu, H. (2020). Deep Matching Network for Handwritten Chinese Character Recognition. *Pattern Recognition*, 107471.

[19] Mustafa, W. A., & Kader, M. M. M. A. (2018, June). Binarization of document images: A comprehensive review. In Journal of Physics: Conference Series (Vol. 1019, No. 1, p. 012023). IOP Publishing.

[20] Guha, R., Das, N., Kundu, M., Nasipuri, M., & Santosh, K. C. (2020). Devnet: an efficient cnn architecture for handwritten devanagari character recognition. International Journal of Pattern Recognition and Artificial Intelligence, 34(12), 2052009.[27]

[21] Wang, Y., Xiao, W., & Li, S. (2021, April). Offline Handwritten Text Recognition Using Deep Learning: A Review. In Journal of Physics: Conference Series (Vol. 1848, No. 1, p. 012015). IOP Publishing.[28]

[22] Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." Thirty-first AAAI conference on artificial intelligence. 2017.

[23] Husham, S., Mustapha, A., Mostafa, S. A., Al-Obaidi, M. K., Mohammed, M. A., Abdulmaged, A. I., & George, S. T. (2020). Comparative analysis between active contour and otsu thresholding segmentation algorithms in segmenting brain tumor magnetic resonance imaging. Journal of Information Technology Management, 12(Special Issue: Deep Learning for Visual Information Analytics and Management.), 48-61.

[24] Raynaud, G., Chane, C. S., Jacob, P., & Histace, A. (2019, February). Active Contour Segmentation based on Histograms and Dictionary Learning for Videocapsule Image Analysis. In VISIGRAPP (4: VISAPP) (pp. 609-615).

[25] Dave, N. (2015). Segmentation methods for hand written character recognition. International journal of signal processing, image processing and pattern recognition, 8(4), 155-164.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, “Rethinking the Inception Architecture for Computer Vision”.

[27] A. Choudhary, “A Review of Various Character Segmentation Techniques for Cursive Handwritten Words Recognition,” 2014. [Online]. Available: http://www.irphouse.com.

[28] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, “Impact of deep learning-based dropout on shallow neural networks applied to stream temperature modelling,” Earth-Science Reviews, vol. 201. Elsevier B.V., Feb. 01, 2020. doi: 10.1016/j.earscirev.2019.103076.