JuktoMala: A Handwritten Bengali Consonant Conjuncts Dataset for Optical Character Recognition Using BiT-based M-ResNet-101x3 Architecture

Md. Mehedi Hasan*, Md. Ali Hossain†, Azmain Yakin Srizon‡ and Abu Sayeed§
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology, Rajshahi, Bangladesh
Emails: *mmehedihasann@gmail.com, †ali.ruet@gmail.com, ‡azmainsrizon@gmail.com, §abusayeed.cse@gmail.com

***Abstract*—Bengali, the seventh most spoken language in the world by the number of speakers, doesn't have a well-established Optical Character Recognition (OCR) system for handwritten texts. One of the major reasons behind this lacking is contributed to having no complete conjuncts database. No dataset available today covers all the conjunct characters that are used by authors around the globe. In this research, we prepared a complete dataset consisting of 292 consonant conjunct characters, which is the biggest consonant conjunct character dataset to date by the number of classes available in the literature to our knowledge. We applied Big Transfer-based M-ResNet-101x3 Deep Convolutional Neural Network (DCNN) which achieves 91.32% accuracy that outperforms the baseline EfficientNetB7 approach which obtained 81.05% accuracy.**

***Index Terms*—Big Transfer, M-ResNet-101x3, Convolutional Neural Network, Handwritten compound character, Transfer Learning**

I. Introduction

One of the most useful tool human created to make computers mimic human intelligence is Optical Character Recognition (OCR) which is converting an image of texts into machine-editable texts [1]. This means the computer that can do OCR, understand the language written on the paper, and can distinguish each letter, number, punctuation mark, etc. This closes the gap between machine and human intelligence. Having a good OCR system can make a huge impact on the day-to-day life of general people as well as researchers. OCR system can be used to make historical documents searchable which will allow researchers to gain insights from those historical documents easily and this will also make them easy to preserve for future generations.

Various languages in this world now have a successful implementation of OCR systems that can convert both printed and handwritten texts into machine-editable texts, such as English. However, this is not true for all languages since some languages are cursive and complex and have more than one representation of the same character. This fact contributes to the lacking of a good OCR system for some languages including Bengali, which doesn't have a very accurate OCR for handwritten texts. The main reason behind this is the lack of a complete dataset that covers all the vowels, consonants, numbers, punctuation, and most important part: consonant conjuncts. Bengali has 292 consonant conjuncts which are being used by authors in the current and recent literature [2]. There are some other factors too for example high variation from person to person, different styles for the same character, cursive nature, etc. All these factors contributed to the lack of a good OCR system for handwritten characters in the Bengali language.

In this research paper, we created a new dataset called JuktoMala which contains 292 classes, each class having 10 characters of consonant conjuncts which are being used in the current literature and we also applied a modified Efficient-NetB7 based CNN as a baseline which achieves good accuracy considering very few available data.

II. Literature Review

In the domain of OCR, several researchers are accomplishing major contributions to various languages available in the world. In the literature, we can see evidence of the amazing work of various languages. English handwritten characters [3], [4], Spanish handwritten characters [5], [6], Hindi handwritten characters [7], [8], etc. got attention from researchers. It is a matter of regret that, the Bengali language did receive some attention from researchers which is not significant enough. And there have been very few numbers of literature that addressed the complex task of recognizing consonant conjuncts or compound characters. In Indo-European languages, compound characters are a common feature. There can be a significant number of compound characters in a language such as the Bengali language has 292 different compound characters [2]. Scientists implemented basic Bengali character recognition for some time now. Early works suggest that the basic 50 characters consisting of 39 consonants and 11 vowels have got some good attention. In the early works, Multi-Layer Perceptron (MLP) on stroke feature of Bengali basic characters has been used with an overall accuracy of 84.33% [9]. Another study has achieved 92.14% accuracy using a chain code histogram [10]. These early works were on 50 basic characters only which led to the development of CMATERdb [11], [12], a Bengali handwritten character dataset that contains the basic 50 characters, as well as numerals, modifiers, and compound characters. There has been huge attention to this dataset due to its diversity and features. Some research still used only 50 basic characters from this dataset but suggested a deep

979-8-3503-4536-0/23/$31.00 ©2023 IEEE

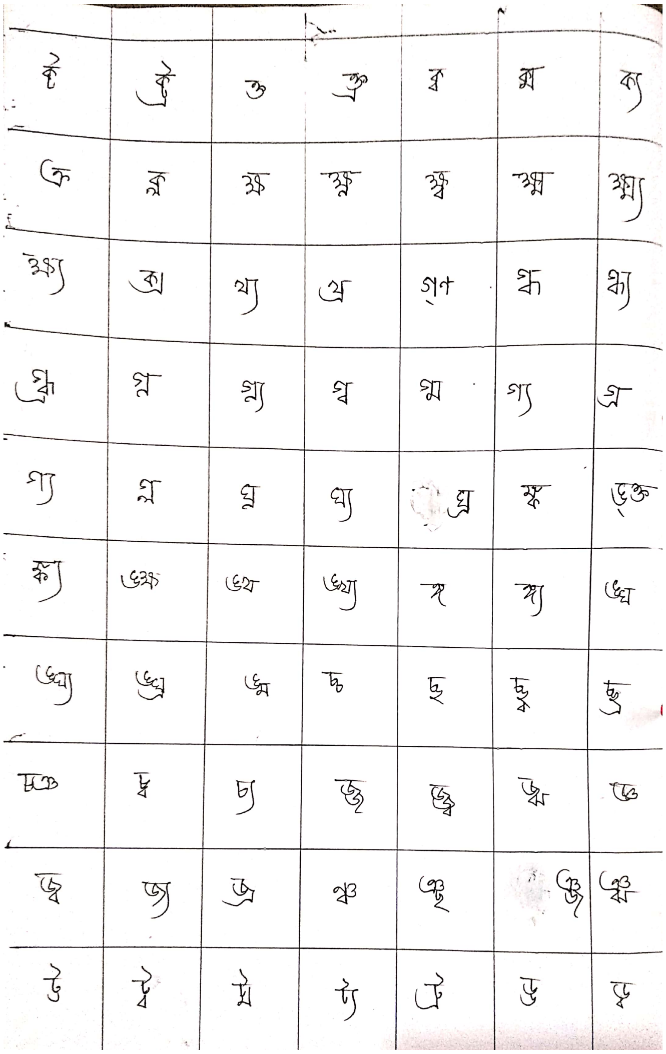

Fig. 1: Sample of collected data in form

convolutional neural network [13]. Later some researchers applied MobileNetV1 to the basic 50 characters[14].

Hybrid HOG-based CNN has been applied by Sharif et. al who achieved 92.57% and 92.77% accuracy for 171 and 199 classes respectively [15]. Ashiquzzaman et al. applied a deep CNN-based approach using ELU and dropout considering 8000 test images and 34000 train images and achieved 93.68% overall accuracy[16]. Deep CNN with point-light source-based shadow features and histogram of oriented pixel positions features was used recently[17]. The Ensemble technique has been investigated as well[18]. Some researchers used deep CNN and outperformed previous studies while considering basic characters, numerals, modifiers, and compound characters[19]. The authors of CMATERdb also applied the soft computing paradigm in a two-pass approach and achieved 87.26% overall accuracy for 256 classes[20]. In this research, we created a new dataset named JuktoMala which contains 292 compound characters which are the most by number of classes in a Bengali handwritten character dataset. We also created an

Fig. 2: Data processing steps

EffiecientNetB7-based Deep CNN which is applied to this dataset. We compared our result with the baseline EfficientNetB7 approach on this dataset.

III. Materials and Methods

This section contains the description of our dataset and the collection process through which the data has been gathered, augmentation, dataset preparation, and attention-based deep CNN architecture.

A. Data Collection Process

The dataset was collected from the students of the Computer Science & Engineering department of Rajshahi University of Engineering & Technology. Both male and female participants took part in the data collection process. They wrote each compound character from the 292 classes of compound characters and those were processed by us later. A sample of such data collection form (filled) is illustrated in Figure 1.

B. Data Processing

Every scanned paper from our participant was fed into data processing steps where we carefully crop out every single compound character individually. Then the images were transformed into binary using a threshold value of 127. After that, we apply the trimming process to reduce the unnecessary blank background areas from those individual images. And finally, they were resized into $256 \times 256 \times 3$ resolution since the model requires RGB image.

C. Dataset Description

There are 292 classes of compound characters in this JuktoMala dataset, each class having 10 images, totaling 2920 images. All the images are grayscale. The dataset is partitioned into the test, train, and validation datasets. The test set has 30% of the dataset. The rest 70% of the total dataset was augmented using the following parameter shown in Table III. After that, the augmented dataset was split again into 80:20 portions from which 80% of the dataset was used as the train set and 20% of the data was used as the validation set. This process has been illustrated in Figure 2. The original dataset can be accessed at https://drive.google.com/file/d/1ngbplIL48_E6MSZnMJN4taJOFnLTvL4o/view?usp=share_link

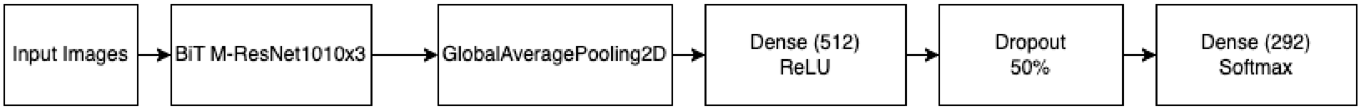

Fig. 3: Proposed Big Transfer-based M-ResNet101x3 architecture

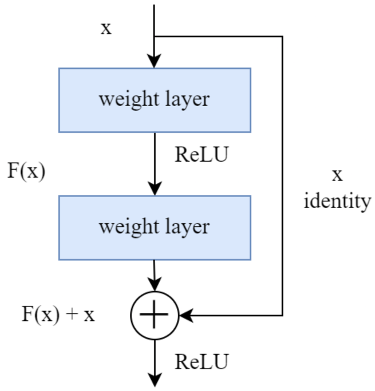

Fig. 4: Residual block of ResNet architecture

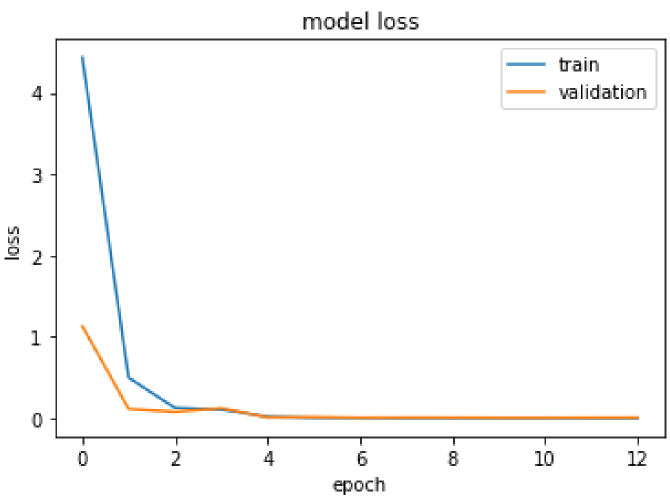

Fig. 5: Loss curve of BiT-based M-ResNet 101x3 model

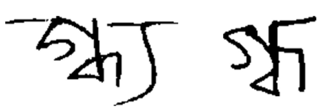

Fig. 6: Misclassification due to similar classes

D. Proposed DCNN Architecture

We have created an M-ResNet 101 × 3 big transfer-based deep convolutional neural network, which is shown in Figure 3. The input image will be fed into the CNN backbone which extracts features from image. This backbone implements the M-Resnet101 × 3 architecture, trained to perform image classification on ImageNet ILSRCV-2012-CLS, a dataset containing around 1.3 million images labeled with 1,000 classes. Its outputs are the 6144-dimensional feature vectors, before the multi-label classification head. Global Average Pooling was added to get the average of the output feature vector, which further reduces the dimension of the feature vector. The output of this will be fed into a dense layer with 512 neurons with Rectified Linear Unit (ReLU) being used. ReLU is very simple and easy to calculate. Its derivative is also easy to calculate. To prevent overfitting, a dropout layer with 50% dropout has been used which will randomly turn off 50% of the neuron during the training period. This will force the network to explore the new unexplored path. And finally, the output layer with 292 neurons for 292 classes with a SoftMax activation layer has been used. We will now discuss Medium ResNet 101 with 3 times wider CNN architecture in detail.

E. Resnet 101

Convolutional Neural Network (CNN) captures the spatial and temporal features from an image using the convolution operation shown in the following equation.

$$g(x,y) = w * f(x,y) = \sum_{dx=-1}^{a} \sum_{dy=-b}^{b} w(dx,dy) f(x-dx, y-dy)$$

where g(x,y) is the filtered or convolved image, f(x,y) is the original image, and w is the filter kernel.

ResNet or residual network is an outcome of Microsoft research which was invented in search of creating more deep CNN without facing vanishing gradient problems [21]. They did it by adding some residual connection between some layers which helps calculate the gradient during backpropagation. ResNet between Resnet 50 and Resnet 152 in terms of layers, performance, and accuracy.

Big Transfer is a google research project where they used group normalization instead of batch normalization and weight standardization and some heuristic approach to creating a transfer learning mechanism that performs well even when there is only one image per class to 1 million images per class[22]. It scales well and achieves state-of-the-art accuracy in various classification and related tasks. The residual block of ResNet architecture is illustrated in Figure 4.

F. Hyperparameters

A learning rate of 0.00001 has been utilized in this research. The batch size was set to 6. Adam optimizer has been used. A total of 50 epochs were set to run initially. However, early stopping was enabled with patience set to 3 and learning rate reduction on the plateau of patience 1 with a factor of 0.2.

TABLE I: Classwise accuracy (for 0 to 200 classes)

Class	Accuracy	Class	Accuracy	Class	Accuracy
0	100	67	100	134	100
1	100	68	66.67	135	100
2	100	69	100	136	66.67
3	100	70	66.67	137	100
4	100	71	100	138	100
5	100	72	66.67	139	0
6	100	73	100	140	100
7	100	74	100	141	100
8	100	75	66.67	142	66.67
9	100	76	100	143	100
10	100	77	100	144	100
11	100	78	100	145	100
12	100	79	100	146	100
13	100	80	100	147	100
14	100	81	100	148	100
15	100	82	66.67	149	100
16	100	83	100	150	100
17	100	84	33.33	151	66.67
18	100	85	100	152	100
19	100	86	100	153	100
20	100	87	100	154	100
21	100	88	100	155	100
22	100	89	66.67	156	100
23	100	90	100	157	100
24	100	91	100	158	100
25	100	92	100	159	33.33
26	100	93	100	160	100
27	100	94	100	161	66.67
28	100	95	100	162	100
29	100	96	100	163	100
30	100	97	100	164	33.33
31	100	98	100	165	100
32	100	99	100	166	100
33	100	100	100	167	66.67
34	100	101	100	168	66.67
35	66.67	102	66.67	169	100
36	100	103	100	170	33.33
37	100	104	100	171	100
38	100	105	33.33	172	100
39	100	106	100	173	66.67
40	100	107	100	174	100
41	33.33	108	100	175	100
42	100	109	66.67	176	33.33
43	100	110	100	177	100
44	100	111	100	178	100
45	100	112	100	179	100
46	33.33	113	66.67	180	100
47	100	114	100	181	100
48	100	115	100	182	100
49	100	116	100	183	100
50	100	117	100	184	66.67
51	100	118	100	185	100
52	66.67	119	100	186	100
53	100	120	33.33	187	100
54	66.67	121	100	188	66.67
55	33.33	122	66.67	189	100
56	100	123	100	190	100
57	100	124	100	191	100
58	100	125	100	192	100
59	66.67	126	100	193	100
60	100	127	100	194	100
61	100	128	100	195	100
62	100	129	33.33	196	100
63	100	130	100	197	100
64	100	131	100	198	100
65	100	132	100	199	100
66	33.33	133	100	200	100

TABLE II: Classwise accuracy (for 201 to 291 classes)

Class	Accuracy	Class	Accuracy
201	100	247	100
202	100	248	100
203	100	249	100
204	100	250	100
205	100	251	100
206	100	252	100
207	100	253	100
208	100	254	100
209	33.33	255	66.67
210	100	256	33.33
211	100	257	0
212	66.67	258	100
213	100	259	100
214	100	260	100
215	100	261	66.67
216	100	262	100
217	100	263	100
218	100	264	100
219	33.33	265	100
220	100	266	66.67
221	100	267	100
222	66.67	268	33.33
223	100	269	100
224	100	270	100
225	66.67	271	100
226	100	272	66.67
227	100	273	100
228	100	274	100
229	66.67	275	100
230	33.33	276	100
231	100	277	33.33
232	100	278	100
233	100	279	100
234	100	280	100
235	100	281	100
236	100	282	100
237	100	283	66.67
238	100	284	100
239	100	285	66.67
240	100	286	100
241	100	287	100
242	100	288	66.67
243	100	289	100
244	100	290	100
245	100	291	100
246	100		

TABLE III: Augmentation parameters

Parameter Name	**Parameter Value**
Distortion	30% probability, grid size=3x3
Rotation	30% probability, max_rotation=3
Zoom	30% probability, zoom range = 1.1-1.5
Resize	100% probability, resolution=256x256

TABLE IV: Values of hyperparameters used in this research

SL	Name of hyperparameter	Value of hyperparameter
1	Learning rate	0.00001
2	Batch size	6
3	Optimizer	Adam
4	Epoch	50
5	Loss function	Categorical Crossentropy
6	Callbacks	Early Stopping with patience 3, Learning rate reduction on Plateau of patience 1 with factor 0.2

The categorical cross-entropy function was used as the loss function. The hyperparameters used in this research work are listed in Table IV.

IV. Results

With the hyperparameters and the architecture, we achieved 800 accurate predictions out of 876 test images which give us an accuracy of 91.32%. The detailed classwise accuracy for each of the 292 classes are shown in table-I and table-II. Our proposed architecture achieved very good results on most of the classes. In some classes, the model misclassified some characters due to similarity with other classes. Only in 3 classes, the model failed to classify any test data correctly. The reasoning in mentioned in the next section. To compare our result with a baseline approach, we used EfficientNetB7 and replaced the Big Transfer model with EfficientNetB7. We ran the code using similar architectures and hyperparameters. We got an accuracy of 81.05% for the baseline EfficientNetB7 approach. The loss curve for our proposed approach is given in Figure 5.

We can see that the model quickly converges and flatlines the curves and forces an early stopping callback. So, the model is stable in training.

V. Comments on Misclassifications

Most of the misclassifications are happening due to the similar curvatures of some classes. Figure 6 illustrates such an example. There are almost 30 classes that are similar to one another. The model is getting confused among these classes while recognizing. Therefore, misclassifications are occuring. In three classes, the misclassification rate is high for this reason.

VI. Conclusion

We prepared a complete dataset called JuktoMala, containing 292 number of compound characters popularly used in bengali literature. We have accumulated 2920 samples, containing 10 samples from each classes. We have labeled the dataset properly. A deep CNN model based on Big Transfer based M-ResNet 101x3 architecture was developed to classify the dataset. Our proposed architecture shows very good performance achieving 91.32% accuracy on the test set beating EfficientNetB7 by more than 10% in terms of accuracy. In future, more dataset can be collected and thus the performance of the model might be improved.

References

[1] J. Memon, M. Sami, R. A. Khan, and M. Uddin, "Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr)," *IEEE Access*, vol. 8, pp. 142 642–142 668, 2020.

[2] M. Ali, *Bangla Academy Bengali-English Dictionary*. Bangla Academy, 2021.

[3] S. R. Zanwar, U. B. Shinde, A. S. Narote, and S. P. Narote, "Handwritten english character recognition using swarm intelligence and neural network," in *Intelligent Systems, Technologies and Applications*. Springer, 2020, pp. 93–102.

[4] V. Sathya Narayanan and N. Kasthuri, "An efficient recognition system for preserving ancient historical documents of english characters," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 6, pp. 6275–6283, 2021.

[5] E. Granell, E. Chammas, L. Likforman-Sulem, C.-D. Martínez-Hinarejos, C. Mokbel, and B.-I. Cîrstea, "Transcription of spanish historical handwritten documents with deep neural networks," *Journal of Imaging*, vol. 4, no. 1, p. 15, 2018.

[6] C. Boufenar, M. Batouche, and M. Schoenauer, "An artificial immune system for offline isolated handwritten arabic character recognition," *Evolving Systems*, vol. 9, no. 1, pp. 25–41, 2018.

[7] J. Mukhoti, S. Dutta, and R. Sarkar, "Handwritten digit classification in bangla and hindi using deep learning," *Applied Artificial Intelligence*, vol. 34, no. 14, pp. 1074–1099, 2020.

[8] S. P. Deore and A. Pravin, "Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset," *Sādhanā*, vol. 45, no. 1, pp. 1–13, 2020.

[9] T. K. Bhowmik, U. Bhattacharya, and S. K. Parui, "Recognition of bangla handwritten characters using an mlp classifier based on stroke features," in *International Conference on Neural Information Processing*. Springer, 2004, pp. 814–819.

[10] U. Bhattacharya, M. Shridhar, and S. K. Parui, "On recognition of handwritten bangla characters," in *Computer vision, graphics and image processing*. Springer, 2006, pp. 817–828.

[11] N. Das, S. Basu, R. Sarkar, M. Kundu, M. Nasipuri *et al.*, "An improved feature descriptor for recognition of handwritten bangla alphabet," *arXiv preprint arXiv:1501.05497*, 2015.

[12] N. Das, S. Basu, R. Sarkar, M. Kundu, M. Nasipuri, and D. Basu, "Handwritten bangla compound character recognition: Potential challenges and probable solution." in *IICAI*, 2009, pp. 1901–1913.

[13] S. Mondal and N. Mahfuz, "Convolutional neural networks based bengali handwritten character recognition," in *International Conference on Cyber Security and Computer Science*. Springer, 2020, pp. 718–729.

[14] T. Ghosh, M. M.-H.-Z. Abedin, S. M. Chowdhury, Z. Tasnim, T. Karim, S. S. Reza, S. Saika, and M. A. Yousuf, "Bangla handwritten character recognition using mobilenet v1 architecture," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 6, pp. 2547–2554, 2020.

[15] S. Sharif, N. Mohammed, S. Momen, and N. Mansoor, "Classification of bangla compound characters using a hog-cnn hybrid model," in *Proceedings of the International Conference on Computing and Communication Systems*. Springer, 2018, pp. 403–411.

[16] A. Ashiquzzaman, A. K. Tushar, S. Dutta, and F. Mohsin, "An efficient method for improving classification accuracy of handwritten bangla compound characters using dcnn with dropout and elu," in *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. IEEE, 2017, pp. 147–152.

[17] S. Ghosh, A. Chatterjee, P. K. Singh, S. Bhowmik, and R. Sarkar, "Language-invariant novel feature descriptors for handwritten numeral recognition," *The Visual Computer*, vol. 37, no. 7, pp. 1781–1803, 2021.

[18] R. Sarkhel, N. Das, A. Das, M. Kundu, and M. Nasipuri, “A multi-scale deep quad tree based feature extraction method for the recognition of isolated handwritten characters of popular indic scripts,” *Pattern Recognition*, vol. 71, pp. 78–93, 2017.
[19] P. Keserwani, T. Ali, and P. P. Roy, “Handwritten bangla character and numeral recognition using convolutional neural network for low-memory gpu,” *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 12, pp. 3485–3497, 2019.
[20] N. Das, R. Sarkar, S. Basu, P. K. Saha, M. Kundu, and M. Nasipuri, “Handwritten bangla character recognition using a soft computing paradigm embedded in two pass approach,” *Pattern Recognition*, vol. 48, no. 6, pp. 2054–2071, 2015.
[21] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[22] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, “Big transfer (bit): General visual representation learning,” in *European conference on computer vision*. Springer, 2020, pp. 491–507.