applied sciences

MDPI

Article

BengaliNet: A Low-Cost Novel Convolutional Neural Network for Bengali Handwritten Characters Recognition

Abu Sayeed [1], Jungpil Shin [2,*], Md. Al Mehedi Hasan [2], Azmain Yakin Srizon [1] and Md. Mehedi Hasan [1]

[1] Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi 6204, Bangladesh; abusayeed.cse@gmail.com (A.S.); azmainsrizon@gmail.com (A.Y.S.); mmehedihasann@gmail.com (M.M.H.)

[2] School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu, Fukushima 965-8580, Japan; mehedi@u-aizu.ac.jp

* Correspondence: jpshin@u-aizu.ac.jp

Featured Application: Optical Character Recognition for Handwritten Bengali Numerals, Basic and Compound Characters.

Abstract: As it is the seventh most-spoken language and fifth most-spoken native language in the world, the domain of Bengali handwritten character recognition has fascinated researchers for decades. Although other popular languages i.e., English, Chinese, Hindi, Spanish, etc. have received many contributions in the area of handwritten character recognition, Bengali has not received many noteworthy contributions in this domain because of the complex curvatures and similar writing fashions of Bengali characters. Previously, studies were conducted by using different approaches based on traditional learning, and deep learning. In this research, we proposed a low-cost novel convolutional neural network architecture for the recognition of Bengali characters with only 2.24 to 2.43 million parameters based on the number of output classes. We considered 8 different formations of CMATERdb datasets based on previous studies for the training phase. With experimental analysis, we showed that our proposed system outperformed previous works by a noteworthy margin for all 8 datasets. Moreover, we tested our trained models on other available Bengali characters datasets such as Ekush, BanglaLekha, and NumtaDB datasets. Our proposed architecture achieved 96–99% overall accuracies for these datasets as well. We believe our contributions will be beneficial for developing an automated high-performance recognition tool for Bengali handwritten characters.

Keywords: banglalekha dataset; bengali handwritten character recognition; CMATERdb dataset; computer vision; dropout; ekush dataset; low-cost convolutional neural network; NumtaDB dataset; pattern recognition; supervised learning

Citation: Sayeed, A.; Shin, J.; Hasan, M.A.M.; Srizon, A.Y.; Hasan, M.M. BengaliNet: A Low-Cost Novel Convolutional Neural Network for Bengali Handwritten Characters Recognition. *Appl. Sci.* **2021**, *11*, 6845. https://doi.org/10.3390/app11156845

Academic Editor: Seokwon Yeom

Received: 14 June 2021
Accepted: 23 July 2021
Published: 25 July 2021

1. Introduction

Optical Character Recognition (OCR) is the mechanical or electrical conversion of pictures of handwritten or printed typed text into machine-encoded texts from either a scanned document or an image of the document [1]. It is widely practiced to recognize the texts in number plates of vehicles, passports, historical documents, business cards, printed materials, bank checks, etc. In this research, we focused on handwritten character recognition as it is more complex to recognize handwritten characters with various writing fashions than typed characters with one specific writing fashion. The main barrier to handwritten character recognition is the variation of literature worldwide. As of 2020, there are nearly 7117 languages in the world [2]. Each of these languages has its own alphabet or letters. Hence, language-specific character recognition has been an area of interest for a long time now. Beforehand, while some of the languages have received many contributions in the area of handwritten character recognition, others did not receive any significant contributions. In this study, we focused on Bengali handwritten character recognition.

The reason for choosing Bengali literature and the challenges of character recognition for the Bengali language have been presented in the following two paragraphs.

Bengali, an Indo-Aryan language, is the seventh most-spoken language worldwide and fifth most-spoken native language globally with nearly 228 million native speakers and approximately 37 million second-language speakers [3–5]. Bengali is the most widely spoken language in Bangladesh, with 98% of Bangladeshis being fluent in Bengali as their primary language and for this reason, Modern Standard Bengali is the national language of Bangladesh [6]. Moreover, Bengali is the second most-spoken language among the 22 scheduled languages of India just behind Hindi and the official language of West Bengal, Tripura, and the Barak Valley of Assam states of India. Moreover, it is the most-spoken language in the Andaman and Nicobar Islands located at the Bay of Bengal [7] and is practiced by notable speakers in Uttarakhand, Nagaland, Mizoram, Meghalaya, Jharkhand, Chhattisgarh, Delhi, and Arunachal Pradesh states of India [7]. Bengali is the only language in the world for which people sacrificed their lives on 21 February 1952 and after the independence of Bangladesh in 1971, UNESCO recognized 21 February as the International Mother Language Day in 1999 [8]. As it is one of the most-spoken, popular, and precious languages of the world, Bengali handwritten character recognition has been an area of interest for researchers for decades now.

The phenomenon which turned Bengali handwritten character recognition into a complex problem is none other than the number of characters in Bengali literature. Unlike other popular languages, Bengali has not only 50 basic characters but also has many compound characters which makes the total number of characters in Bengali literature approximately 400 [9]. Although the recent development in the domains of Transfer Learning and Deep Learning, 1000 classes can be taken care of by using complex neural networks, still Bengali character recognition is a tough task as Bengali characters are more complex and cursive in shape, especially the compound characters. Moreover, because it has a huge number of characters, some of the characters are extremely similar and some characters have multiple writing fashions which makes the recognition process more complex. Therefore, it is a profound domain of interest.

In this research, first, we chose publicly available and one of the most popular benchmark datasets, the CMATERdb Bengali handwritten character dataset for the training phase which contains 50 basic characters, 7 modifiers, 171 unique compound characters with 199 different writing styles, and 10 numerals of Bengali literature. After choosing the datasets, we applied some necessary preprocessing steps on the datasets and prepared different formations of the datasets as per the previous studies and conveniences. After that, we applied our proposed low-cost novel deep convolutional architecture, BengaliNet, along with hyperparameters tuning. Then, we measured the performance of the classifiers in terms of class-wise and overall precision, recall, f1-score, and accuracy. Finally, we compared the results with the previous studies and investigations. Experimental and comparative findings proved that our proposed architecture outperformed previously achieved highest performance and therefore, can recognize the Bengali handwritten characters more accurately. Additionally, our proposed architecture achieved decent overall accuracies while testing our trained models on other available Bengali datasets i.e., Ekush, BanglaLekha, and NumtaDB. Contributions of our research have been more precisely described in the "Our Contributions" section.

2. Literature Review

Previously, various studies have been conducted for the accurate recognition of the handwritten characters of some popular languages, for example, English handwritten characters [10,11], Chinese handwritten characters [12,13], Spanish handwritten characters [14,15], Hindi handwritten characters [16,17] and so on. However, not a significant amount of studies have been conducted on the Bengali handwritten characters. Moreover, very few contributions are notable in the recognition process of Bengali compound characters. Similar and popular non-Latin languages such as Arabic [18], Chinese [12,13],

Farsi [19], Greek [20], Hindi [16,17], Japanese [21], Korean [22], Nepali [23], Russian [24], Thai [25], Urdu [26], etc. have received contributions in the domain of handwritten character recognition. Compound characters are the common property of Indian scripts among which Devanagari and Bengali have a huge set of compound characters [27]. Sometimes the shape of the compound character is so complex that it becomes difficult to identify the constituent consonants [27]. Languages that use the Devanagari script such as Hindi [27] suppress the vowel most of the time to bypass the use of compound characters [28] and as a result, few significant works have been conducted regarding Devanagari handwritten compound character recognition previously. However, compound characters are frequently used in Bengali scripts. Therefore, it is important to design a model that can recognize this huge number of compound characters which are highly stylized and morphologically complex. In this section, we have mentioned only the noteworthy studies which paved away the path of the recognition process.

Researchers started to implement Bengali basic character recognition long before. Early research on Bengali 50 basic character recognition suggested a Multilayer Perceptron (MLP) technique on stroke features and achieved the highest overall accuracy of 84.33% [29]. The number of samples used for training and testing was 350 and 90 for each of the 50 classes. In another research, the authors applied a Multilayer Perceptron (MLP) classifier on the 76 extracted features (36 longest-run features, 16 centroid features, and 24 shadow features) and achieved an overall accuracy of 75.05% [30]. A total of 10,000 images of size 64×64 of 50 classes were used in this research and the train-test ratio was 80:20. Later, the study of chain code histogram features suggested an overall accuracy of 92.14% by applying a Multilayer Perceptron (MLP) classifier [31]. In this research, the authors used 200 samples for each of the 50 classes for training and 10,187 samples for testing where each of the test classes had at least 60 samples per class. The images were scanned at 300 dpi resolution.

After that, CMATERdb datasets were introduced [32,33] which not only contained Bengali basic characters but also contained compound characters, numerals, and modifiers. Since then, a notable number of studies have been conducted on different CMATERdb datasets. One research suggested a deep convolutional neural network with a learning rate of 0.001 and the train-validation ratio of 80:20 for the recognition of 50 basic Bengali handwritten characters [34]. Later, MobileNetV1 architecture was used for recognizing the basic Bengali handwritten characters [35]. Sharif et al. applied a Hybrid-HOG-based Convolutional Neural Network approach and achieved 92.57% and 92.77% accuracy for 171 and 199 Bengali compound characters respectively [36]. Ashiquzzaman et al. suggested a deep convolutional network-based approach with ELU and dropout to achieve an overall accuracy of 93.68% although they considered 8,000 test images and 34,000 train images [37]. Although a significant amount of research has not been conducted on Bengali basic and compound characters, a significant amount of works has been conducted on the Bengali numerals recognition. A recent study suggested a Deep Convolutional Neural Network on point-light source-based shadow features and histogram of oriented pixel positions features [38]. Ensemble learning was used in one of the studies as well [39]. Another recent work suggested a Convolutional Neural Network-based architecture considering a lesser number of trainable parameters, applied their proposed model on Bengali numerals, basic characters, compound characters, modifiers, and the dataset of all 256 characters and outperformed previous studies [40]. Another research that worked with the class-wise performance of 256 classes previously, achieved an overall accuracy of 87.26% using a soft computing paradigm embedded in a two-pass approach, and this work was suggested by the original authors of the CMATERdb datasets [41]. In this research, we worked with the CMATERdb datasets and outperformed all these previous studies.

Moreover, there have been some studies where the authors have focused on developing cost-effective Bengali character recognition architectures. One study suggested a multiobjective approach towards cost-effective Bengali handwritten characters recognition [42]. Another research suggested multiobjective optimization for recognition of

handwritten Indic scripts [43]. Furthermore, transfer learning has been used by applying various architectures i.e., AlexNet, VGG-16, ResNet-50, etc. beforehand [44,45]. Our proposed architecture has not only outperformed all these studies in terms of overall accuracy but also in terms of the number of parameters as well. Details on experimental results and comparative analysis can be discovered in the "Experimental Analysis" section.

3. Our Contributions

In this research, we have proposed a low-cost novel deep convolutional neural network architecture, BengaliNet, for Bengali handwritten character recognition. The architecture uses 2.24 million to 2.43 million parameters only, based on the number of output classes (7, 10, 50, 171, 199, and 256). The reason for focusing on a fewer number of parameters is that Bengali handwritten character recognition is supposed to be an application for mobile devices. Hence, low-cost models are suitable for such applications. The existing architectures in previous research used more parameters than ours. For example, VGG-16, Alex Net, and ResNet-50 have been implemented in previous research and used almost 138 million, 61 million, and 23 million parameters [44,45]. MobileNetV1 architecture uses the least number of parameters that is a total of 4.2 million parameters, but this architecture traded off low-cost implementation with accuracy, hence, produced a lower accuracy [35].

However, the latest work in the domain of Bengali handwritten character recognition used 4.75 million parameters for training by not trading off the overall accuracy [40]. This work outperformed all previous studies and the authors compared their architecture with AlexNet, VGGNet, and GoogLeNet as well. They used 64×64 size input images for their research. In our research, we used a novel deep convolutional neural network with an input size of 28×28. As the main focus of this research is to design a low-cost high-performance network for mobile devices, the input size was kept as small as possible to minimize the overall number of parameters without sacrificing the performance. We not only used only 2.24 to 2.43 million parameters (almost half parameters of the latest research) but also outperformed the latest work by a fine margin as well. Details on our architecture can be found in the "Materials and Methods" section and comments on why our proposed architecture is novel and works better can be discovered in the "Experimental Analysis" section.

Further inspection of misclassifications revealed that almost all the misclassifications were occurring due to some common issues. Some of the classes had highly similar samples, some of the classes had multiple writing fashions and some of the classes had some noisy samples that are even difficult to recognize by human eyes. With proper examples and figures, misclassifications have been discussed in the "Experimental Analysis" section. We believe commenting on misclassifications will further showcase the efficiency of our proposed architecture and significantly justify our contributions. To measure the efficiency and impact of the proposed architecture in developing Bengali handwritten character recognition tools, other Bengali datasets of different distributions were used for testing i.e., Ekush, BanglaLekha, and NumtaDB. These datasets were not used while training, hence, the samples are unknown to the classifier. Despite coming from different distributions, our architecture achieved an overall accuracy of 98.36%, 99.13%, 96.09%, 98.11%, and 97.50% for Ekush basic and numerals, BanglaLekha basic and numerals, and NumtaDB numerals respectively. Thus, we believe our contributions will be significant in the domain of Bengali handwritten character recognition.

4. Materials and Methods

This section contains explanations on dataset description, augmentation, dataset preparation, deep convolutional neural network, and the proposed BengaliNet architecture.

4.1. Dataset Description

In this research, we considered the CMATERdb datasets [41] for training. The datasets considered in this research are Bengali Numerals (CMATERdb 3.1.1), Bengali basic charac-

ters (CMATERdb 3.1.2), Bengali compound characters of 2 versions (CMATERdb 3.1.3.1 and CMATERdb 3.1.3.3), and Bengali modifiers (CMATERdb 3.1.4). For all the datasets, the training set and the test set were provided separately. Only the training set was split into the train and validation set. The test set was independent and had no impact on training whatsoever. CMATERdb 3.1.1 has 10 classes of Bengali numerals. CMATERdb 3.1.2 has 50 classes of Bengali basic characters. CMATERdb 3.1.3.1, and 3.1.3.3 have 171 classes of Bengali compound characters. However, among these 171 characters, some characters have multiple writing fashions. A total of 199 distinct classes are present in these datasets in terms of writing fashions. Lastly, CMATERdb 3.1.4 has 7 classes of Bengali modifiers. Moreover, Ekush [46], BanglaLekha [47] and NumtaDB [48] datasets were used to test the classification accuracy of the architecture as well.

4.2. Augmentation

In deep learning, lack of adequate data has been a remarkable dilemma from the very beginning. The reason behind this is that if a model learns from fewer samples for a specific class, that class will have less probability to be predicted correctly, as the model does not know the pattern well compared to other patterns with enough samples. As a result, misclassifications are bound to happen. With more data, deep models can generalize the learning procedure. Augmentation allows the architectures to explore additional learning which can further enhance the classification outcome. For example, suppose a cat picture is provided to the architecture but the picture is rotated 45 degrees to the left or 45 degrees to the right. Despite being rotated to left or right, the picture is still a cat picture. Moreover, if the picture is translated, zoomed or if noise is added with the picture, still the picture is a cat picture. Augmentation procedure computationally produces these sorts of images by processes such as rotating, translating, shearing, flipping, etc. These processes produce artificial samples that not only increase the training size but also increase the learning outcome of the classifier. In addition, data augmentation can be used to make equal samples for training classes to solve the imbalance issue. In this research, we have applied augmentation on the train data and then split the train data into training sets and validation sets. No augmentation was applied on the test set. Moreover, the test set was provided separately with the considered datasets.

Augmentation was implemented with the compensation of the Augmentor Library [49]. When using the method of augmentation, the max left rotation, the max right rotation, and the probability of rotation of the rotation function was anchored to 3, 3, and 0.4 respectively. The values of grid width, grid height, probability, and magnitude of the random distortion function were adjusted to 4, 4, 0.4, and 4 respectively. Moreover, the percentage ranges and the probability of the zoom random function were adjusted to 0.9 and 0.2 respectively.

4.3. Data Preparation

We took the CMATERdb datasets under consideration, augmented the training datasets to generate 500 samples per class, and after some preprocessing steps, prepared 8 different formations of datasets as works on one or more datasets have been conducted in the previous research [34–38,40,41]. We considered the CMATERdb datasets and augmented the training datasets to generate 500 samples per class. After the augmentation procedure, the dilemma of imbalanced training data was resolved. After that, some preprocessing steps were employed and 8 different formations of datasets were prepared as works on one or more such formations have been conducted in the previous research [34–38,40,41]. The details of preprocessing steps can be discovered in the "Experimental Analysis" section. Table 1 illustrates the dataset formation, number of classes, number of samples in train, validation and test sets, and whether the datasets are balanced or imbalanced. Figures 1–3 illustrate samples of Bengali numerals, Bengali basic characters, and Bengali modifiers. The list of 171 compound characters can be located in the primary research work [41].

Table 1. Dataset formations, number of classes, number of training, validation, and testing samples, and whether dataset is balanced or imbalanced.

Dataset Name	Dataset Formation	No. of Classes	Train Size	Train Size After Augmentation	Train Size after Splitting	Validation Size after Splitting	Testing Samples	Balanced?
Dataset-1	3.1.1	10	3480	5000	4000	1000	2520	Yes
Dataset-2	3.1.2	50	12,000	25,000	20,000	5000	3000	Yes
Dataset-3	3.1.3.1	171	33,404	85,500	68,400	17,100	8383	No
Dataset-4	3.1.3.3	171	33,404	85,500	68,400	17,100	8520	No
Dataset-5	3.1.3.3	199	33,404	99,500	79,600	19,900	8520	No
Dataset-6	3.1.4	7	1644	3500	2800	700	414	No
Dataset-7	3.1.2 + 3.1.3.1 + 3.1.4	256	47,048	128,000	102,400	25,600	11,804	No
Dataset-8	3.1.2 + 3.1.3.3 + 3.1.4	256	48,083	128,000	102,400	25,600	11,935	No

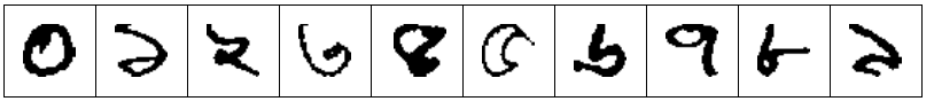

Figure 1. Bengali Numerals Samples from CMATERdb Dataset.

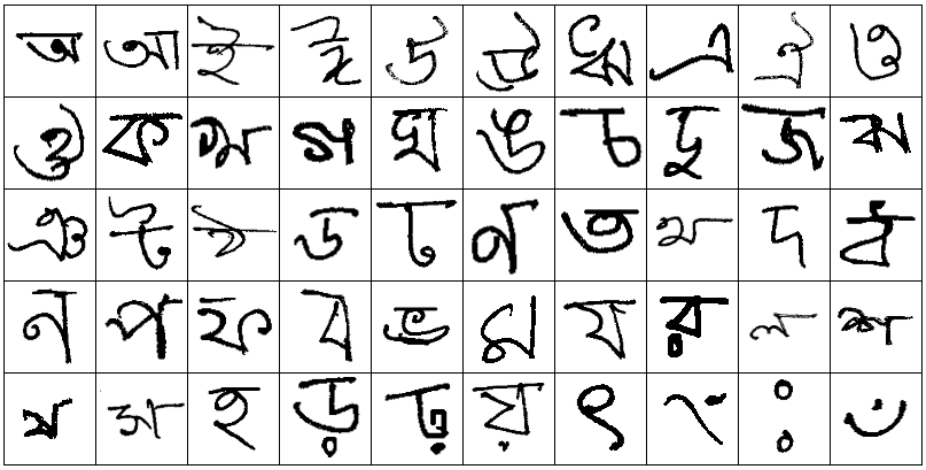

Figure 2. Bengali Basic Characters Samples from CMATERdb Dataset.

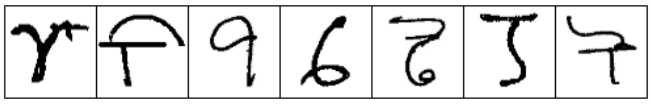

Figure 3. Bengali Modifiers Samples from CMATERdb Dataset.

4.4. Deep Convolutional Neural Network

Deep learning [50,51] is being used in the domain of computer vision for a long time now. The convolutional neural network, also recognized as CNN, is one of the most common forms of deep neural networks which is generally exercised for interpreting optical images of various research domains [52]. Convolutional neural networks are employed for object or person identification in images and videos, image classification, recommender systems, natural language processing, medical image processing, and financial investigation [53–55]. CNNs were motivated by the connectivity among neurons which can be observed in the animal cortex and are usually functioned for image recognition or identification in various domains of research.

4.5. Proposed BengaliNet Architecture

The purpose of this research was to design a low-cost deep convolutional architecture that can accurately predict the Bengali handwritten characters. That is why BengaliNet was chosen as the name of the architecture. The size of the input image was set to 28×28 for BengaliNet architecture. The input image was then passed to three different paths. All these three paths have essentially the same combination of layers, but the filter size of convolution is different. 3×3, 5×5, and 7×7 filters were used for the three paths. It turned out because of having both simple and complex shaped characters, any one of the 3×3, 5×5, and 7×7 filters could not produce better learning but in combination, the classification can be enhanced dramatically. This phenomenon is also referred to as multicolumn CNN design as the end network consists of multiple convolutional neural networks or columns [39]. More about the intuition behind classification enhancement can be discovered in the "Experimental Analysis" section with proper experimental results.

Each of the paths started with two convolutional layers of depth 32 followed by a max-pooling layer and a dropout of 25%. After that, the output was provided to two branches. The first branch had one convolutional layer followed by a batch normalization layer and another convolutional layer followed by another batch normalization layer. Then, max-pooling was added followed by a dropout of 20%. The second branch started with two convolutional layers followed by a max-pooling layer and a dropout of 25%. For both branches, the depth of the convolutional layers was set to 64. Outputs of the two branches were then concatenated and fed to a convolutional layer of depth 64 followed by a max-pooling layer and a dropout of 25%. Next, a fully connected layer of size 256 was added after flattening followed by a dropout of 50%. Finally, an output layer was connected with the SoftMax activation function with a size of 7, 10, 50, 171, 199, or 256 based on the number of classes. For all the convolutional layers, the ReLU activation function was used and for all max-pooling layers, the max-pooling filter size was 2×2.

Each of the three paths generated a prediction but despite applying a voting process to predict the outcome, we applied the SoftMax averaging technique here. The average of three SoftMax outputs was considered to be the final SoftMax output and was used to predict the outcome. More intuitions about this idea can be discovered in the "Experimental Analysis" section. Figure 4 illustrates the architecture of BengaliNet. Our architecture has 2.24–2.43 million parameters based on the number of classes in consideration. Table 2 illustrates the number of parameters for each layer considering 256 classes as 256 is the maximum number of classes in our experimental datasets.

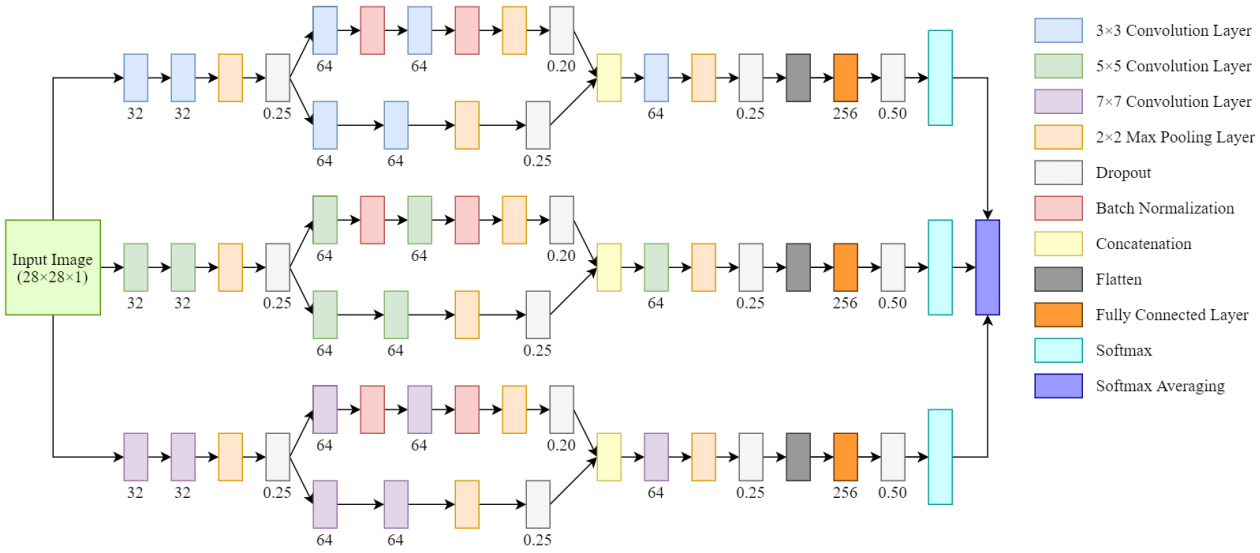

Figure 4. Proposed BengaliNet Architecture: A Low-Cost Novel Deep Convolutional Neural Network. Numbers below the layers indicate number of filters for convolutional layers, dropout values and number of neurons for fully connected layers.

Table 2. Layer description, output shape, and number of parameters for each layer of BengaliNet considering 256 classes.

Layer Number	Layer Name	Output Shape	Layer Connected to	3 × 3 Path Parameters	5 × 5 Path Parameters	7 × 7 Path Parameters
1	Input	28 × 28 × 1	-	0	0	0
2	Conv2D	28 × 28 × 32	1	320	832	1600
3	Conv2D	28 × 28 × 32	2	9248	25,632	50,208
4	Max-Pooling2D	14 × 14 × 32	3	0	0	0
5	Dropout	14 × 14 × 32	4	0	0	0
6	Conv2D	14 × 14 × 64	5	18,496	51,264	100,416
7	Batch Normalization	14 × 14 × 64	6	256	256	256
8	Conv2D	14 × 14 × 64	7	36,938	102,464	200,768
9	Batch Normalization	14 × 14 × 64	8	256	256	256
10	Max-Pooling2D	7 × 7 × 64	9	0	0	0
11	Dropout	7 × 7 × 64	10	0	0	0
12	Conv2D	14 × 14 × 64	5	18,496	51,264	100,416
13	Conv2D	14 × 14 × 64	12	36,938	102,464	200,768
14	Max-Pooling2D	7 × 7 × 64	13	0	0	0
15	Dropout	7 × 7 × 64	14	0	0	0
16	Concatenate	7 × 7 × 128	11, 15	0	0	0
17	Conv2D	7 × 7 × 64	16	73,792	204,864	401,472
18	Max-Pooling2D	3 × 3 × 64	17	0	0	0
19	Dropout	3 × 3 × 64	18	0	0	0
20	Flatten	576	19	0	0	0
21	Fully Connected Layer	256	20	147,712	147,712	147,712
22	Dropout	256	21	0	0	0
23	SoftMax Output	256	22	65,792	65,792	65,792
Total Parameters: 408,224 (3 × 3) + 752,800 (5 × 5) + 1,269,664 (7 × 7) = 2,430,688						

5. Experimental Analysis

In this section, preprocessing, experimental settings, results analysis, individual vs. BengaliNet architecture, reason of better performance, comparison with previous works, and comments of misclassification have been described and illustrated.

5.1. Preprocessing

Because of providing images to a convolutional neural network, a heavy preprocessing of the pictures was skipped as CNN is a compelling network that can identify significant features from raw images. However, some preprocessing steps were required. First, all the images were auto-cropped as most of the images had a large area with only background pixels. Figure 5 illustrates the effect of auto-cropping. After that, all the images were converted to binary images as there were some grayscale images in some of the classes. Figure 6 illustrates the effect of binarization. As the proposed architectures take inputs of size 28 × 28 × 1, all the input pictures were reshaped to 28 × 28 × 1.

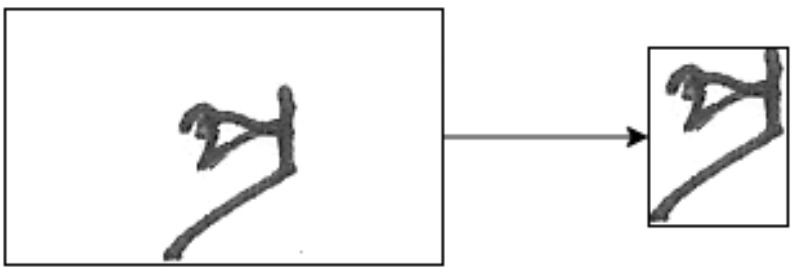

Figure 5. Example of auto-cropping on a sample image.

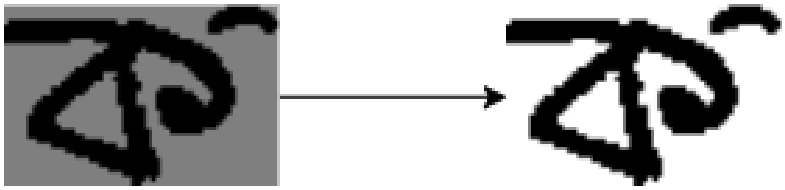

Figure 6. Example of binarization on a sample image.

5.2. Experimental Settings

The model was trained for 200 epochs with a batch size of 512 as after that the validation loss grew almost constant for the rest of the epochs. 'Adam' optimizer [56] was employed to maximize the error function. A categorical cross-entropy function was employed for the loss or error function. For avoiding overfitting, the dropout technique was used. SoftMax activation [57] was used while applying the output layers. Learning rate of 0.001, 0.0001, 0.00001, 0.000005, and 0.000001 were used for 100, 40, 30, 20, and 10 epochs respectively in a sequential order while learning.

5.3. Result Analysis

After applying the preprocessing approaches, the training sets and validation sets were prepared. After that, the proposed BengaliNet architecture was applied to all the datasets. Table 3 presents the number of classes and number of parameters for all 8 dataset formations. Tables 4 and 5 represent the overview of overall accuracies for BengaliNet architecture applied on the above-mentioned datasets and other independent datasets i.e., Ekush, BanglaLekha, and NumtaDB. When testing with Ekush, 15% data were taken under consideration and while testing with BanglaLekha and NumtaDB, the whole dataset was considered. For each dataset, we also calculated the class-wise accuracy, precision, recall, and f1-score which can be located in the supplementary file. Kaggle GPU environment i.e., 4 CPU cores, 16 Gigabytes of RAM, 20 Gigabytes of disk space, was used while calculating the test results. In that environment, the average recognition time per sample for 8 datasets using BengaliNet was 3.6, 4.1, 10.5, 10.5, 11.0, 1.7, 27.3, and 28.6 ms per sample respectively. The training times required for this research for the 8 considered datasets are 1.24, 6.07, 8.36, 8.42, 9.56, 0.95, 25.23, and 26.01 h respectively. Here, it should be mentioned that training the models was a one-time process, and the weights were saved after the training. Therefore, no retraining was needed while testing for independent and unseen datasets. In later sections, it can be observed that for all 8 datasets, BengaliNet achieved the highest accuracy.

Table 3. The number of classes and number of parameters for individual and BengaliNet architectures for all the datasets.

Dataset Name	Number of Classes	3 × 3 Params.	5 × 5 Params.	7 × 7 Params.	Total Params.
Dataset-1	10	345 K	690 K	1.21 M	2.24 M
Dataset-2	50	355 K	700 K	1.22 M	2.27 M
Dataset-3	171	386 K	731 K	1.25 M	2.37 M
Dataset-4	171	386 K	731 K	1.25 M	2.37 M
Dataset-5	199	394 K	738 K	1.26 M	2.39 M
Dataset-6	7	344 K	689 K	1.21 M	2.24 M
Dataset-7	256	408 K	753 K	1.27 M	2.43 M
Dataset-8	256	408 K	753 K	1.27 M	2.43 M

Table 4. Overall Performance of CMATERdb datasets in terms of overall accuracy.

Name of Datasets	Number of Classes	BengaliNet Accuracy
Dataset-1	10	99.01%
Dataset-2	50	98.97%
Dataset-3	171	98.35%
Dataset-4	171	96.07%
Dataset-5	199	95.89%
Dataset-6	7	97.79%
Dataset-7	256	97.82%
Dataset-8	256	95.71%

Table 5. Overall Performance of Ekush, BanglaLekha and NumtaDB datasets in terms of overall accuracy.

Name of Datasets	Number of Classes	Overall Accuracy
Ekush Basic	50	98.36%
Ekush Numerals	10	99.13%
BanglaLekha Basic	50	96.09%
BanglaLekha Numerals	10	98.11%
NumtaDB Numerals	10	97.50%

5.4. Individual vs. BengaliNet Architecture

To further prove the fact that our proposed architecture can enhance the recognition of Bengali handwritten characters, we measured the performances using 3×3, 5×5, and 7×7 paths individually as well. Table 6 illustrates the comparison of performances among the BengaliNet architecture and single path architectures for the considered datasets. From the table, it is clear that BengaliNet works better than any of the three architectures alone.

Table 6. Comparison of performance among individual and BengaliNet architectures for the considered datasets.

Dataset Name	3×3 Accuracy	5×5 Accuracy	7×7 Accuracy	Overall Accuracy
Dataset-1	98.57%	98.81%	98.85%	**99.01%**
Dataset-2	98.30%	98.43%	98.37%	**98.97%**
Dataset-3	97.66%	97.67%	97.38%	**98.35%**
Dataset-4	94.87%	94.67%	95.01%	**96.07%**
Dataset-5	94.85%	94.87%	94.62%	**95.89%**
Dataset-6	97.79%	97.05%	97.54%	**97.79%**
Dataset-7	96.84%	93.03%	95.90%	**97.82%**
Dataset-8	94.42%	94.83%	94.80%	**95.71%**
Ekush Basic	95.18%	96.06%	94.99%	**98.36%**
Ekush Numerals	97.44%	98.13%	97.74%	**99.13%**
BanglaLekha Basic	91.86%	90.64%	88.74%	**96.09%**
BanglaLekha Numerals	96.27%	96.41%	96.68%	**98.11%**
NumtaDB Numerals	96.93%	96.92%	96.64%	**97.50%**

5.5. Tuning of Hyperparameters

The hyperparameters that have been used in this study are the number of epochs, learning rate, and batch size. To justify the tuning of these hyperparameters, classification results for various values of hyperparameters were calculated. In this section, results for dataset-2 have been presented. Figure 7 illustrates the accuracy vs. number of epochs graph when the learning rate and batch size are fixed as mentioned in experimental settings.

Figure 8 illustrates the accuracy vs. batch size graph when the learning rate and the number of epochs are set to the values mentioned in experimental settings. Finally, Figure 9 shows the accuracy for different learning rates. It can be noticed that better accuracy can be achieved for the proposed setup.

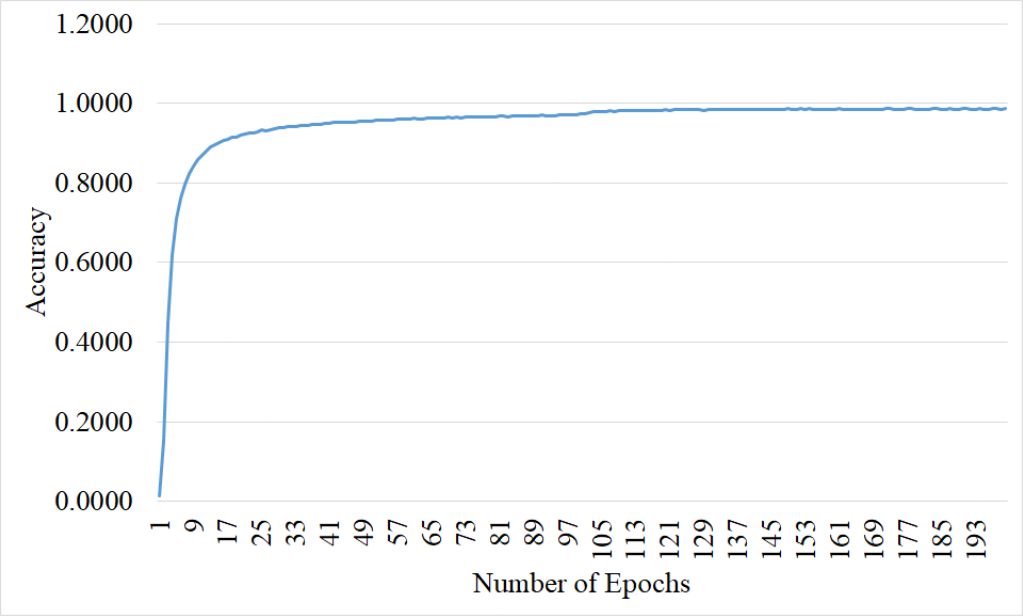

Figure 7. Accuracy vs. number of epochs graph for dataset-2.

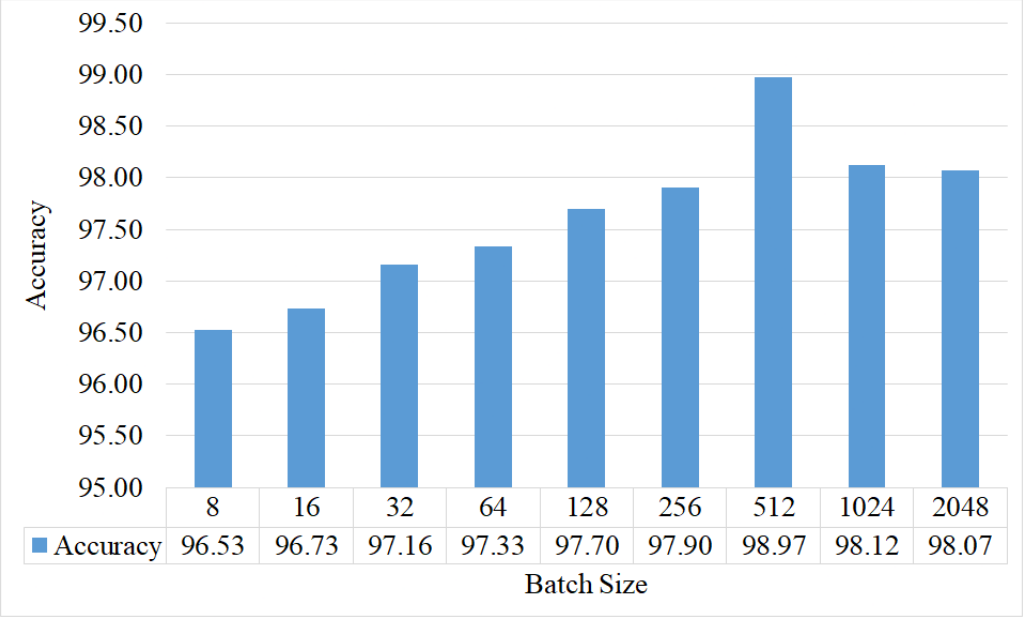

Figure 8. Accuracy vs. batch size graph for dataset-2.

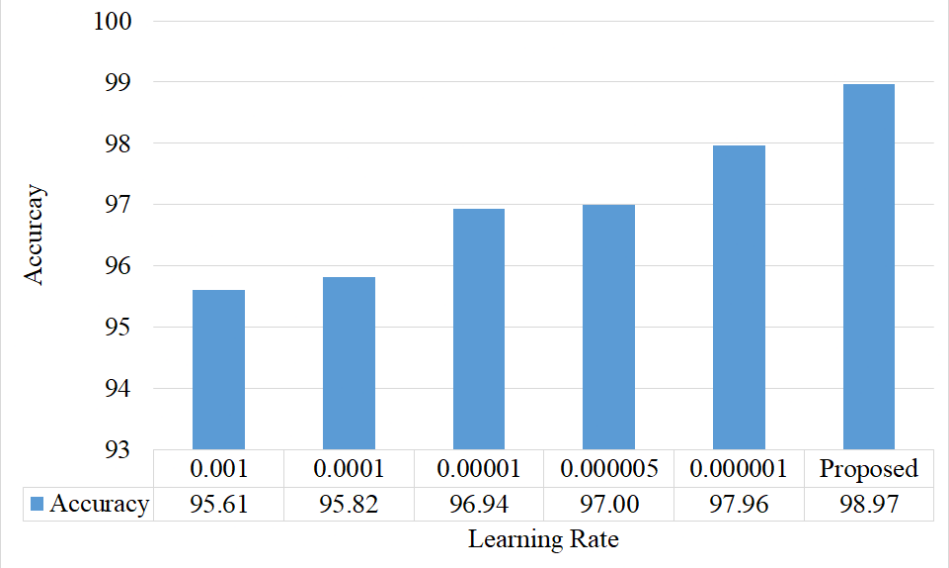

Figure 9. Accuracy vs. learning rate graph for dataset-2.

5.6. Comparison with Previous Works

After measuring the performance of our proposed models, we compared the results with the previous works. Table 7 illustrates the comparison between our proposed approach and previous works. Figure 10 illustrates the comparative results in terms of accuracy for all 8 datasets.

Table 7. Comparison among results obtained by our proposed BengaliNet and previous approaches.

Dataset Name	Number of Classes	Method/ Approach	Classification Accuracy	Remarks
Dataset-1	10	DCNN on PLSS and HOPS [38]	98.50%	-
		Support Vector Machine [43]	98.61%	-
		CNN [40]	98.80%	-
		BengaliNet	**99.01%**	Proposed
Dataset-2	50	Deep CNN [34]	97.80%	Custom Test Set, No tuning of learning rate
		Support Vector Machine [44]	86.10%	-
		Support Vector Machine [42]	87.28%	-
		MobileNet V1 [35]	98.37%	Did not consider entire test set
		CNN [40]	98.56%	-
		BengaliNet	**98.97%**	Proposed
Dataset-3	171	CNN [40]	98.12%	Did not consider the entire train and test sets
		BengaliNet	**98.35%**	Proposed
Dataset-4	171	Hybrid-HOG-based CNN [36]	92.57%	-
		DCNN with ELU, dropout [37]	93.68%	Did not consider the entire train and test sets
		CNN [40]	95.70%	Did not consider the entire train and test sets
		BengaliNet	**96.07%**	Proposed
Dataset-5	199	Hybrid-HOG-based CNN [36]	92.57%	-
		BengaliNet	**95.89%**	Proposed
Dataset-6	7	CNN [40]	97.29%	Applied augmentation
		BengaliNet	**97.79%**	Augmentaion+Proposed
Dataset-7	256	CNN [40]	97.42%	Did not consider the entire train and test sets
		BengaliNet	**97.82%**	Proposed
Dataset-8	256	Two-passed Approach [41]	87.26%	Publication by authors of the datasets
		BengaliNet	**95.71%**	Proposed

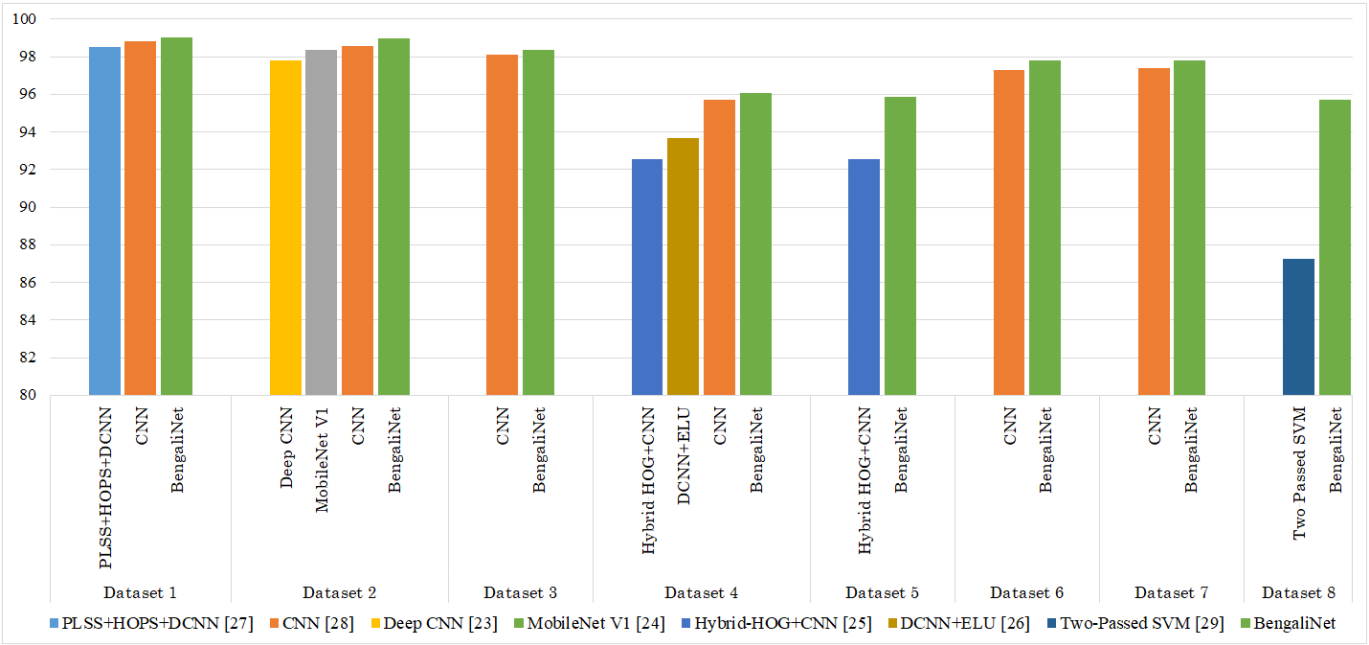

Figure 10. Illustration of performance comparison for all 8 datasets among previous noteworthy approaches and our proposed architecture in terms of overall accuracy.

From Table 7, it can be observed that for each formation our proposed approach has outperformed the previous noteworthy works. In the remarks section, we also have provided comments (if required) so that the comparison can be fair. For example, for dataset 7, one research [40] achieved 97.42% accuracy, our proposed architecture has achieved 97.82% accuracy. It does not seem much improvement although there were 11,804 samples and this slight improvement denotes a lot of corrected misclassifications. However, the research [40] that achieved 97.42% did not consider the entire train and test sets which makes our outcomes better as ignoring some samples while training and testing may affect the performance positively. Moreover, we provided the results for some popular CNN architectures i.e., GhostNet, InceptionV4, Xception, MobileNetV1, MobileNetV2, MobileNetV3Small, and MobileNetV3Large architectures in Table 8. Despite being a smaller architecture of 6.5 million parameters, GhostNet did not achieve the desired performance for the considered dataset. InceptionV4 with 43 million parameters and Xception with 23 million parameters achieved a high performance but could not outperform the performance achieved by the proposed BengaliNet architecture. On the other hand, the MobileNet architectures achieved decent performance but could not outperform BengaliNet as well.

Table 8. Comparison among results obtained by our proposed BengaliNet and popular CNN architectures.

Method/Approach	Dataset-1	Dataset-2	Dataset-3	Dataset-4	Dataset-5	Dataset-6	Dataset-7	Dataset-8
GhostNet	87.13%	83.33%	79.48%	77.22%	80.12%	75.80%	80.42%	76.04%
InceptionV4	97.05%	96.07%	97.04%	95.51%	94.99%	95.89%	97.57%	95.41%
Xception	97.45%	97.13%	97.18%	94.96%	95.34%	96.86%	97.32%	95.05%
MobileNetV1	96.04%	94.07%	95.11%	93.02%	93.25%	92.99%	95.07%	93.52%
MobileNetV2	96.52%	94.20%	95.09%	93.00%	93.39%	93.72%	95.27%	94.48%
MobileNetV3Small	96.60%	94.70%	95.38%	93.39%	94.08%	93.96%	95.38%	94.87%
MobileNetV1Large	96.68%	94.63%	96.77%	94.37%	94.47%	94.69%	96.15%	94.86%
BengaliNet	**99.01%**	**98.97%**	**98.35%**	**96.07%**	**95.89%**	**97.79%**	**97.82%**	**95.71%**

Additionally, Table 9 illustrates a comparison with some previous studies that focused on the recognition of Bengali handwritten characters by using transfer learning. It can be observed that BengaliNet has a fewer number of parameters than previous transfer learning

methods which denotes that BengaliNet needs less recognition time. Moreover, in terms of overall accuracy, BengaliNet outperforms all these studies as well. It can be noticed that BengaliNet achieved decent accuracy for Ekush, BanglaLekha, and NumtaDB datasets also as the last achieved highest accuracy for these datasets are 97.73% [46], 93.45% [58], and 98.40% [59] respectively.

Table 9. Comparison with previous studies in terms of total parameters and overall accuracy.

Methods and Datasets	Criteria	Previous Works	BengaliNet
VGG-16 on 3.1.1 [44]	Parameters Accuracy	138 M 93.33%	**2.24 M** **99.01%**
AlexNet on 3.1.1 [44]	Parameters Accuracy	61 M 95.00%	**2.24 M** **99.01%**
ResNet-50 on BanglaLekha [45]	Parameters Accuracy	23 M 96.12%	**2.24 M** **98.11%**

5.7. Why Proposed Architecture Performs Better?

BengaliNet works better mainly because of its three-way architecture. When designing the BengaliNet architecture, different combinations of filter sizes were taken into consideration. It was observed that many Bengali characters have complex shapes. To extract these complex shapes, multiple convolutions were needed but to design a deep network, the input size needs to be larger too. These type of architectures already exists i.e., AlexNet, VGG-16, ResNet-50, etc. But the main problem of these architectures is that these architectures are heavy and have 23–138 million parameters. Bengali handwritten character recognition is supposed to be conducted in mobile devices, hence, a low-cost architecture was needed. 3×3, 5×5, or 7×7 individual architectures were able to produce a decent accuracy as even in these individual architectures, there were branches. Convolutions with batch normalization and normal convolutions ensured better learning than sequential models. When designing BengaliNet architecture, we took filters of multiple sizes under consideration. The filter size of the convolution layers in three pathways were 3×3, 5×5, and 7×7 respectively. Moreover, each of these paths has its own branches with multiple convolution layers, batch normalization, and dropouts. Merging features obtained from the general convolution and the convolution with batch normalization ensured better learning than a simple sequential model. It was observed that 5×5 and 7×7 architectures were being able to learn some complex features which were not being learned by 3×3 architecture. Hence, these three architectures were combined.

Instead of choosing the voting method, the SoftMax averaging technique was chosen for this research. As there were only three architectures in consideration, the voting method was depending on random class selection when the three architectures were producing different test outcomes. But for the SoftMax averaging technique, this randomness can be reduced. Suppose, for a specific test sample, the three paths were predicting A, B, and C classes. In such cases, the voting method chooses a random class among three different predicted classes which can either produce a better accuracy or a worse accuracy. On the other hand, the SoftMax averaging technique takes the average of three SoftMax arrays produced by three paths and therefore, there is a high possibility to omit the possible randomness of the voting method. In Table 10, the overall accuracy for the voting method and the SoftMax averaging method have been presented. It can be noticed that the SoftMax averaging technique is producing better performance. When there are many architectures under consideration, the efficiency of the voting method will increase. But as in this research, the focus was specifically on cost-effectiveness so that recognition can be performed in mobile devices, only three small architectures were considered. Therefore, with evidence of Table 10, we chose the SoftMax averaging technique.

Table 10. Comparison of performance among SoftMax averaging technique and voting method for the considered datasets.

Dataset Name	Voting Method	SoftMax Averaging Technique
Dataset-1	98.21%	**99.01%**
Dataset-2	98.03%	**98.97%**
Dataset-3	97.84%	**98.35%**
Dataset-4	95.65%	**96.07%**
Dataset-5	94.93%	**95.89%**
Dataset-6	95.17%	**97.79%**
Dataset-7	96.59%	**97.82%**
Dataset-8	94.96%	**95.71%**
Ekush Basic	97.91%	**98.36%**
Ekush Numerals	98.57%	**99.13%**
BanglaLekha Basic	94.55%	**96.09%**
BanglaLekha Numerals	97.47%	**98.11%**
NumtaDB Numerals	96.64%	**97.50%**

Because of combining three lightweight architectures, excessive learning by extracting deep features using big or transfer learned architectures could be ignored. Furthermore, our architecture outperformed MobileNet architecture which used 4.2 million parameters and other studies that worked with low-cost mechanisms [40,42,43] in terms of accuracy and number of parameters. Additionally, the dropout technique was used for bypassing overfitting so that the output cannot rely on some specific neurons only. However, selecting the values of dropouts was an experimental tuning procedure. Moreover, we chose the 'Adam' optimizer rather than the momentum function or RMSprop algorithm as the 'Adam' optimizer bears both properties of momentum and RMSprop. Furthermore, the learning rate was not kept consistent. The first epoch started with a 0.001 learning rate and ended with a 0.000001 learning rate after the completion of 200 epochs. Because of all these reasons, the proposed BengaliNet performed better despite being a low-cost architecture.

From this discussion, some major differences between the proposed model and the baseline models can be noticed. Unlike the baseline models, the proposed model does not repeat the same convolution block multiple times. Rather, the proposed model introduces a three-way architecture with filters of three sizes with different convolution blocks and branches in each way. Instead of using a voting method, the proposed model uses the SoftMax averaging technique. The input of the proposed model is a 28×28 grayscale image which is much smaller in size than the baseline approaches. Moreover, dropouts and batch normalization have been used here for better learning.

5.8. Comments on Misclassifications

To find out why misclassifications were occurring, the focus was needed on the worst-performing classes for each of the datasets. This was the reason for which we decided to calculate class-wise performance in the first place. Table 11 represents the two worst-performing classes for each of the datasets while applying the proposed architectures. If noticed closely, the two worst-performing class accuracy is high for datasets 1, 2, and 6 as these datasets have 10, 50, and 7 classes. Two worst-performing class accuracy is low for other datasets where 171, 199, and 256 classes are under consideration. As the number of test samples is higher in these datasets, a few classes with low accuracy does not hurt the overall accuracy. Additionally, some classes have a very small number of test samples i.e., 10–20 test samples. For these classes, a small number of misclassifications will cause low class-wise accuracy. For example, if 2 out of 10 samples are misclassified, the class-wise accuracy becomes 80%. On the other hand, another class with 2 misclassifications out of 100 samples will produce 98% class-wise accuracy.

Table 11. Two Worst-Performing Classes for 8 Datasets.

Dataset Name	Number of Classes	Class Number	Class Accuracy
Dataset-1	10	6 9	96% 96%
Dataset-2	50	30 44	91% 95%
Dataset-3	171	19 139	70% 86%
Dataset-4	171	141 59	69% 72%
Dataset-5	199	180 141	65% 69%
Dataset-6	7	3 6	91% 96%
Dataset-7	256	40 182	70% 74%
Dataset-8	256	123 237	60% 63%

From Table 11, it can be observed that some of the classes are frequently victims of misclassification. We have two observations or comments on these misclassifications. First, some classes are similar in terms of curves and looks. That is why misclassifications happened frequently as the models become confused while recognizing these extremely similar classes. Figure 11 presents some classes of this type.

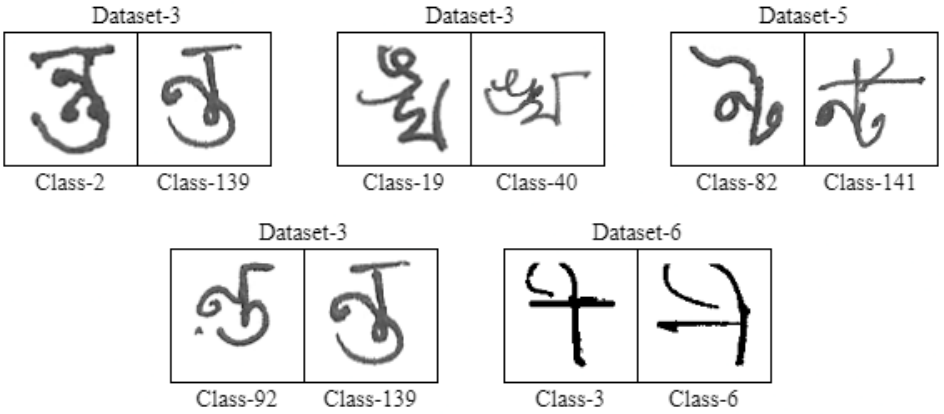

Figure 11. Examples of extremely similar classes for different datasets.

Secondly, for compound characters, some characters had more than one writing fashion. For some characters, the writing fashions were similar and for some, the writing fashions were different. Hence, the model struggled whether these writing fashions were merged into one class or considered to be separate classes. Figure 12 represents some classes of this type. Moreover, there are some samples in the test set which are so noisy that even human eyes cannot predict the classes correctly.

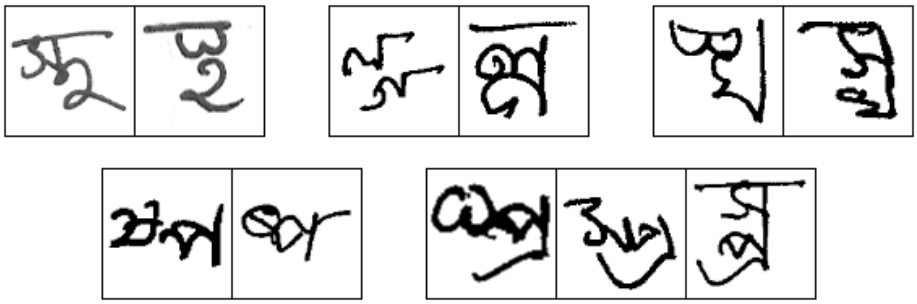

Figure 12. Examples for characters with multiple writing fashions.

6. Conclusions and Future Scopes

Despite being the seventh most-spoken language in the world, Bengali has not received many contributions in this domain due to a huge number of characters with complex shapes. Previously, very few studies on Bengali handwritten character recognition addressed the importance of low-cost architecture. However, they traded off the performance with fewer parameters. In this study, the focus was on designing a low-cost novel convolutional neural architecture for the Bengali handwritten character recognition so that the recognition can be done on mobile devices. The proposed model called BengaliNet uses only 2.24 to 2.43 million parameters based on the number of output classes and takes an input of size 28×28 only. When designing the architecture, branches, and filters of multiple sizes i.e., 3×3, 5×5, and 7×7 were used which extracted better deep features. Finally, the SoftMax averaging technique was used to make the final decision for the character recognition. The experimental results showed that the proposed architecture can achieve better accuracy than the baseline approaches and the methods used in previous research in terms of performance and number of parameters. The results also showed that the proposed model obtains similar performance for previously unknown datasets as well. We believe the contributions of our study will be beneficial for future research such as automatic digitization of historical documents, automated license plate recognition, automated postal service, and so on. In the future, we will focus on improving the architecture more and apply the architecture to more datasets. Furthermore, the lack of an ideal Bengali handwritten character dataset is also a dilemma in this domain of research. In the future, we will work in this regard also.

Supplementary Materials: The following are available online at https://www.mdpi.com/2076-3417/11/15/6845/s1.

Author Contributions: Conceptualization, A.S.; methodology, A.S. and M.A.M.H.; software, A.Y.S. and M.M.H.; validation, M.A.M.H. and J.S.; formal analysis, A.Y.S. and M.M.H.; investigation, M.A.M.H. and J.S.; resources, A.S.; data curation and collection, A.Y.S. and M.M.H.; writing—original draft preparation, A.S.; writing—review and editing, A.Y.S. and M.M.H.; visualization, A.Y.S. and M.M.H.; supervision, M.A.M.H.; project administration, J.S.; funding acquisition, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the datasets used in this research are publicly accessible. CMATERdb Datasets are available at: https://code.google.com/archive/p/cmaterdb, accessed on 16 May 2021. Ekush dataset can be accessed at https://shahariarrabby.github.io/ekush, accessed on 16 May 2021, BanglaLekha dataset can be found at https://data.mendeley.com/datasets/hf6sf8zrkc/2, accessed on 16 May 2021, and NumtaDB dataset is available at https://www.kaggle.com/BengaliAI/numta, accessed on 16 May 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mori, S.; Nishida, H.; Yamada, H. *Optical Character Recognition*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1999.
2. Berhard, D.; Simons, G.; Fennig, C. *Ethnologue: Languages of the World*; SIL International: Dallas, TX, USA, 2020. Available online: https://www.ethnologue.com (accessed on 16 May 2021).
3. Ethnologue. Summary by Language Size. 2018. Available online: https://www.ethnologue.com/statistics/summary-language-size-18 (accessed on 16 May 2021).
4. Agency, C.I.; Staff, C.I.A.C. *The World Factbook 2007*; Government Printing Office: Washington, DC, USA, 2007.
5. Eberhard, D.M.; Simons, G.F.; Fennig, C.D. *Ethnologue: Languages of the World*; SIL International: Dallas, TX, USA, 2019. Available online: https://www.ethnologue.com (accessed on 16 May 2021)
6. Rumnaz Imam, S. English as a global language and the question of nation-building education in Bangladesh. *Comput. Educ.* **2005**, *41*, 471–486. [CrossRef]
7. Al Farabi, R. Identity Crisis of the Linguistic Minorities in The Process of Acculturation with Reference to Secondary Curriculum in the State of West Bengal, India. In Proceedings of The 2nd International Conference on New Approaches in Education, Oxford, UK, 27–29 March 2020; pp. 153–161.
8. Pandey, A. Using mother tongues as building blocks in childhood education. *Child. Educ.* **2014**, *90*, 61–67. [CrossRef]
9. Das, N.; Acharya, K.; Sarkar, R.; Basu, S.; Kundu, M.; Nasipuri, M. A benchmark image database of isolated Bangla handwritten compound characters. *Int. J. Doc. Anal. Recognit. IJDAR* **2014**, *17*, 413–431. [CrossRef]
10. Zanwar, S.R.; Shinde, U.B.; Narote, A.S.; Narote, S.P. Handwritten English Character Recognition Using Swarm Intelligence and Neural Network. In *Intelligent Systems, Technologies and Applications*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 93–102.
11. Narayanan, V.S.; Kasthuri, N. An efficient recognition system for preserving ancient historical documents of English characters. *J. Ambient. Intell. Humaniz. Comput.* **2020**, 1–9. [CrossRef]
12. Gan, J.; Wang, W.; Lu, K. Compressing the CNN architecture for in-air handwritten Chinese character recognition. *Pattern Recognit. Lett.* **2020**, *129*, 190–197. [CrossRef]
13. Cao, Z.; Lu, J.; Cui, S.; Zhang, C. Zero-Shot Handwritten Chinese Character Recognition with Hierarchical Decomposition Embedding. *Pattern Recognit.* **2020**, *107*, 107488. [CrossRef]
14. Granell, E.; Chammas, E.; Likforman-Sulem, L.; Martínez-Hinarejos, C.D.; Mokbel, C.; Cîrstea, B.I. Transcription of spanish historical handwritten documents with deep neural networks. *J. Imaging* **2018**, *4*, 15. [CrossRef]
15. Boufenar, C.; Batouche, M.; Schoenauer, M. An artificial immune system for offline isolated handwritten arabic character recognition. *Evol. Syst.* **2018**, *9*, 25–41. [CrossRef]
16. Mukhoti, J.; Dutta, S.; Sarkar, R. Handwritten Digit Classification in Bangla and Hindi Using Deep Learning. *Appl. Artif. Intell.* **2020**, *34*, 1–26. [CrossRef]
17. Deore, S.P.; Pravin, A. Devanagari Handwritten Character Recognition using fine-tuned Deep Convolutional Neural Network on trivial dataset. *Sādhanā* **2020**, *45*, 1–13. [CrossRef]
18. Balaha, H.M.; Ali, H.A.; Saraya, M.; Badawy, M. A new Arabic handwritten character recognition deep learning system (AHCR-DLS). *Neural Comput. Appl.* **2021**, *33*, 6325–6367. [CrossRef]
19. Nanehkaran, Y.A.; Zhang, D.; Salimi, S.; Chen, J.; Tian, Y.; Al-Nabhan, N. Analysis and comparison of machine learning classifiers and deep neural networks techniques for recognition of Farsi handwritten digits. *J. Supercomput.* **2021**, *77*, 3193–3222. [CrossRef]
20. Papantoniou, K.; Tzitzikas, Y. NLP for the Greek Language: A Brief Survey. In Proceedings of the 11th Hellenic Conference on Artificial Intelligence, New York, NY, USA, 7–8 February 2020; pp. 101–109.
21. Nguyen, H.T.; Nakamura, T.; Nguyen, C.T.; Nakawaga, M. Online trajectory recovery from offline handwritten Japanese kanji characters of multiple strokes. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; pp. 8320–8327.
22. Heo, J.H.; Lee, S.W.; Lee, H.W. A Comparative Study on the Perception Performance of Handwriting in Korean and English Using Machine Learning. In Proceedings of the 2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter), Ho Chi Minh City, Vietnam, 28–30 January 2021; pp. 274–275. [CrossRef]
23. Dokare, I.; Gadge, S.; Kharde, K.; Bhere, S.; Jadhav, R. Recognition of Handwritten Devanagari Character using Convolutional Neural Network. In Proceedings of the 2021 3rd International Conference on Signal Processing and Communication (ICPSC), Paris, France, 14–16 March 2021; pp. 353–359.
24. Abdallah, A.; Hamada, M.; Nurseitov, D. Attention-based Fully Gated CNN-BGRU for Russian Handwritten Text. *J. Imaging* **2020**, *6*, 141. [CrossRef]
25. Mookdarsanit, P.; Mookdarsanit, L. ThaiWrittenNet: Thai Handwritten Script Recognition using Deep Neural Networks. *Azerbaijan J. High Perform. Comput.* **2020**, *3*, 75–93. [CrossRef]
26. KO, M.A.; Poruran, S. OCR-Nets: Variants of Pre-trained CNN for Urdu Handwritten Character Recognition via Transfer Learning. *Procedia Comput. Sci.* **2020**, *171*, 2294–2301. [CrossRef]
27. Bag, S.; Harit, G. A survey on optical character recognition for Bangla and Devanagari scripts. *Sadhana* **2013**, *38*, 133–168. [CrossRef]
28. Salomon, R. Typological observations on the Indic scripts and their relationship with other alphasyllburies [sic]. *Indic Scripts Palaeogr. Linguist. Perspect.* **2007**, 30, 25–43.

29. Bhowmik, T.K.; Bhattacharya, U.; Parui, S.K. Recognition of Bangla handwritten characters using an MLP classifier based on stroke features. In *International Conference on Neural Information Processing*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 814–819.
30. Basu, S.; Das, N.; Sarkar, R.; Kundu, M.; Nasipuri, M.; Basu, D.K. Handwritten Bangla alphabet recognition using an MLP based classifier. *arXiv* **2012**, arXiv:1203.0882.
31. Bhattacharya, U.; Shridhar, M.; Parui, S.K. On recognition of handwritten Bangla characters. In *Computer Vision, Graphics and Image Processing*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 817–828.
32. Das, N.; Basu, S.; Sarkar, R.; Kundu, M.; Nasipuri, M. An improved feature descriptor for recognition of handwritten Bangla alphabet. *arXiv* **2015**, arXiv:1501.05497.
33. Das, N.; Basu, S.; Sarkar, R.; Kundu, M.; Nasipuri, M.; Basu, D. *Handwritten Bangla Compound Character Recognition: Potential Challenges and Probable Solution*; IICAI: Tumkur, India, 2009; pp. 1901–1913.
34. Mondal, S.; Mahfuz, N. Convolutional Neural Networks Based Bengali Handwritten Character Recognition. In *International Conference on Cyber Security and Computer Science*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 718–729.
35. Ghosh, T.; Abedin, M.M.H.Z.; Chowdhury, S.M.; Tasnim, Z.; Karim, T.; Reza, S.S.; Saika, S.; Yousuf, M.A. Bangla handwritten character recognition using MobileNet V1 architecture. *Bull. Electr. Eng. Inform.* **2020**, *9*, 2547–2554. [CrossRef]
36. Sharif, S.; Mohammed, N.; Momen, S.; Mansoor, N. Classification of bangla compound characters using a hog-cnn hybrid model. In *Proceedings of the International Conference on Computing and Communication Systems*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 403–411.
37. Ashiquzzaman, A.; Tushar, A.K.; Dutta, S.; Mohsin, F. An efficient method for improving classification accuracy of handwritten Bangla compound characters using DCNN with dropout and ELU. In Proceedings of the 2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, India, 3–5 November 2017; pp. 147–152.
38. Ghosh, S.; Chatterjee, A.; Singh, P.K.; Bhowmik, S.; Sarkar, R. Language-invariant novel feature descriptors for handwritten numeral recognition. *Vis. Comput.* **2020**, 1–23. [CrossRef]
39. Sarkhel, R.; Das, N.; Das, A.; Kundu, M.; Nasipuri, M. A multi-scale deep quad tree based feature extraction method for the recognition of isolated handwritten characters of popular indic scripts. *Pattern Recognit.* **2017**, *71*, 78–93. [CrossRef]
40. Keserwani, P.; Ali, T.; Roy, P.P. Handwritten Bangla character and numeral recognition using convolutional neural network for low-memory GPU. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 3485–3497. [CrossRef]
41. Das, N.; Sarkar, R.; Basu, S.; Saha, P.K.; Kundu, M.; Nasipuri, M. Handwritten Bangla character recognition using a soft computing paradigm embedded in two pass approach. *Pattern Recognit.* **2015**, *48*, 2054–2071. [CrossRef]
42. Sarkhel, R.; Das, N.; Saha, A.K.; Nasipuri, M. A multi-objective approach towards cost effective isolated handwritten Bangla character and digit recognition. *Pattern Recognit.* **2016**, *58*, 172–189. [CrossRef]
43. Gupta, A.; Sarkhel, R.; Das, N.; Kundu, M. Multiobjective optimization for recognition of isolated handwritten Indic scripts. *Pattern Recognit. Lett.* **2019**, *128*, 318–325. [CrossRef]
44. Pramanik, R.; Dansena, P.; Bag, S. A study on the effect of CNN-based transfer learning on handwritten Indic and mixed numeral recognition. In *Workshop on Document Analysis and Recognition*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 41–51.
45. Chatterjee, S.; Dutta, R.K.; Ganguly, D.; Chatterjee, K.; Roy, S. Bengali Handwritten Character Classification Using Transfer Learning on Deep Convolutional Network. In *International Conference on Intelligent Human Computer Interaction*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 138–148.
46. Rabby, A.S.A.; Haque, S.; Islam, M.S.; Abujar, S.; Hossain, S.A. Ekush: A multipurpose and multitype comprehensive database for online off-line bangla handwritten characters. In *International Conference on Recent Trends in Image Processing and Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 149–158.
47. Biswas, M.; Islam, R.; Shom, G.K.; Shopon, M.; Mohammed, N.; Momen, S.; Abedin, A. Banglalekha-isolated: A multi-purpose comprehensive dataset of handwritten bangla isolated characters. *Data Brief* **2017**, *12*, 103–107. [CrossRef]
48. Alam, S.; Reasat, T.; Doha, R.M.; Humayun, A.I. Numtadb-assembled bengali handwritten digits. *arXiv* **2018**, arXiv:1806.02452.
49. Bloice, M.D.; Stocker, C.; Holzinger, A. Augmentor: an image augmentation library for machine learning. *arXiv* **2017**, arXiv:1708.04680.
50. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
51. Skansi, S. *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2018.
52. Valueva, M.V.; Nagornov, N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Math. Comput. Simul.* **2020**, *177*, 232–243. [CrossRef]
53. Collobert, R.; Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 160–167.
54. Tsantekidis, A.; Passalis, N.; Tefas, A.; Kanniainen, J.; Gabbouj, M.; Iosifidis, A. Forecasting stock prices from the limit order book using convolutional neural networks. In Proceedings of the 2017 IEEE 19th Conference on Business Informatics (CBI), Thessaloniki, Greece, 24–27 July 2017; Volume 1, pp. 7–12.
55. Van den Oord, A.; Dieleman, S.; Schrauwen, B. Deep content-based music recommendation. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 2643–2651.
56. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

57. Goodfellow, I.; Bengio, Y.; Courville, A. Softmax Units for Multinoulli Output Distributions. In *Deep Learning*; MIT Press: Cambridge, MA, USA, 2018.
58. Zayed, M.M.; Utsha, S.N.K.; Waheed, S. Handwritten Bangla Character Recognition Using Deep Convolutional Neural Network: Comprehensive Analysis on Three Complete Datasets. In *Proceedings of International Conference on Trends in Computational and Cognitive Engineering*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 77–87.
59. Paul, D.; Pattnaik, P.K.; Mukherjee, P. A Robust Approach with Text Analytics for Bengali Digit Recognition Using Machine Learning. In *Multimedia Technologies in the Internet of Things Environment*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 157–169.