







Research Article

Recognition of Gurmukhi Handwritten City Names Using Deep Learning and Cloud Computing

**Sandhya Sharma,¹ Sheifali Gupta ,² Deepali Gupta ,² Sapna Juneja ,³
Gaurav Singal ,⁴ Gaurav Dhiman ,⁵ and Sandeep Kautish ,⁶**

¹Chitkara University Institute of Engineering and Technology, Chitkara University, Solan, Himachal Pradesh, India

²Chitkara University Institute of Engineering and Technology, Chitkara University, Patiala, Punjab, India

³KIET Group of Institutions, Delhi, Ghaziabad, India

⁴Netaji Subhash University of Technology, Delhi NCR, India

⁵Govt. Bikram College of Commerce, Patiala, Punjab, India

⁶LBEF Campus, Kathmandu, Nepal

Correspondence should be addressed to Sandeep Kautish; dr.skautish@gmail.com

Received 19 October 2021; Accepted 13 December 2021; Published 4 January 2022

Academic Editor: Punit Gupta

Copyright © 2022 Sandhya Sharma et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The challenges involved in the traditional cloud computing paradigms have prompted the development of architectures for the next generation cloud computing. The new cloud computing architectures can generate and handle huge amount of data, which was not possible to handle with the help of traditional architectures. Deep learning algorithms have the ability to process this huge amount of data and, thus, can now solve the problem of the next generation computing algorithms. Therefore, these days, deep learning has become the state-of-the-art approach for solving various tasks and most importantly in the field of recognition. In this work, recognition of city names is proposed. Recognition of handwritten city names is one of the potential research application areas in the field of postal automation. For recognition using a segmentation-free approach (Holistic approach). This proposed work demystifies the role of convolutional neural network (CNN), which is one of the methods of deep learning technique. Proposed CNN model is trained, validated, and analyzed using Adam and stochastic gradient descent (SGD) optimizer with a batch size of 2, 4, and 8 and learning rate (LR) of 0.001, 0.01, and 0.1. The model is trained and validated on 10 different classes of the handwritten city names written in Gurmukhi script, where each class has 400 samples. Our analysis shows that the CNN model, using an Adam optimizer, batch size of 4, and a LR of 0.001, has achieved the best average validation accuracy of 99.13.

1. Introduction

Cloud computing is equipped with good solutions to meet the increasing demand of data storage. It has provided users with various benefits, thereby reducing the efforts to manage the data in an efficient and effective manner. The cloud relies mainly on the data centers, and the data centers, which are located far away from the user, are further linked together to build data center networks. So, the next generation cloud computing architectures have made it possible to process the data closer to the user instead of processing it at the data center. These emerging paradigms of cloud computing

generate huge amount of data. But there is the possibility that this huge data may not be analyzed to uncover the new information [1]. Various algorithms can be applied for the analysis of data. Performance of traditional algorithms decreases when the amount of data increases. But, the performance of these deep learning algorithms improves when the amount of data increases. Deep learning is the subset of machine learning, and it has gained the attention of various researchers due to its strength of handling huge amount data. This is the reason why applications of deep learning in the emerging paradigms of cloud computing are also gaining the attention of research community. These days, all the

information in our lives is being processed through electronic devices. As computers are involved in every field, there is a requirement to transfer all the information between humans and computers with the help of some efficient and fast algorithms. So, there exists text recognition, which helps in providing an interface, so that humans and computers can interact with each other. An example of such systems lies in the digitization of documents images, which may be handwritten or printed. Such systems are relevant in many applications like automatic form processing, postal automation, cheques processing, preservation of historical documents, etc. In this proposed work, recognition of city names that are written in Gurmukhi script is implemented and is one of the application research areas of postal automation. As manual sorting of mails is a cumbersome as well as labor-intensive task due to the high labor cost involved in the process. So, it is required to develop a postal automation system that can read all the necessary fields of the postal document and can help in reaching the document to its destination. Gurmukhi script is the Punjab state's official language, which is used by the Punjab state's government officials to communicate their documents. An example of such a document is shown in Figure 1, in which the address field of the document to be posted is written in Gurmukhi [2] script, and the city name "Mohali" is also highlighted.

Several researchers are already working in the field of postal automation, and research is already going on for the recognition of various scripts like Devanagari, Bangla, English, Russian, etc., but still no postal automation system exists for Gurmukhi script. This proposed work aims to employ a deep learning technique based on CNN for the recognition of Gurmukhi handwritten city names. Out of various deep learning techniques, CNN is widely used for the purpose of text recognition [1]. The most important challenge for the recognition of Gurmukhi script is its cursive writing style, and the characters are so closely written, which makes their segmentation a difficult task. So, this paper has proposed the recognition technique using a segmentation-free approach, which is known as the Holistic approach. Figure 2 is showing the sample of handwritten Gurmukhi script.

2. Related Work

Emerging cloud computing paradigms have helped the user by generating data at the edge of the network without being transferred at the far cloud center. In this section, a literature survey is given on the use of ConvNet in the emerging cloud computing architectures. Huang et al. [3] have proposed a ConvNet model using edge computing algorithm for the classification of various mosquitoes. A device was developed for the detection of mosquitoes and preprocessing of videos before sending them to data center. Later, a ConvNet model is employed for the classification of mosquitoes. Similarly, Liu et al. [4] have also proposed a CNN based ConvNet model for the recognition of food. A server equipped with the centos 7.0 was used at the cloud. Later, again, the CNN based model ConvNet is implemented for the purpose of

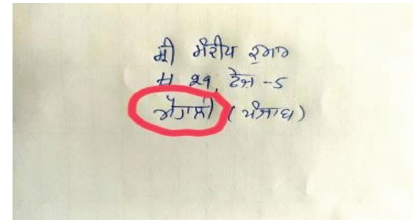


FIGURE 1: Postal document with city name written in Gurmukhi.

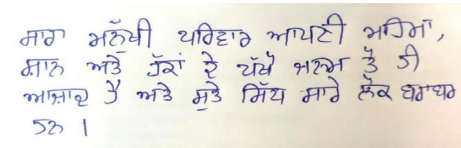


FIGURE 2: Sample of Gurmukhi script.

identification and classification of food images. Azimi et al. [5] have used ConvNet for the diagnosis of heart diseases. For the reading of files, a local Wi-Fi is programmed and for the uploading POST request to the edge device, a machine with Apache server is used. Hosain et al. [6] have proposed ConvNet model using edge cloud computing, in which data is sent to the center using radio access technology. For the formation of edge cloud, a MEC server is used, and later, a CNN based ConvNet is used for the classification. Similarly, in the field of postal automation, i.e., for the recognition of various fields like city name, pin code, and street name, a huge amount of data is required. This huge data can be processed efficiently by deep learning algorithms. Among the relevant works, recognition of such fields is done in various scripts using character-level recognition, in which the word is segmented into individual characters before recognition (analytical approach) or word level recognition, which is segmentation-free approach (holistic approach). Pal et al. [7] have employed character-level recognition using machine learning for the recognition of multilingual script-based city names for Indian postal automation. Similarly, Thadchanamoorthy et al. [8] have proposed a technique for the recognition of Tamil city names. The accuracy obtained is 96.89%. Some have worked on the recognition of pin codes only, which are written in different scripts like English, Bangla, etc. One such example is Vajda et al. [9], where the authors have recognized pin codes written in Bangla as well as in English script using a nonsymmetric half-plane hidden Markov model and achieved an accuracy of 94.13% and 93%, respectively. Sahoo et al. [10] have implemented a holistic approach for the recognition of city names written in Bangla script using shape-context features, while multiclassifiers are employed for the classification purpose. The datasets used here are the large datasets that are stored on the cloud. There are some other examples where authors have implemented only a holistic approach for the recognition of various scripts like English, Bangla, and Arabic and achieved an accuracy of 90.3%, 83.64%, and 63% [11–13]. Manchala et al. [9] have used NN for the recognition of English script using the Holistic approach. Similarly, Bhowmik et al. [10] have proposed a technique for the recognition of Bangla script.

Wahbi et al. [11] have worked on the recognition of Arabic script again by holistic approach, and the model employed for the recognition is hidden Markov model. Few other authors have also used the holistic approach for the recognition of text, in which features are manually extracted [14, 15], while others have used CNN for character recognition, in which features are automatically extracted [16]. The aim of this proposed work is to employ CNN with the holistic approach for the recognition of Gurmukhi handwritten city names. All these techniques have been employed on huge datasets, which are stored on cloud.

2.1. Contributions of the Proposed Work

- (i) A dataset having 4000 samples of the handwritten images in the Gurmukhi script for the 10 city names has been generated.
- (ii) A CNN model for the automation of postal system for the recognition of Gurmukhi handwritten city names has been prepared. The model can recognize all the 10 city names with an average validation accuracy of 99.13%.

3. Present Work

In this proposed work, a dataset of 4000 Gurmukhi handwritten city names is created for 10 different classes (city names), where each class is having 400 samples that can be fed to the model. As manual sorting of postal documents is a labor-intensive task, so, recognition of city names will help in the automation of postal system for the state of Punjab. Finally, the designed model has predicted various parameters for the recognition of city names. The methodology followed for the recognition is shown in Figure 3.

3.1. Dataset. For the preparation of dataset, each sample is written 10 times by 40 different writers generating 4000 samples. For collecting the dataset, two sheets were given to each writer to write each word five times on each sheet, generating total 10 samples for each word from both sheets. So, each writer generated 100 samples. Writers were selected from different age groups, educational levels, and different dialects. Writers were free to use any colored pen.

3.2. Digitization and Preprocessing of Dataset. For the digitization of collected samples, a scanner with 300 dpi was used to scan the collected sheets containing samples. For the preprocessing of dataset, each scanned sheet is converted into gray scale image, and later, normalization is performed. Further, brightness adjustment, contrast adjustment, and intensity level adjustment are performed to improve the quality of the image before cropping the word samples from the digitized sheets. Adobe Photoshop is used for the purpose of contrast adjustment, brightness adjustment, intensity level adjustment, and cropping of images. Later, cropped samples were placed in their respective folders that were named as per the city name. Later, the prepared dataset

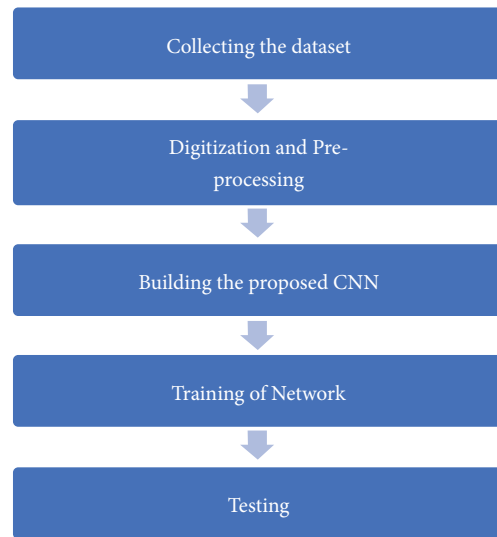


FIGURE 3: Methodology of the proposed work.

is stored on the cloud as it has to be accessed using the deep learning network.

Once the preprocessing is done, the dataset is divided into training and validation dataset [17, 18]. 80% of the data is kept for training the model, and 20% is kept for validating the performance of the model. Table 1 below is showing the city name and its corresponding handwritten digitized image in the Gurmukhi script.

3.3. Data Augmentation. Data augmentation is the technique that helps in increasing the available data. So, available data is further increased by flipping or rotating the images, and data augmentation is the inbuilt function of the proposed model. Rotation of city name “Amritsar” is shown in Figure 4.

3.4. Model Design. To build the CNN model, three layers are required: (i) convolution layer; (ii) pooling layer; (iii) output layer. The primary function of the first convolution layer is to apply the predefined filter weight to derive the features from an image. Based upon the weighting filter used, the number of feature maps is produced. The complexity of the extracted features keeps on increasing with the increasing model depth, while the last convolution layer of the model generates the feature maps, which are much closer to the required recognition task. The next layer is the pooling layer, and the most commonly used pooling technique is max pooling. This further helps in preserving the features by selecting the maximum value as this has the closest similarity to the required features. The pooling layer also helps in reducing the size of the image by getting rid of the features, which are not important. The last layer is the fully connected layer, and from here, the output classes are obtained. The proposed CNN model is shown in Figure 5.

The first convolution layer used in this proposed work has 32 filters of size 3×3 with a stride of 1×1 , 32 feature maps are derived from this, and the convolution layer is

TABLE 1: Ten city names with their corresponding handwritten digitized image in Gurmukhi.

S. no.	City name	Written by "writer 1"	Written by "writer 2"
1	Amritsar	ਅੰਮ੍ਰਿਤਸਰ	ਅੰਮ੍ਰਿਤਸਰ
2	Fazilka	ਫ਼ਤਿਹਗੜ੍ਹ	ਫ਼ਤਿਹਗੜ੍ਹ
3	Hoshiarpur	ਹੁਸ਼ਿਆਰਪੁਰ	ਹੁਸ਼ਿਆਰਪੁਰ
4	Jalandhar	ਜਲੰਧਰ	ਜਲੰਧਰ
5	Ludhiana	ਲੁਧਿਆਣਾ	ਲੁਧਿਆਣਾ
6	Mansa	ਮਾਨਸਾ	ਮਾਨਸਾ
7	Mohali	ਮੁਹਾਲੀ	ਮੁਹਾਲੀ
8	Muktsar	ਮੁਕਤਸਰ	ਮੁਕਤਸਰ
9	Pathankot	ਪਠਾਨਕੋਟ	ਪਠਾਨਕੋਟ
10	Patiala	ਪਟਿਆਲਾ	ਪਟਿਆਲਾ

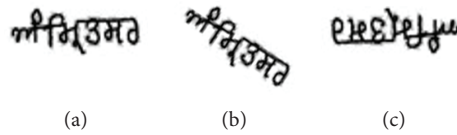


FIGURE 4: Augmentation. (a) Actual image. (b) Rotated by 60°deg. (c) Rotated by 180°deg.

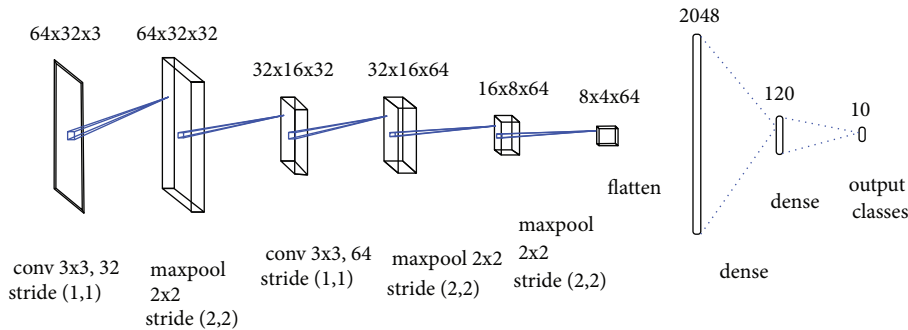


FIGURE 5: Proposed CNN model.

followed by the ReLU activation function. The obtained feature maps are then passed to the max pooling layer of $2 * 2$ filter size and a stride of $2 * 2$, which means that the pooling layer has reduced the size of the feature map by a factor of 2. The obtained pooled feature maps are passed to the next convolution layer, which has 64 filters with a size of $3 * 3$ and a stride of $1 * 1$, which is again followed by max pooling layer of size $2 * 2$ and a stride of $2 * 2$, which is further followed by another max pooling layer of the same size. Lastly, a fully connected layer is introduced with the SoftMax activation function, having 2048 neurons in the input layer, then 120 neurons in the middle layer, and 10 neurons in the last, which is the output layer. The fully connected layer transforms the obtained feature maps to the 10 classes. Rectified linear unit (ReLU) is used as the activation function for all the layers in the model, except for the pooling layers.

4. Experiments and Results

The proposed model is implemented on the dataset of 4000 images using Python with the help of Keras and Tensorflow, which are machine learning libraries.

4.1. Experimental Setup and Performance Metrics Used. The efficiency of the model is impacted by various parameters, but in this paper, three important parameters are considered: the optimizer, LR, and the batch size of the model. The optimizer is used to update the network weights; also, the choice of optimizer means good results in minutes, hours, or days. LR tells how rapidly the neuron weights will be adapted, while the batch size tells the number of samples that are processed before the model is being updated. The

proposed model is analyzed using two different optimizers, Adam and stochastic gradient descent (SGD), three different LRs, 0.001, 0.01, and 0.1, and the batch size: 2, 4, and 8. It is known that the batch size is an important hyperparameter for deep learning systems. Large batch size helps in speeding up the computation, but it leads to poor generalization [19]. So, it is always preferable to use a small batch size. This proposed model has given good generalization with batch size of 2, 4, and 8 only, while the accuracy drastically reduced when the batch size is further increased. Response time for the training and validation of the model varies from model to model, depending on the dataset to be trained, batch size, and LR, and it also depends on the hardware of the system used like CPU, GPU, RAM, etc. [20]. To evaluate the proposed CNN model, various parameters are calculated like training and validation loss, validation accuracy [21, 22], precision, and recall also. All these parameters are calculated using the different metrics of the confusion matrix, which are true positive (TP), false positive (FP), true negative (TN), and false negative (FN). The parameters are defined as follows:

4.1.1. Accuracy. Accuracy is defined as the ratio of the number of correct predictions made by the model to the total number of predictions made as shown in

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})}. \quad (1)$$

4.1.2. Precision. It is a metric that tells about the proportion of cases that report true and are actually true as shown in

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}. \quad (2)$$

4.1.3. Recall (Sensitivity). The recall measures the ability of the designed model to detect positive samples. It is calculated as the sum of true positive across all classes divided by the sum of true positive and false negative across all classes as shown in

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}. \quad (3)$$

4.2. Results Obtained Using Adam Optimizer with a Batch Size of 4. In this section, various parameters are obtained on three different LRs, while the Adam optimizer is used with a batch size of 4.

4.2.1. Results Obtained with a LR of 0.001. Table 2 shows the values of various parameters obtained with the LR of 0.001, while the optimizer used is Adam with a batch size of 4. Maximum obtained validation accuracy on the validation dataset is 99.8% with a minimum validation loss of 0.01, and the average obtained validation accuracy is 99.13%.

Figure 6 shows the different parameters convergence plots for the training dataset, as well as for the validation dataset. Y-axis is showing the particular value obtained, while the X-axis is the number of epochs, for which the model is trained. Validation accuracy is the main parameter; while checking the model's performance for the recognition of text, it can be observed that the accuracy plot is almost increasing after the run of few epochs. The maximum validation accuracy obtained is 99.8%. The value of loss should be less, and the minimum value of the loss obtained is 0.01 and 0.07 for the validation and training dataset. Values of other parameters are approaching 1, which shows that the designed model is reasonably good.

Results can also be analyzed by plotting the confusion matrix. Figure 7 is showing the plot of the confusion matrix for multiclassification results obtained in Figure 6. On the X-axis, the predicted labels are depicted, while on the Y-axis, true labels are depicted. The confusion matrix tells the information about the actual (true) and the predicted classification done by the designed classification model. The highlighted value in blue boxes represents the true positive values, which tell how much the designed model has correctly predicted the positive classes as positive [23]. For example, for the city Amritsar, the designed model has correctly predicted all the 80 samples, while, for the city Fazilka, the model has correctly predicted 78 samples and incorrectly predicted 2 samples as Ludhiana, which can be observed in Figure 7.

4.2.2. Results Obtained with a LR of 0.01. Now, the LR is changed to 0.01, while the optimizer used is Adam. Table 3 shows the outcomes of various parameters, when the LR is changed from 0.001 to 0.01, while other parameters are kept the same. From the obtained results, it can be observed that the LR of the model impacts the accuracy results. The validation accuracy obtained on 10th epoch is 99%, and the values of training loss, validation loss, validation precision, and validation recall are 0.11, 0.01, 0.99, and 0.99, respectively. The average obtained validation accuracy is 96.95%, which is less than the average validation accuracy obtained, with the LR of 0.001, which was 99.13%. Figure 8 shows the parameters convergence plot, and the confusion matrix for the same is shown in Figure 9, where it has misclassified 4 classes. It can be observed from Figure 8 that the obtained plots are not so linear as compared to the plots obtained in Figure 6, while linearity can be observed in the last few epochs only [24].

4.2.3. Results Obtained with a LR of 0.1. Table 4 shows the outcomes of various parameters, when the LR is further changed from 0.01 to 0.1, while other parameters are kept the same. On the 10th epoch, maximum validation accuracy obtained is 99.3%. The obtained values for training loss, validation loss, validation precision, and validation recall are 0.10, 0.00, 0.99, and 0.99. The average obtained validation accuracy is 98.17%, which is less than the average validation accuracy obtained with the LR of 0.001, which was 99.13%. Figure 10 shows the parameters convergence plot, and the

TABLE 2: Results obtained with a LR of 0.001 by employing Adam optimizer.

Epoch	Training loss	Validation loss	Validation accuracy (%)	Validation precision	Validation recall
1	0.49	0.08	98.5	0.98	0.97
2	0.19	0.07	98.3	0.98	0.97
3	0.13	0.05	99.2	0.99	0.99
4	0.12	0.04	99.1	0.99	0.98
5	0.09	0.02	99.3	0.99	0.98
6	0.09	0.01	99.5	0.99	0.99
7	0.09	0.03	99.6	0.99	0.98
8	0.09	0.02	99.6	0.99	0.99
9	0.09	0.02	99.8	0.99	0.99
10	0.07	0.01	99.8	0.99	0.99

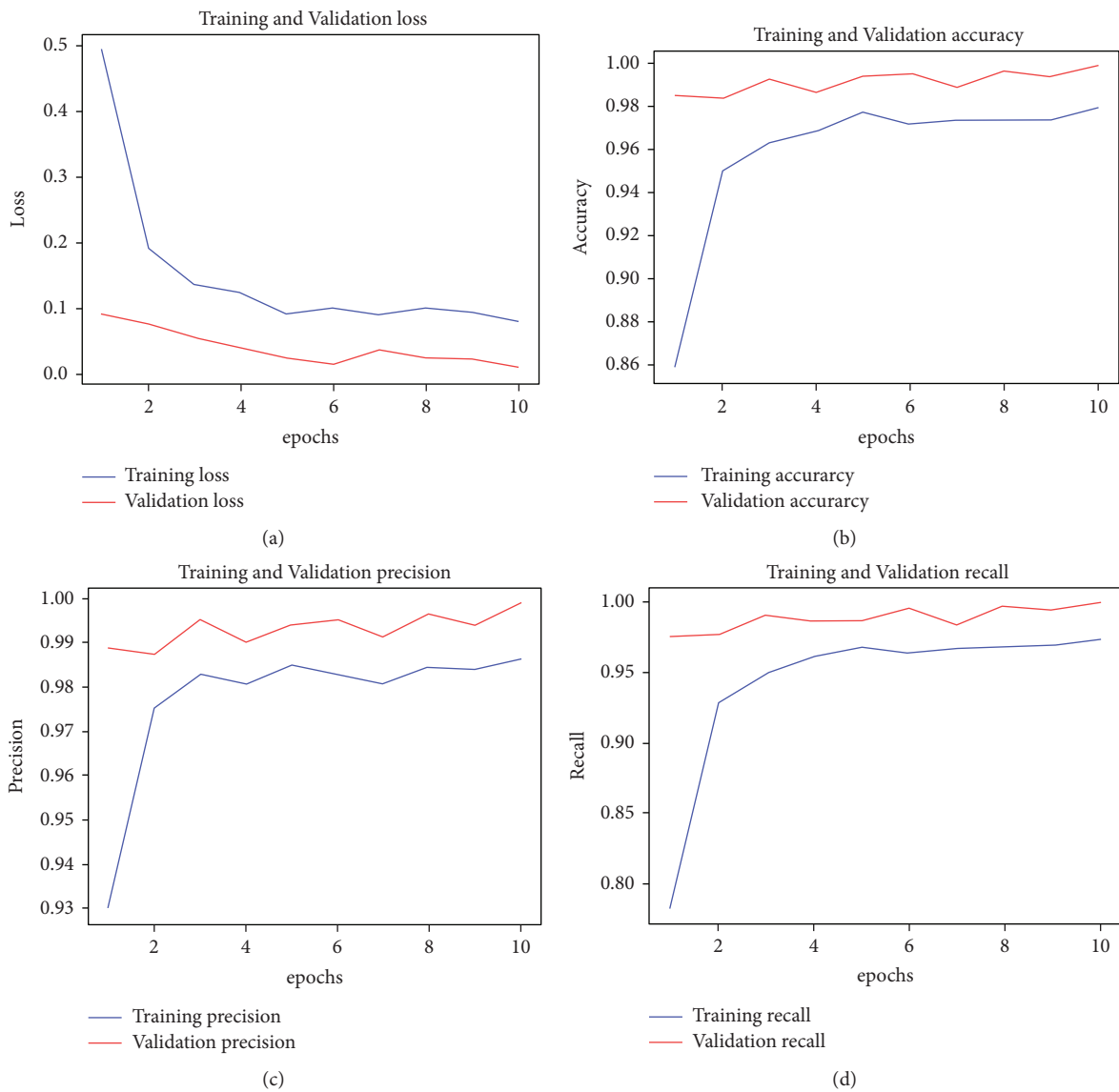


FIGURE 6: Plots obtained with a LR of 0.001 by employing Adam optimizer. (a) Training and validation loss plot. (b) Training and validation accuracy plot. (c) Training and validation precision plot. (d) Training and validation recall plot.

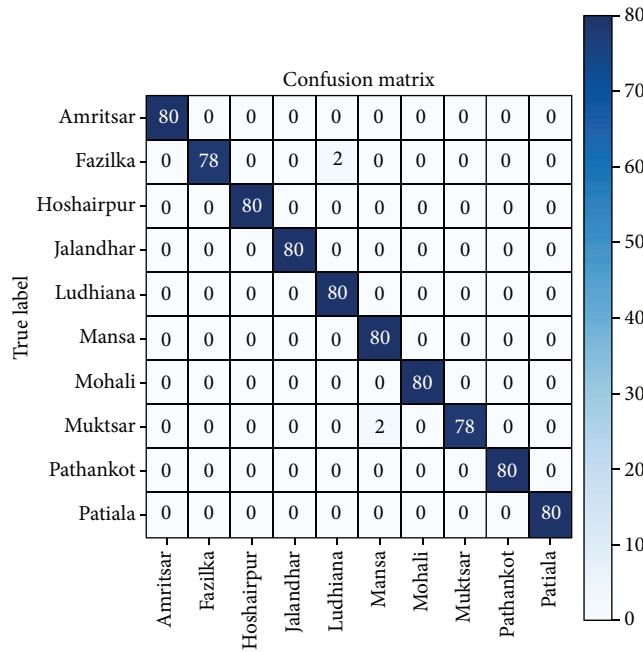


FIGURE 7: Confusion matrix for LR 0.001.

TABLE 3: Results obtained with a LR of 0.01 by employing Adam optimizer.

Epoch	Training loss	Validation loss	Validation accuracy (%)	Validation precision	Validation recall
1	0.58	0.12	95	0.95	0.95
2	0.28	0.74	90	0.90	0.89
3	0.18	0.19	97	0.97	0.97
4	0.15	0.04	98	0.98	0.98
5	0.12	0.25	94	0.95	0.94
6	0.14	0.38	97	0.96	0.96
7	0.10	0.49	99	0.99	0.99
8	0.14	0.32	98	0.98	0.98
9	0.10	0.24	99	0.99	0.99
10	0.11	0.01	99	0.99	0.99

confusion matrix for the same is shown in Figure 11. In Figure 10, all the plots are linearly varying, the loss plot is almost decreasing, and other plots are approaching to 1.

From Figure 11 of confusion matrix for LR 0.1, it can be observed that the proposed model has misclassified few more classes as compared to the previous confusion matrix. 56 samples of the city “Fazilka,” 73 samples of the city Hoshiarpur, 78 of Jalandhar, and so on are correctly predicted.

It can be observed from Tables 2, 3, and 5 that the LR of the model impacts the accuracy results obtained for recognition. A LR of 0.001 has generated the average validation accuracy of 99.13%, which is highest as compared to the validation accuracies obtained by other LRs.

4.2.4. *Optimal LR Selection with Adam Optimizer.* Analysis of the validation accuracy obtained from Tables 2, 3, and 5 is done in Figure 12. It shows that the Adam optimizer with a LR of 0.001 has performed better for the available dataset. Figure 12 shows the validation accuracy obtained on

10 epochs for all the three LRs. Y-axis shows the validation accuracy obtained, while the X-axis shows three different LRs on 10 epochs. It can be observed on the 10th epoch that the validation accuracy obtained with a LR of 0.001 is the highest as compared to the LR of 0.01 and 0.1. It can also be observed that the validation accuracy is high for most of the epochs using LR of 0.001. Figure 13 shows some of the misclassified results obtained by Adam optimizer. City name “Ludhiana” is misclassified as “Patiala” in Figure 13(a), while city “Fazilka” is misclassified as “Ludhiana” in Figure 13(b).

From the above discussions of the accuracy results obtained using Adam optimizer with a LR of 0.001, 0.01, and 0.1, it can be observed that the proposed model has achieved the best validation accuracy with a LR of 0.001, as the proposed model has also misclassified few images. Results for the same are shown in Figure 13.

4.3. *Results Obtained Using Adam Optimizer: LR of 0.001 on Different Batch Sizes.* It has been analyzed from Figure 12 that LR of 0.001 has achieved better validation accuracy as

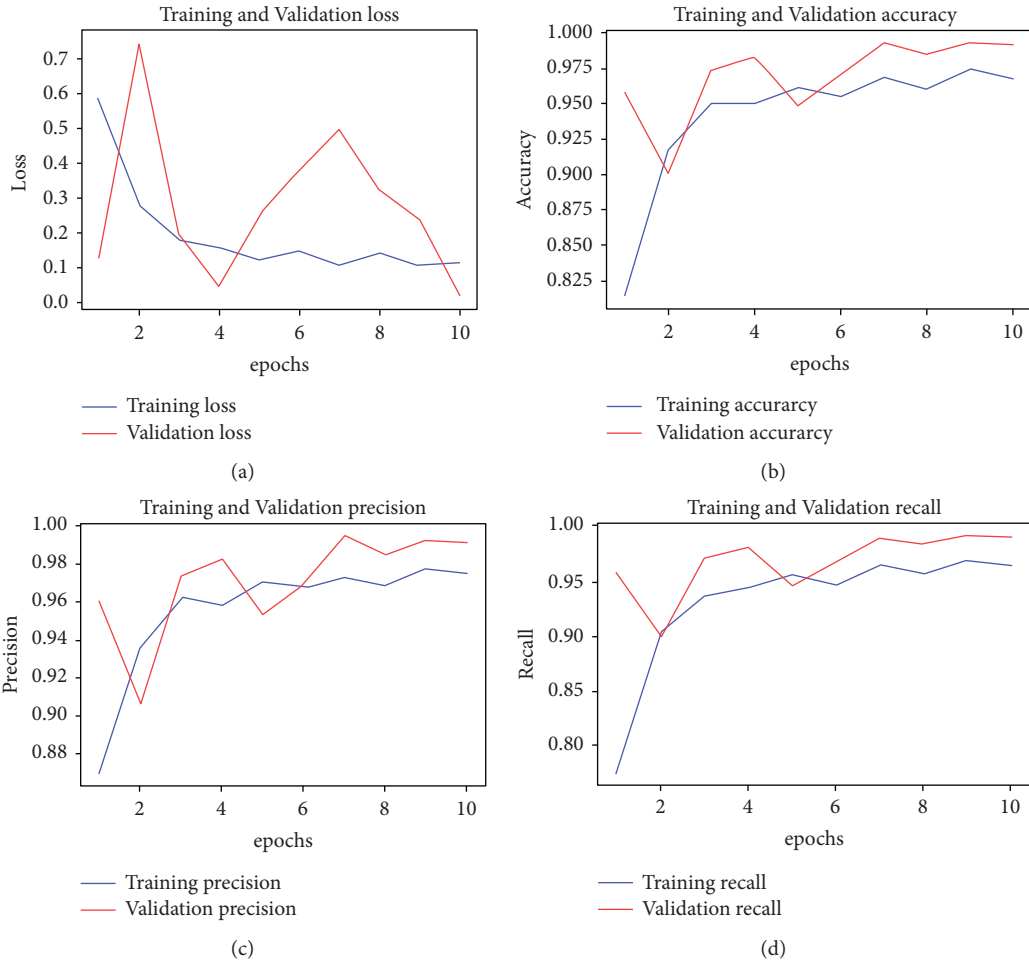


FIGURE 8: Plots obtained with a LR of 0.01 by employing Adam optimizer. (a) Training and validation loss plot. (b) Training and validation accuracy plot. (c) Training and validation precision plot. (d) Training and validation recall plot.

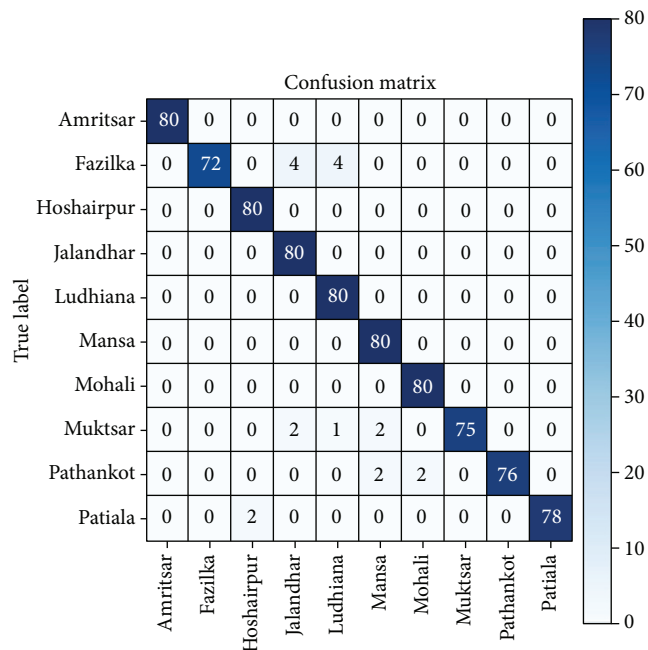


FIGURE 9: Confusion matrix for LR 0.01.

TABLE 4: Results obtained with a LR of 0.1 by employing Adam optimizer.

Epoch	Training loss	Validation loss	Validation accuracy (%)	Validation precision	Validation recall
1	1.12	0.17	93.5	0.94	0.92
2	0.32	0.05	98.3	0.98	0.98
3	0.21	0.01	98.5	0.99	0.99
4	0.18	0.11	99	0.99	0.99
5	0.10	0.02	99.5	0.99	0.99
6	0.14	0.00	99.6	0.99	0.99
7	0.11	0.01	99.5	0.99	0.99
8	0.11	0.02	99.1	0.99	0.98
9	0.09	0.02	99.4	0.99	0.98
10	0.10	0.00	99.3	0.99	0.99

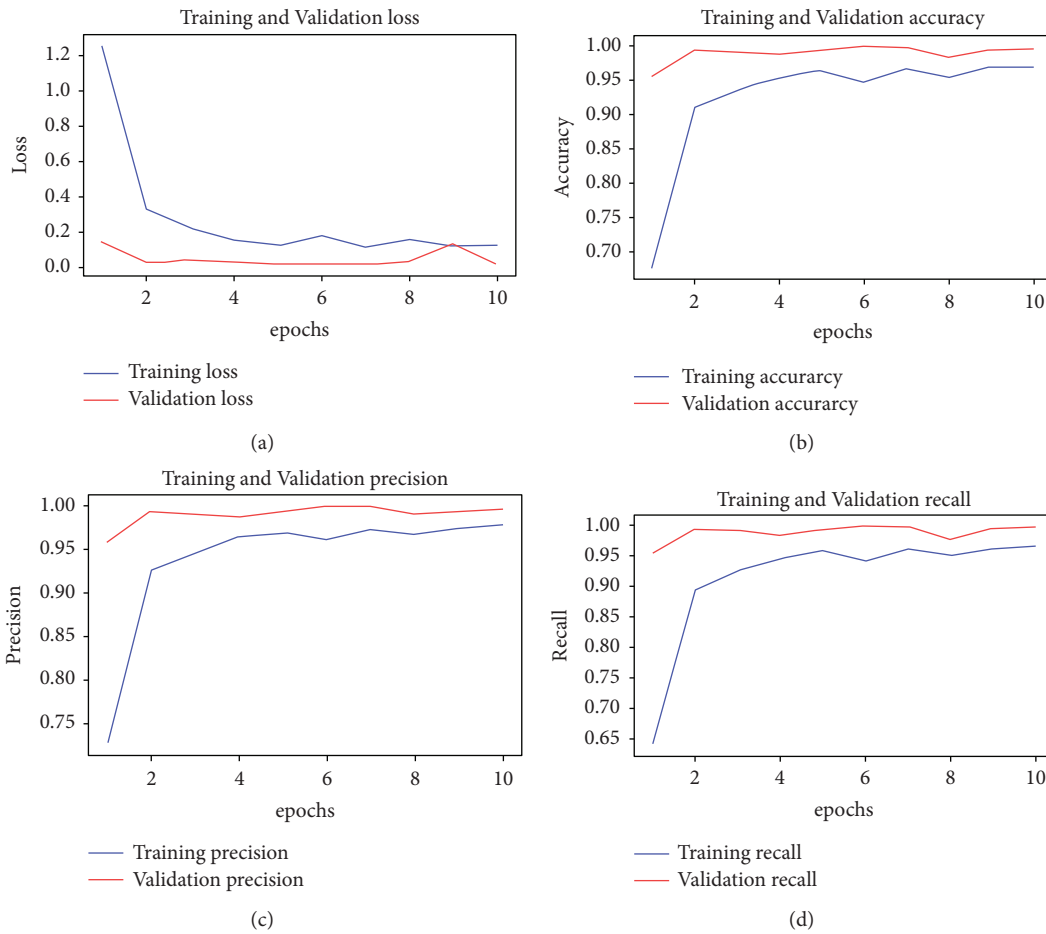


FIGURE 10: Plots obtained with a LR of 0.1 by employing Adam optimizer. (a) Training and validation loss plot. (b) Training and validation accuracy plot. (c) Training and validation precision plot. (d) Training and validation recall plot.

compared to the LR of 0.01 and 0.1. It has been observed that the batch size of the model also impacts the accuracy results [25]. Now, the model is analyzed using three different batch sizes (2, 4, and 8) with Adam optimizer and a LR of 0.001. Table 6 shows the various results obtained by changing the batch size, while the LR of 0.001 is used. It can be observed from the table that batch size of 4 has given good accuracy results as compared to batch size of 2 and 8. The best average validation accuracy achieved is 99.13% when the batch size is

kept at 4, while it is 97.24% and 92.07% with a batch size of 8 and 2 with a LR of 0.001.

4.3.1. Comparison of Three Different Batch Sizes Using Adam with a LR of 0.001. It can be observed from Figure 12 that LR of 0.001 has given good results as compared to LR of 0.1 and 0.01. In Table 4, various parameters are obtained using three different batch sizes. Average validation accuracy obtained

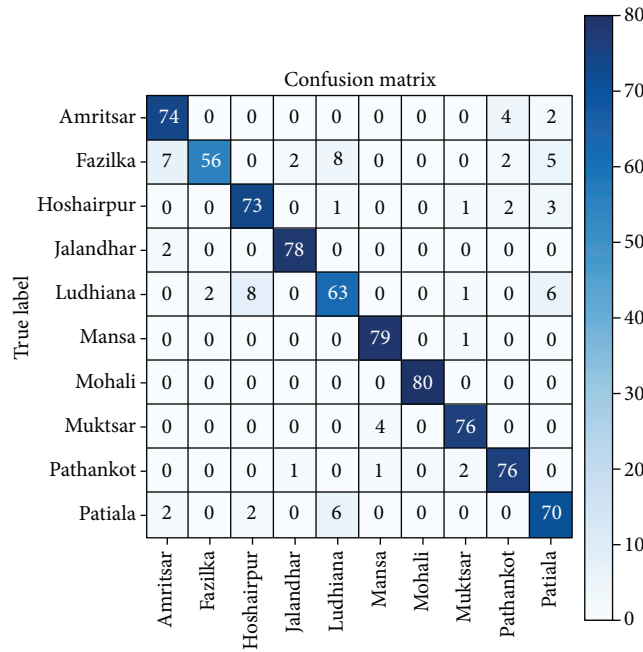


FIGURE 11: Confusion matrix for LR 0.1.

TABLE 5: Layered architecture for each layer.

Layers	Input image size	Filter size	No of filters	Activation function	Output without padding	Parameters
Input image	64 * 32 * 3					
Convolution	64 * 32 * 3	3 * 3	32	ReLU	64 * 32 * 32	896
Max Pooling	64 * 32 * 32	Pool size (2 * 2)	—	—	32 * 16 * 32	0
Convolution	32 * 16 * 32	3 * 3	64	ReLU	32 * 16 * 64	18496
Max Pooling	32 * 16 * 64	Pool size (2 * 2)	—	—	16 * 8 * 64	0
Max Pooling	16 * 8 * 64	Pool size (2 * 2)	—	—	8 * 4 * 64	0
Flatten	8 * 4 * 64	—	—	—	1024	—
Dense	1024	120	—	ReLU	120	123000
Dense	120	—	—	SoftMax	10	1210

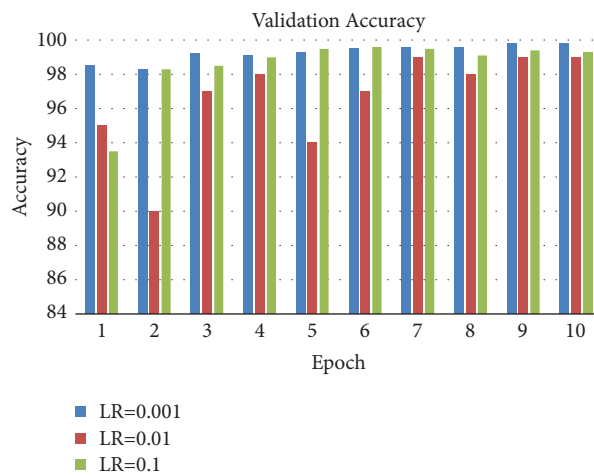


FIGURE 12: Analysis of validation accuracy on three different LRs using Adam Optimizer.

by batch size 2 is 92.07%, batch size 4, 99.13%, and batch size 8, 97.24%. Now, validation accuracy obtained on 10 different epochs of Table 4 is compared in Figures 14 and 15 by three

different batch sizes. In Figure 14, Y-axis is representing the validation accuracy obtained, while the X-axis is representing the three different batch sizes on 10 epochs and in

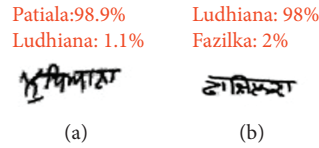


FIGURE 13: Misclassification results. (a) “Ludhiana” is misclassified as “Patiala.” (b) “Fazilka” is misclassified as “Ludhiana.”

TABLE 6: Results obtained for different batch sizes.

Batch size	Epochs	Training loss	Validation loss	Validation accuracy (%)	Average validation accuracy
2	1	1.59	0.70	80.2	92.07%
	2	1.22	0.39	85.1	
	3	1.08	0.31	92.4	
	4	0.92	0.30	92.5	
	5	0.91	0.16	94.2	
	6	0.76	0.24	93.5	
	7	0.78	0.11	95.4	
	8	0.76	0.43	92.6	
	9	0.73	0.10	97.1	
	10	0.68	0.08	97.5	
4	1	0.49	0.08	98.5	99.13%
	2	0.19	0.07	98.3	
	3	0.13	0.05	99.2	
	4	0.12	0.04	99.1	
	5	0.09	0.02	99.3	
	6	0.09	0.01	99.5	
	7	0.09	0.03	99.6	
	8	0.09	0.02	99.6	
	9	0.09	0.02	99.8	
	10	0.07	0.01	99.8	
8	1	0.23	0.54	78.6	97.24%
	2	0.03	0.02	99.1	
	3	0.02	0.03	98.5	
	4	0.01	0.00	99.2	
	5	0.01	0.00	99.2	
	6	0.03	0.04	97.8	
	7	0.02	0.02	99.1	
	8	0.01	0.02	99.7	
	9	0.00	0.01	99	
	10	0.00	0.01	99	

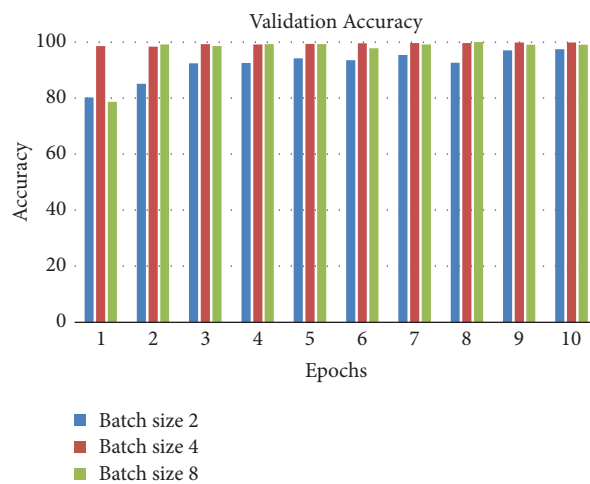


FIGURE 14: Comparison of validation accuracy using batch sizes 2, 4, and 8 for Adam optimizer and with a LR of 0.001.

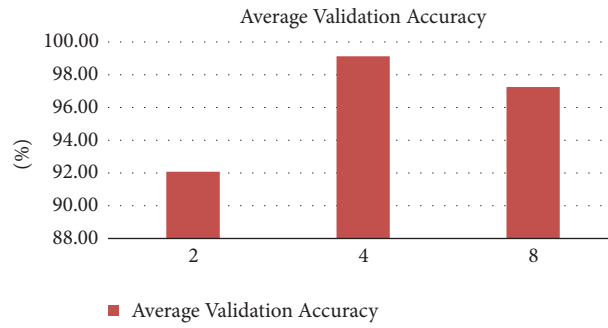
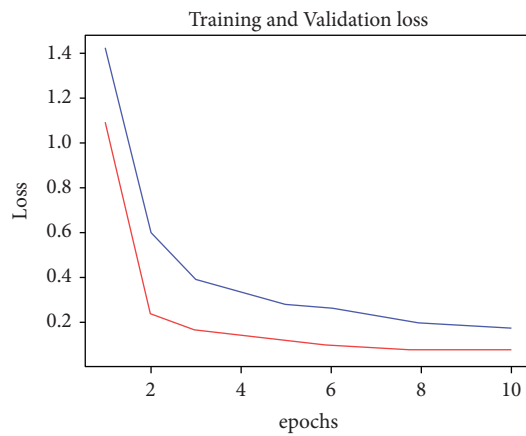


FIGURE 15: Comparison of average validation accuracy using batch sizes 2, 4, and 8 for Adam optimizer and with a LR of 0.001.

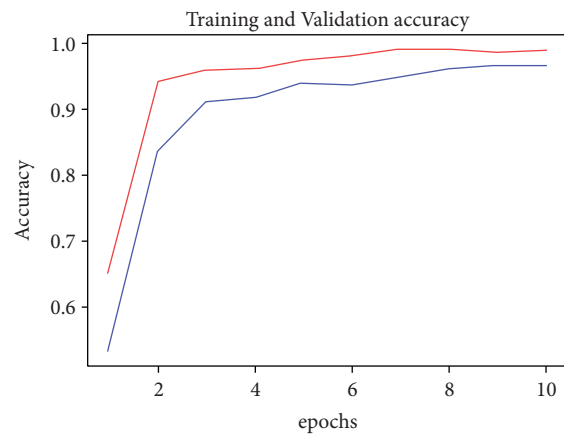
TABLE 7: Results obtained using SGD optimizer with a LR of 0.001.

Epoch	Training loss	Validation loss	Validation accuracy (%)	Validation precision	Validation recall
1	1.4	1.08	65.1	0.84	0.42
2	0.60	0.23	94.1	0.95	0.91
3	0.38	0.15	95.7	0.96	0.94
4	0.33	0.13	96	0.97	0.94
5	0.27	0.11	97.3	0.97	0.96
6	0.25	0.07	99.0	0.99	0.98
7	0.22	0.07	99.0	0.99	0.98
8	0.19	0.07	99.0	0.98	0.98
9	0.18	0.07	98.8	0.98	0.98
10	0.17	0.07	98.8	0.98	0.98



— Training loss
— Validation loss

(a)



— Training accuracy
— Validation accuracy

(b)

FIGURE 16: Continued.

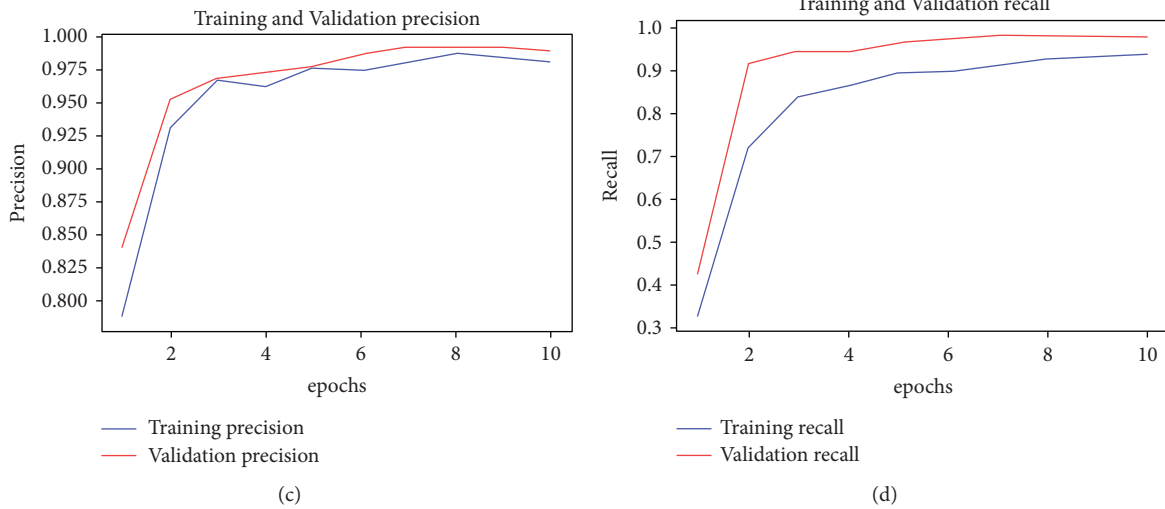


FIGURE 16: Plots obtained with a LR of 0.001 by employing SGD optimizer. (a) Training and validation loss plot. (b) Training and validation accuracy plot. (c) Training and validation precision plot. (d) Training and validation recall plot.

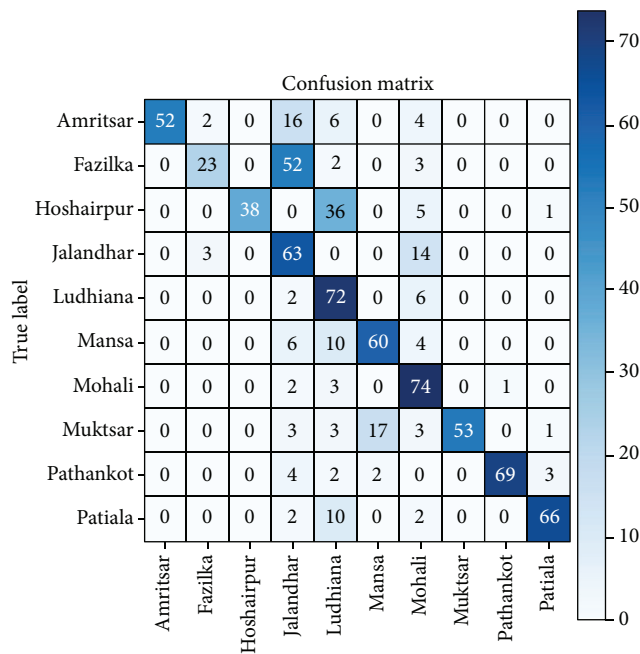


FIGURE 17: Confusion Matrix for SGD optimizer with a LR of 0.001.

Figure 15, Y-axis is representing the average validation accuracy obtained, while the X-axis is representing the batch sizes 2, 4, and 8. It can be observed from Figure 14 that batch size 4 has given higher accuracies as compared to other batch sizes, while Figure 15 shows that the high average validation accuracy is given by a batch size of 4.

4.4. Results Obtained Using SGD Optimizer with a LR of 0.001 and Batch Size of 4. The optimizer employed in CNN model also affects the accuracy obtained. In this section, the results are obtained using SGD optimizer with a LR of 0.001 and a

batch size of 4. LR of 0.001 and batch size of 4 are chosen based upon the best accuracy results obtained by them in the previous sections. Table 7 shows the results for various parameters obtained using SGD optimizer, while Figure 16 shows the parameters convergence plot, and Figure 17 shows the confusion matrix for the same. Figure 16 shows that the plots are almost linear for all the parameter values obtained in Table 7. The plot in Figure 16(a), training and validation loss, is approaching value “0” with each epoch, while plots in Figures 16(b)–16(d) for training and validation accuracy, precision, and recall are approaching “1.” The maximum validation accuracy obtained in the last epoch is 98.8%,

TABLE 8: Comparison of SGD optimizer with three different batch sizes.

Batch size	Epochs	Training loss	Validation loss	Validation accuracy	Average validation accuracy
2	1	1.96	1.11	63.2	87.6%
	2	1.51	0.84	76.1	
	3	1.25	0.83	75.3	
	4	1.08	0.31	91.1	
	5	1.03	0.30	90.7	
	6	0.85	0.26	93.6	
	7	0.88	0.21	95.0	
	8	0.78	0.10	98	
	9	0.68	0.10	97.2	
	10	0.71	0.10	97.6	
4	1	1.4	1.08	65.1	94.18%
	2	0.60	0.23	94.1	
	3	0.38	0.15	95.7	
	4	0.33	0.13	96	
	5	0.27	0.11	97.3	
	6	0.25	0.07	99.0	
	7	0.22	0.07	99.0	
	8	0.19	0.07	99.0	
	9	0.18	0.07	98.8	
	10	0.17	0.07	98.8	
8	1	1.1	0.99	64.3	93.31%
	2	0.37	0.28	92.12	
	3	0.24	0.15	95.3	
	4	0.17	0.15	96.5	
	5	0.13	0.13	96.3	
	6	0.11	0.11	96.7	
	7	0.10	0.09	97.0	
	8	0.09	0.09	98.5	
	9	0.08	0.08	98.2	
	10	0.07	0.08	98.3	

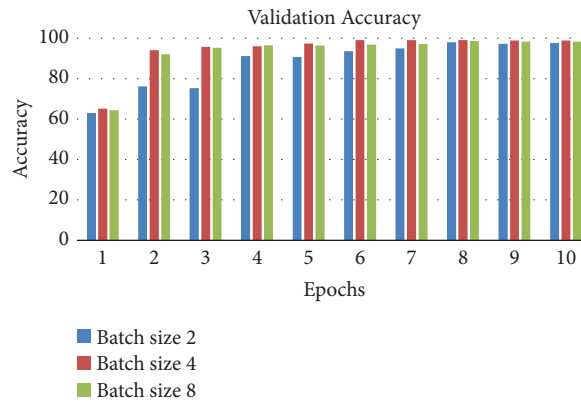


FIGURE 18: Comparison of validation accuracy using batch sizes 2, 4, and 8 for SGD optimizer with a LR of 0.001.

which is less than the best accuracy obtained by Adam optimizer.

4.4.1. Comparison of Three Different Batch Sizes Using SGD Optimizer with a LR of 0.001. The comparison of SGD optimizer on three different batch sizes is carried out in this section. Training loss, validation loss, validation accuracy, and average validation accuracy are compared using 10

epochs. The average validation accuracy obtained by batch size 2 is 87.6%, batch size 4, 94.18%, and batch size 8, 93.31%. It can be observed from Table 8 that the highest average validation accuracy is obtained by batch size 4. Figure 18 shows the comparison plot for validation accuracy using batch sizes 2, 4, and 8 for SGD optimizer with a LR of 0.001, while Figure 19 shows the comparison plot for average validation accuracy for the same. It can be concluded that, here, also batch size of 4 has worked better for the available

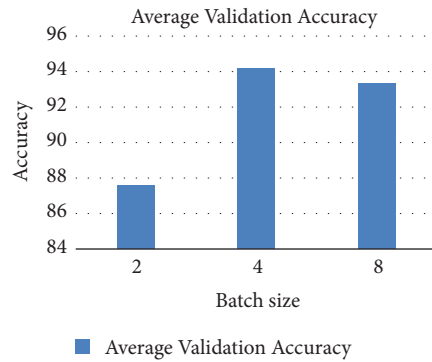


FIGURE 19: Comparison of average validation accuracy using batch sizes 2, 4, and 8 for SGD optimizer with a LR of 0.001.

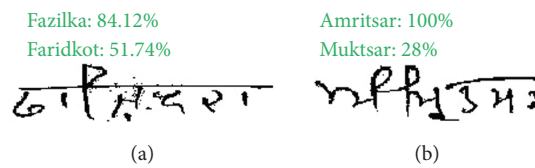


FIGURE 20: Testing of images. (a) Fazilka image. (b) Amritsar image.

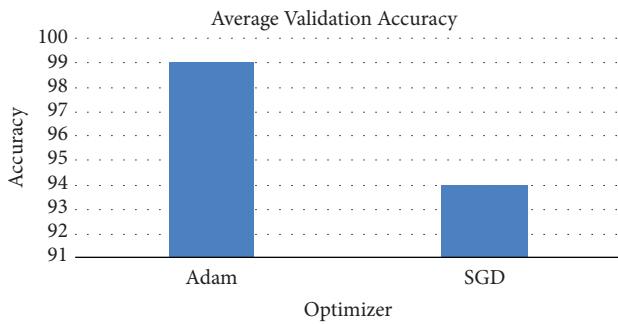


FIGURE 21: Analysis plot of average validation accuracy obtained by Adam optimizer and SGD optimizer.

dataset using SGD optimizer although the achieved validation accuracy is less than the validation accuracy achieved by Adam optimizer.

5. Testing of Some Images

In this section, some randomly selected images are given to the CNN model, which has employed Adam optimizer with a LR of 0.001 and BS of 4. Figure 20(a) shows that the image of city name “Fazilka” has been recognized with an accuracy of 84.12%, while the image of city name “Amritsar” has been recognized 100% correctly.

6. Analysis

It can be analyzed from Tables 4 and 7 that Adam optimizer with a LR of 0.001 and batch size of 4 has given the good results in terms of average validation accuracy. The best average validation accuracy obtained by Adam is 99.13%, while best average validation accuracy obtained by SGD optimizer is 94.18%. Analysis plot of average validation

accuracy obtained by Adam optimizer and SGD optimizer is shown in Figure 21. It can be concluded that Adam with a LR of 0.001 and batch size of 4 has performed much better as compared to SGD optimizer using the same LR and batch size.

7. Conclusion

Deep learning is the machine learning, which is applied to the large datasets. Deep learning requires huge amount of data to train the network, which can be stored on cloud. Therefore, cloud computing helps in making deep learning accessible and easier in handling large amount of data, and also, the training of algorithms can be easily done on the distributed hardware. It also helps in providing the access to configurations of various hardware such as FPGAs, GPUs, and high performance computing systems. It can be concluded that the emerging paradigms of cloud computing work very well with the deep learning algorithms instead of traditional algorithms. The performance of the deep learning model depends on various hyperparameters like the LR, batch size, choice of optimizers, and so on. In Tables 2, 3, and 5, various parameters are obtained using an Adam optimizer with a LR of 0.001, 0.01, and 0.1, and the obtained average validation accuracies are 99.13%, 96.9%, and 98.17%. Table 4 has also obtained the average validation accuracy by changing the batch size of 2, 4, and 8, while the LR is kept fixed at 0.001. Results are also analyzed by changing Adam optimizer to SGD optimizer for the LR of 0.001. Average validation accuracy obtained for SGD optimizer using batch size of 2, 4, and 8 is 87.6%, 94.18%, and 93.31%, while the best average validation accuracy obtained by Adam and SGD is 99.13% and 94.18%, respectively. From the above experimentation, it has been observed that Adam optimizer with a batch size of 4 and a LR of 0.001 has achieved the best

average validation accuracy of 99.13%. In the future, this model can be implemented for word recognition for other scripts also.

Data Availability

The data will be available from the author upon request (sandhya.sharma@chitkara.edu.in).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] P. Zhang, Q. Zhao, J. Gao, W. Li, and J. Lu, "Urban street cleanliness assessment using mobile edge computing and deep learning," *IEEE Access*, vol. 7, pp. 63550–63563, 2019b.
- [2] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: a new paradigm to machine learning," *Archives of Computational Methods in Engineering*, vol. 27, no. 4, pp. 1071–1092, 2020.
- [3] L.-p. Huang, M. Hong, C. Luo, S. Mahajan, and L. Chen, "A vector mosquitoes classification system based on edge computing and deep learning," in *Proceedings of the 2018 Conference on Technologies and Applications of Artificial Intelligence*, pp. 24–27, TAAI, Taichung, Taiwan, December 2018.
- [4] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, and M. Yunsheng, "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 249–261, 2018.
- [5] I. Azimi, J. Takalo-mattila, A. Anzanpour, and M. Rahmani, "Empowering healthcare iot systems with hierarchical edge-based deep learning," in *Proceedings of the International Conference on Connected Health: Applications, Systems and Engineering Technologies*, pp. 63–68, IEEE/ACM, Washington, DC, USA, September 2018.
- [6] M. S. Hossain, G. Muhammad, and S. Umar, "Improving consumer satisfaction, in smart cities using edge computing and caching: a case study of: date fruits classification improving consumer satisfaction in smart cities, using edge computing and caching: a case study of date fruits classification," *Future Generation Computer Systems*, vol. 88, pp. 333–341, 2018.
- [7] U. Pal, R. K. Roy, and F. Kimura, "Multi-lingual city name recognition for Indian postal automation," in *Proceedings of the International Conference on Frontiers in Handwriting Recognition, Bari*, pp. 169–173, Bari, Italy, September 2012.
- [8] S. Thadchanamoorthy, N. D. Kodikara, H. L. Premaretne, U. Pal, and F. Kimura, "Tamil handwritten city name database development and recognition for postal automation," in *Proceedings of the 12th International Conference on Document Analysis and Recognition*, pp. 793–797, IEEE, Washington, DC, USA, August 2013.
- [9] S. Vajda, K. Roy, U. Pal, B. B. Chaudhuri, and A. Belaid, "Automation of Indian postal documents written in Bangla and English," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 8, pp. 1599–1632, 2009.
- [10] S. Sahoo, S. K. Nandi, S. Barua, S. Malakar, and R. Sarkar, "Handwritten Bangla city name recognition using the shape-context feature," in *Intelligent Engineering Informatics*, pp. 451–460, Springer, Singapore, 2018.
- [11] S. Y. Manchala, J. Kinthali, K. Kotha, K. S. Kumar, and J. Jayalaxmi, "Handwritten text recognition using deep learning with TensorFlow," *International Journal of Engineering Research & Technology*, vol. 9, no. 5, pp. 594–600, 2020.
- [12] S. Bhowmik, S. Malakar, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "Off-line Bangla handwritten word recognition: a holistic approach," *Neural Computing & Applications*, vol. 31, no. 10, pp. 5783–5798, 2019.
- [13] T. M. Wahbi and M. E. Musa, "Holistic approach for Arabic word recognition," *International Journal of Computer Applications Technology and Research*, vol. 5, no. 3, pp. 141–146, 2016.
- [14] N. Sharma, A. Sengupta, R. Sharma, U. Pal, and M. Blumenstein, "Pincode detection using deep CNN for postal automation," in *Proceedings of the International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pp. 1–6, IEEE, Christchurch, New Zealand, December 2017.
- [15] J. Dasgupta, K. Bhattacharya, and B. Chanda, "A holistic approach for Off-line handwritten cursive word recognition using directional feature based on Arnold transform," *Pattern Recognition Letters*, vol. 79, pp. 73–79, 2016.
- [16] U. Jindal, S. Gupta, V. Jain, and M. Paprzycki, "Offline handwritten gurmukhi character recognition system using deep learning," in *Advances in Bioinformatics, Multimedia, and Electronics Circuits and Signals*, pp. 121–133, Springer, Singapore, 2020.
- [17] S. Bansal, M. Kumar, and M.. Garg, "A New approach for Handwritten city name recognition," in *Proceedings of the International Conference on Advances in Engineering and Technology(ICAET)*, pp. 106–109, Roorkee, India, May 2014.
- [18] U. Pal, R. K. Roy, and F. Kimura, "Handwritten street name recognition for Indian postal automation," in *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, pp. 483–487, IEEE, Beijing, China, September 2011.
- [19] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT express*, vol. 6, no. 4, pp. 312–315, 2020.
- [20] Q. Abbas, M. E. Ibrahim, and M. A. Jaffar, "A comprehensive review of recent advances on deep vision systems," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 39–76, 2019.
- [21] Y. Wen, Y. Lu, and P. Shi, "Handwritten Bangla numeral recognition system and its application to postal automation," *Pattern Recognition*, vol. 40, no. 1, pp. 99–107, 2007.
- [22] K. Roy, S. Vajda, U. Pal, and B. B. Chaudhuri, "A system towards Indian postal automation," in *Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, pp. 580–585, IEEE, Kokubunji, Japan, October 2004.
- [23] S. Juneja, M. Gahlan, G. Dhiman, and S. Kautish, "Futuristic cyber-twin architecture for 6G technology to support internet of everything," *Scientific Programming*, vol. 20217 pages, 2021.
- [24] S. Sharma, S. Gupta, and N. Kumar, "Holistic approach employing different optimizers for the Recognition of District names using CNN model," *Annals of RSCB*, vol. 25, no. 3, pp. 3294–3306, 2021.
- [25] S. Juneja, S. Jain, A. Suneja et al., "Gender and age classification enabled blockchain security mechanism for assisting mobile application," *IETE Journal of Research*, pp. 1–13, 2021.