Handwritten Character Recognition using Convolutional Neural Network

Gitanjali V. Salve
Rajashri Shahu School of Engineering & Reearch, Narhe, Pune, MH, India
Savitribai Phule Pune University, Ganeshkhind Pune -411007

Abstract: Neural networks have made big strides in image classification. Convolutional neural networks (CNN) work successfully to run neural networks on direct images. Handwritten character recognition (HCR) is now a very powerful tool to detect traffic signals, translate language, and extract information from documents, etc. Although handwritten character recognition technology is in use in the industry, present accuracy is not outstanding, which compromises both performance and usability. Thus, the character recognition technologies in use are still not very reliable and need further improvement to be extensively deployed for serious and reliable tasks. On this account, characters of the English alphabet and digit recognition are performed by proposing a custom-tailored CNN model with two different datasets of handwritten images, i.e., Kaggle and MNIST, respectively, which are lightweight but achieve higher accuracies than state-of-the-art models. The best two models from the total of twelve designed are proposed by altering hyper-parameters to observe which models provide the best accuracy for which dataset. In addition, the classification reports (CRs) of these two proposed models are extensively investigated considering the performance matrices, such as precision, recall, specificity, and *F*1 score, which are obtained from the developed confusion matrix (CM). To simulate a practical scenario, the dataset is kept unbalanced and three more averages for the F measurement (micro, macro, and weighted) are calculated, which facilitates better understanding of the performances of the models. The highest accuracy of 99.642% is achieved for digit recognition, with the model using 'RMSprop', at a learning rate of 0.001, whereas the highest detection accuracy for alphabet recognition is 99.563%, which is obtained with the proposed model using 'ADAM' optimizer at a learning rate of 0.00001. The macro *F*1 and weighted *F*1 scores for the best two models are 0.998, 0.997:0.992, and 0.996, respectively, for digit and alphabet recognition.

Keywords: handwritten character recognition; English character recognition; convolutional neural networks (CNNs); deep learning in character recognition; digit recognition; English alphabet recognition

INTRODUCTION

Handwriting is the most typical and systematic way of recording facts and information. The handwriting of an individual is idiosyncratic and unique to individual people. The capability of software or a device to recognize and analyze human handwriting in any language is called a handwritten character recognition (HCR) system. Recognition can be performed from both online and offline handwriting. In recent years, applications of handwriting recognition are thriving, widely used in reading postal addresses, language translation, bank forms and check amounts, digital libraries, keyword spotting, and traffic sign detection.
Image acquisition, preprocessing, segmentation, feature extraction, and classification are the typical processes of an HCR system, as shown in Figure 1. The initial step is to receive an image form of handwritten characters, which is recognized as image acquisition that will proceed as an input to preprocessing. In preprocessing, distortions of the scanned images are removed and converted into binary images. Afterward, in the segmentation step, each character is divided into sub images. Then, it will extract every characteristic of the features from each image of the character. This stage is especially important for the last step of the HCR system, which is called classification [1]. Based on classification accuracy and different approaches to recognize the images, there are many classification methods, i.e., convolutional neural networks (CNNs), support vector machines (SVMs), recurrent neural networks (RNNs), deep belief networks, deep Boltzmann machines, and K-nearest neighbor (KNN) [2].

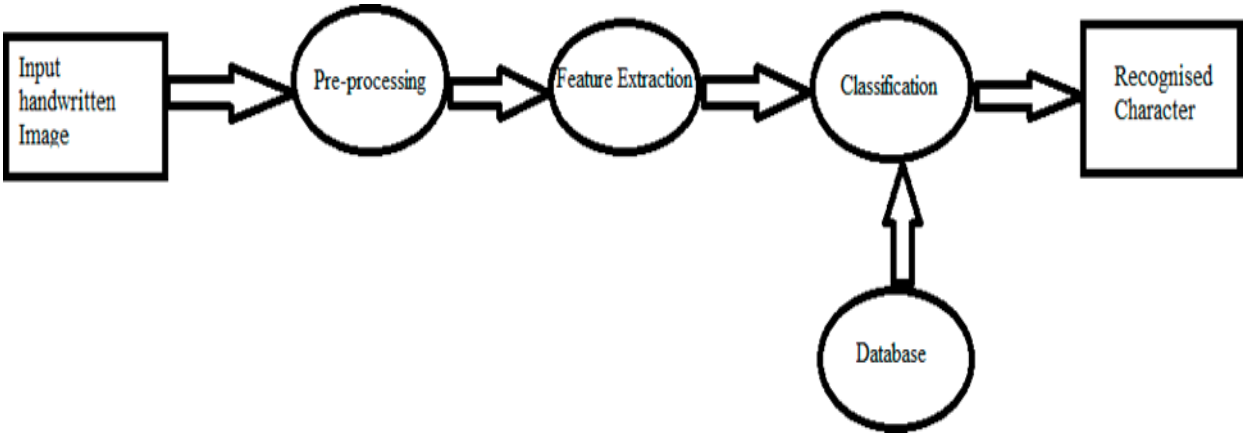

Figure 1. Representation of a common handwritten character recognition (HCR) system

A subclass of machine learning comprises neural networks (NNs), which are information-processing methods inspired by the biological process of the human brain. Figure 2 represents the basic neural network. The number of layers is indicated by deep learning in a neural network. Neurons, being the information-processing element, build the foundation of neural networks that draws parallels from the biological neural network. Weights associated with the connection links, bias, inputs, and outputs are the primary components of an NN. Every node is called a perceptron in a neural network (NN) [3]. Research is being conducted to obtain the best accuracy, but the accuracy using a CNN is not outstanding, which compromises the performance and usability for handwritten character recognition. Hence, the aim of this paper is to obtain the highest accuracy by introducing a handwritten character recognition (HCR) system using a CNN, which can automatically extract the important features from the images better than multilayer perceptron (MLP) [4–9].

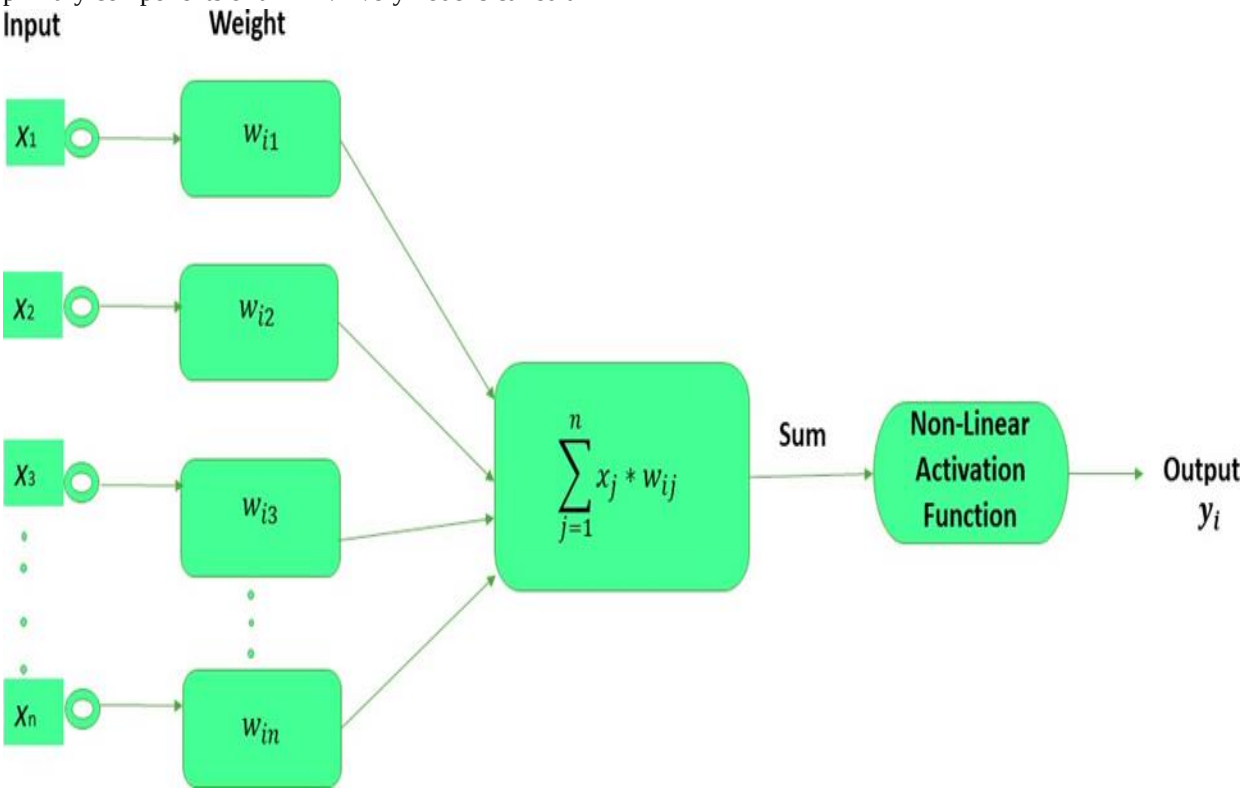

Figure 2. Representation of a basic neural network (NN).

CNNs were first employed in 1980 [10]. The conception of convolutional neural networks (CNNs) was motivated by the human brain. People can identify objects from their childhood because they have seen hundreds of pictures of those objects, which is why a child can guess an object that they have never seen before. CNNs work in a similar way. CNNs used for analyzing visual images are a variation of an MLP deep neural network that is fully connected. Fully connected means that each neuron in the layer is fully connected to all the neurons in the subsequent layer. Some of the renowned CNN architectures are AlexNet (8 layers), VGG (16, 19 layers), GoogLeNet (22 layers), and ResNet (152 layers) [11]. CNN models can provide an excellent recognition result because they do not need to collect prior knowledge of designer features. As for CNNs, they do not depend on the rotation of input images.

A CNN model has been broadly set for the HCR system, using the MNIST dataset. Such research has been carried out for several years. A few researchers have found the accuracy to be up to 99% for the recognition of handwritten digits [12]. An experiment was carried out using a combination of multiple CNN models for MNIST digits and had 99.73% accuracy [13]. Afterward, for the same MNIST dataset, the recognition accuracy was improved to 99.77%, when this experiment of the 7-net committee was extended to a 35-net committee [14]. Niu and Suen minimized the structural risk by integrating the SVM for the MNIST digit recognition and obtain the astonishing accuracy of 99.81% [15]. Chinese handwritten character recognition was investigated using a CNN [16]. Recently, Alvear-Sandoval et al. worked on deep neural networks (DNN) for MNIST and obtained a 0.19% error rate [17]. Nevertheless, after a vigilant investigation, it has been observed that the maximal recognition accuracy of the MNIST dataset can be attained by using only ensemble methods, as these aid in improving the classification accuracy. However, there are tradeoffs, i.e., high computational cost and increased testing complexity [18]. In this paper, a tailored CNN model is proposed which attains higher accuracy with light computational complexity.

Research on HCR technology has been going on for long time now and it is in use by the industry, but the accuracy is low, which compromises the usability and overall performance of the technology. Until now, the character recognition technologies in use are still not very dependable and need more development to be deployed broadly for unfailing applications. On this account, characters of the English alphabet and digit recognition are performed in this paper by proposing a custom-tailored CNN model with two different datasets of handwritten images, i.e., Kaggle and MNIST, respectively, which achieve higher accuracies. The important features of these proposed projects are as follows:

1. In the proposed CNN model, four 2D convolutional layers are kept the same and unchanged to obtain the maximum comparable recognition accuracy into two different datasets, Kaggle and MNIST, for handwritten letters and digits, respectively. This proves the versatility of our proposed model.
2. A custom-tailored, lightweight, high-accuracy CNN model (with four convolutional layers, three max-pooling layers, and two dense layers) is proposed by keeping in mind that it should not overfit. Thus, the computational complexity of our model is reduced.
3. Two different optimizers are used for each of the datasets, and three different learning rates (LRs) are used for each of the optimizers to evaluate the best models of the twelve models designed. This suitable selection will assist the research community in obtaining a deeper understanding of HCR.
4. To the best of the authors' knowledge, the novelty of this work is that no researchers to date have worked with the classification report in such detail with a tailored CNN model generalized for both handwritten English alphabet and digit recognition. Moreover, the proposed CNN model gives above 99% recognition accuracy both in compact MNIST digit datasets and in extensive Kaggle datasets for alphabets.
5. The distribution of the dataset is imbalanced. Hence, only the accuracy would be ineffectual in evaluating model performance, so advanced performances are analyzed to a great extent with a classification report for the best two proposed models for the Kaggle and MNIST datasets, respectively. Classification reports indicate the *F*1 score for each of the 10 classes for digits (0–9) and each of the 26 classes for alphabet (A–Z). In our case of multiclass classification, we

examined averaging methods for the *F*1 score, resulting in different average scores, i.e., micro, macro, and weighted average, which is another novelty of this proposed project.

The rest of the paper is organized as follows: Section 2 describes the review of the literature and related works in the handwritten character recognition research arena; Sections 3 and 4 present datasets and proposed CNN model architecture, respectively; Section 5 discusses the result analysis and provides a comparative analysis; and Section 6 describes the conclusion and suggestions for future directions.

REVIEW OF LITERATURE AND RELATED WORKS

Many new techniques have been introduced in research papers to classify handwrit- ten characters and numerals or digits. Shallow networks have already shown promis- ing results for handwriting recognition [19–26]. Hinton et al. investigated deep belief networks (DBN), which have three layers along with a grasping algorithm, and recorded an accuracy of 98.75% for the MNIST dataset [27]. Pham et al. improved the perfor- mance of recurrent neural networks (RNNs), reducing the word error rate (WER) and character error rate (CER) by employing a regularization method of dropout to recognize unconstrained handwriting [28].

The convolutional neural network (CNN) delivered a vast change as it delivers a state-of-the-art performance in HCR accuracy [29–33]. In 2003, for visual document analysis, a common CNN architecture was introduced by Simard et al., which loosened the training of complex methods of neural networks [34]. Wang et al. used multilayer CNNs for end-to-end text recognition on benchmark datasets, e.g., street view text and ICDAR 2003, and accomplished brilliant results [35].

Recently, for scene text recognition, Shi et al. introduced a new approach, the conventional recurrent neural network (CRNN), integrating both the deep CNN (DCNN) and recurrent neural network (RNN), and announced its superiority to traditional methods of character recognition [36]. For semantic segmentation, Badrinarayanan et al. proposed a deep convolutional network architecture where the max-pooling layer was used to obtain good performance; the authors also compared their model with current techniques. The segmentation architecture known as SegNet consists of a pixel-wise classification layer, an encoder network, and a decoder network [37,38]. In offline handwritten character recognition, CNN has shown outstanding performance for different regional and inter- national languages. Researchers have conducted studies on Chinese handwritten text recognition [39–41]; Arabic language [42]; handwritten Urdu text recognition [43,44]; hand- written Tamil character recognition [45]; Telugu character recognition [46]; and handwritten character recognition on Indic scripts [47]. Gupta et al. used features extracted from a CNN in their model and recognized the informative local regions in [48] from recent character images, accomplishing a recognition accuracy of 95.96% by applying a novel multi-objective optimization framework for HCR which comprises handwritten Bangla numerals, handwritten Devanagari characters, and handwritten English numerals. High performance of the CROHME dataset was observed in the work of Nguyen et al. [49]. The author employed a multiscale CNN for clustering handwritten mathematical expression (HME) and concluded by identifying that their model can be improved by training the CNN with a combination of global, attentive, and max-pooling layers.

Recognition of word location in historical books, for example on Gutenberg's Bible pages, is wisely addressed in the work of Ziran et al. [50] by developing an R-CNN-based deep learning framework. Ptucha et al. introduced an intelligent character recognition (ICR) system, logically using a conventional neural network [51].

DATASET

The Kaggle alphabet dataset was sourced from the National Institute of Standards and Technology (NIST), NMIST, and other google images [60]. Kaggle English handwritten alphabets of 26 classes are shaped by training with over 297,000 sets of examples and a test set, which is made up of over 74,490 examples. The total distribution of Kaggle letters is illustrated in Figure 4. Each letter is of uniform size and by computing the center of mass of the pixels, each binary image of a handwritten letter is centered into a 28 × 28 image.

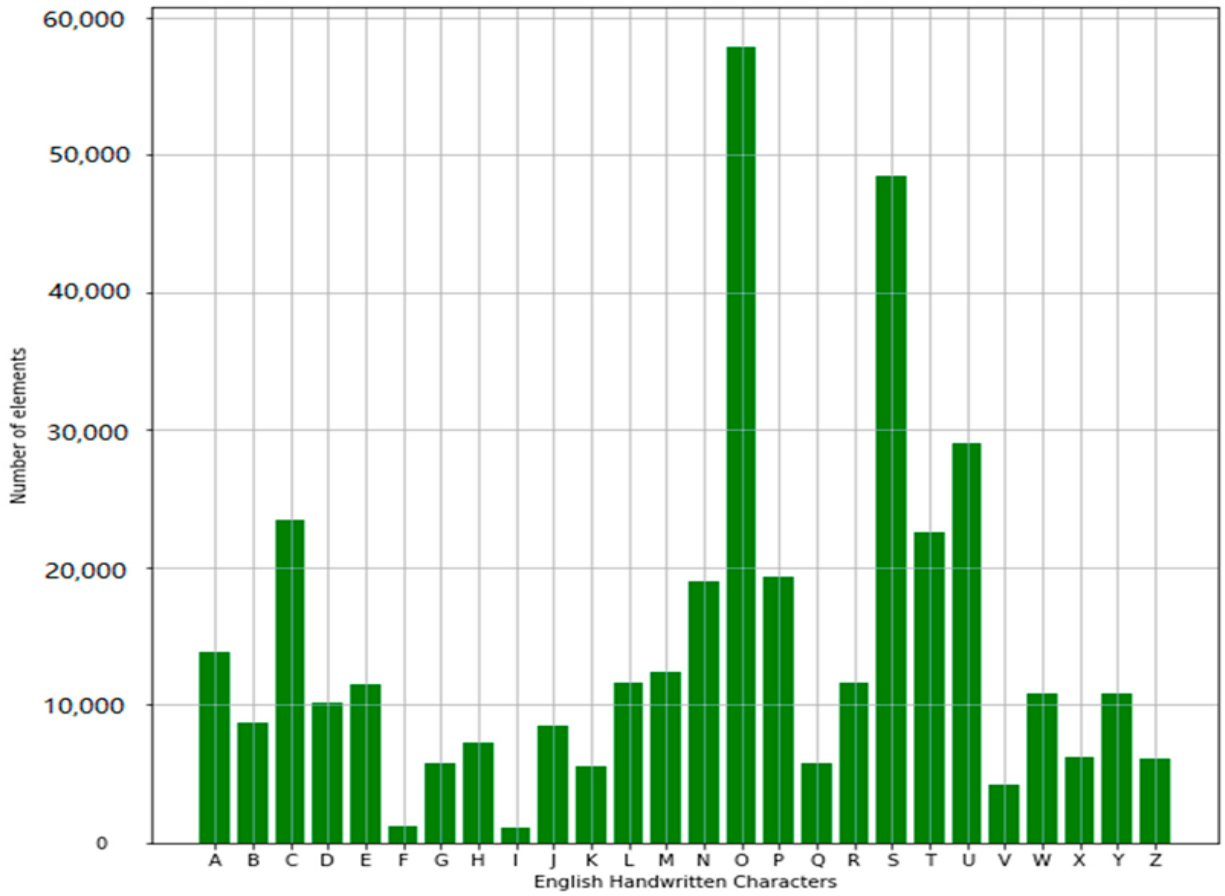

Figure 4. Total distribution of Kaggle letters (A–Z).

PROPOSED CONVOLUTIONAL NEURAL NETWORK

Of all the deep learning models in image classifications, CNN has become very popular due to its high performance in recognizing image patterns. This has opened up various application opportunities in our daily life and industries which include medical image classification, traffic monitoring, autonomous object recognition, facial recognition, and much more.

CNNs are sparse, feed-forward neural networks [61]. The idea of an artificial neuron was first conceptualized in 1943. Hubel and Wisel first found that, for detecting lights in the receptive fields, visual cortex cells have a major role, which greatly inspired building models such as neocognitron. This model is considered to be the base and predecessor of CNN. CNN is formed of artificial neurons which have a self-optimization property, learning like brain neurons. Due to this self-optimizing property, it can extract and classify the features extracted from images more precisely than any other algorithm. Moreover, it needs very limited preprocessing of the input data, while yielding highly accurate and precise results. CNNs are vastly used in object detection and image classification, including medical imaging. In image classification, each pixel is considered a feature for the neural network. CNN tries to understand and differentiate among the images depending on these features. Conventionally, first few convolutional layers capture very low-level features, such as the edges, gradient orientation, or color. However, with the increased number of convolutional layers, it starts extracting high-level features. Due to the higher dimensionality and convolution, the parameters of the network increase exponentially. This makes the CNN computationally heavy. However, with the development of computational technology and GPU, these jobs have become much more efficient. Moreover, the development of the CNN algorithms has also

prompted the ability to reduce dimensionality by considering small patches at a time which reduces the computational burden without losing the important features.

Handwritten character recognition (HCR) with deep learning and CNN was one of the earliest endeavors of researchers in the field. However, with increased modeling efficacy and the availability of a huge dataset, current models can perform significantly better than the models of ten years ago. However, one of the challenges of the current models is generalization. The model that performs excellently with one dataset may perform poorly with a different one. Thus, it is important to develop a robust model which can perform with the same level of accuracy across different datasets, which would give the model versatility. Thus, a CNN model is designed which is computationally proficient because of its optimized number of CNN layers, while performing with high accuracy across multisource massive datasets.

Owing to the lower resolution of the handwritten character images, the images which were fed to the input layers were sized 28 × 28 pixels. The input layer feeds the images to the convolutional layers, where the features are convolved. The model has only four convolutional layers, which makes it lightweight and computationally efficient. The first layer is a 2D convolutional layer with a 3 × 3 kernel size and rectified linear unit (ReLU)-activation function. ReLU is one of the most widely used activation functions in deep learning algorithms. ReLU is computationally effective because the neurons are not activated altogether like the other activation functions, e.g., tanh [62]. ReLU is a piecewise linear function which is also continuous and differentiable at all points except for 0. Besides providing simplicity and empirical simplicity, it also has reduced likelihood of vanishing gradient. Because of the abovementioned benefits, and as per the suggestion of the literature that ReLUs tend to converge early, it was chosen for our model. The idea behind ReLU is simple, it returns positive values input directly to the output, whereas the negative values are returned as 0, as depicted in Figure 5.

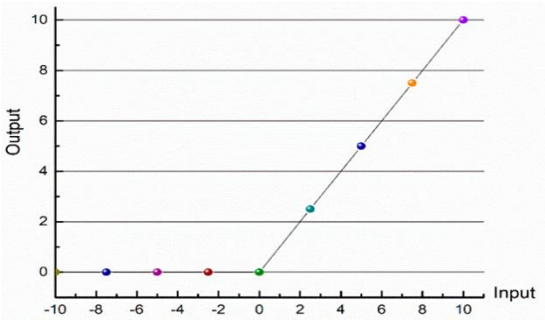

Figure 5. Rectified linear unit (ReLU) function.

The subsequent three layers are the 2D convolutional layers, which are accompanied by one max-pooling layer and a ReLU-activation function. Max pooling is a sample-based discretization process which is used to downsize our input images. It pools the maximum value from each patch of each feature map, thus helping to reduce the dimensionality of the network. Moreover, it reduces the number of parameters by discarding insignificant ones, which decreases the computational burden as well as helping to avoid overfitting. Thus, a 2 × 2 max-pooling layer is integrated in each of the convolutional layers except for the first one. The output of the fourth convolutional layer is fed to the flattening layer to convert the input to a 1D string, which is then fed to the fully connected layer, i.e., the dense layer.

In the fully connected layer, as the name suggests, all the neurons are linked to the activation units of the following layer. In the proposed model, there are two fully connected layers where all the neurons of the first layer are connected to the activation unit of the second fully connected layer. In the second fully connected layer, all the inputs are passed to the Softmax activation function, which categorizes the features into multiclass as needed. Finally, the determined class of any input image is declared in the output. The proposed model is illustrated in Figure 6 and the resultant parameters of each layer are tabulated in Table 1.

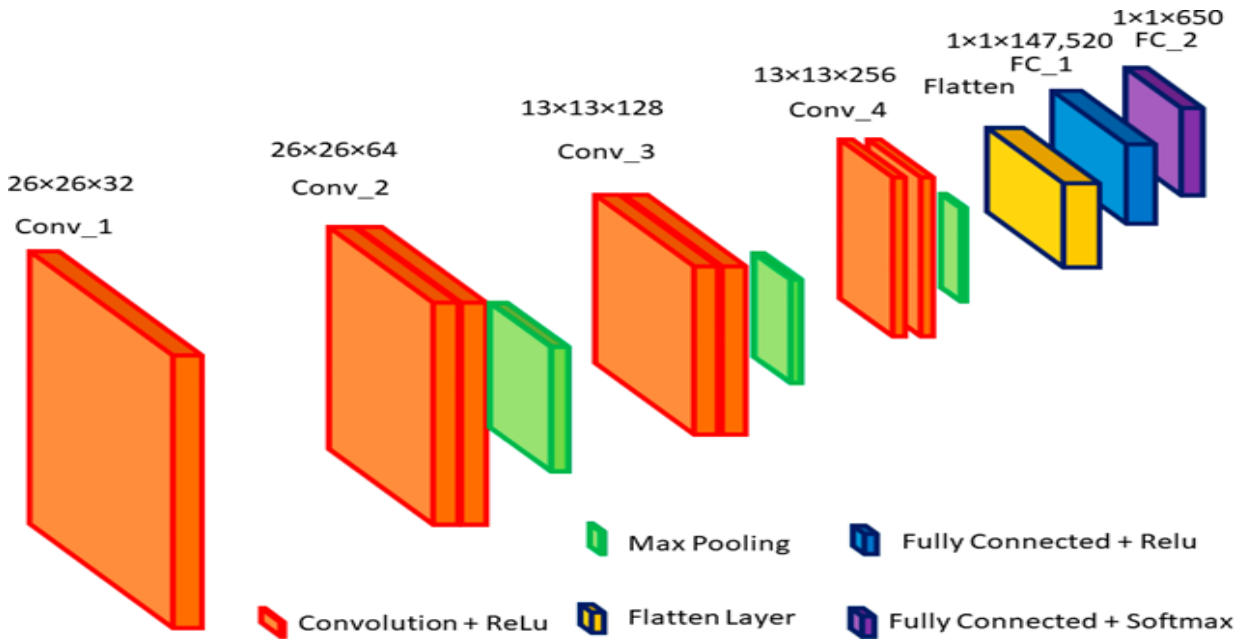

Figure 6. Proposed CNN model for character recognition

Layer (Type)	Output Shape	Param #
conv_1 (Conv 2D)	(None, 26, 26, 32)	320
conv_2 (Conv 2D)	(None, 26, 26, 64)	18,496
max_pooling2D_18 (MaxPooling2D)	(None, 13, 13, 64)	0
conv_3 (Conv 2D)	(None, 13, 13, 128)	73,856
max_pooling2D_19 (MaxPooling2D)	(None, 6, 6, 128)	0
conv_4 (Conv 2D)	(None, 6, 6, 256)	295,168
max_pooling2D_20 (MaxPooling2D)	(None, 3, 3, 256)	0
flatten (Flatten)	(None, 2304)	0
FC_1 (Dense)	(None, 64)	147,520
FC_2 (Dense)	(None, 10)	650
Total Params #		536,010
Trainable Params #		536,010
Non-Trainable Params #		0

Table 1. Details of the proposed model.

For generalization, the same proposed model is used to classify both the English alphabets and digits. The only difference is the number of output classes defined in the last fully connected layer, which is the 'fully connected + Softmax' layer, as depicted by Figure 6, and the FC_2 layer, as presented by Table 1. The number of classes is 10 for digit recognition as depicted by the table, and the number of classes is 26 for alphabet recognition. Moreover, for extensive comparative analysis, we also analyzed how the proposed model performs with different optimizers, 'ADAM' and 'RMSprop', which also include the variation of the learning rates (LRs). This analysis helps in understanding how the model performance might vary with the change of optimizers and variation of learning rates which are discussed in detail in Section 5—Results and Analysis.

In order to avoid the difficulties posed by the problem of latency in data process- ing, this project utilizes Colab-pro by Google, which has a 2.20 GHz Intel Xeon Proces- sor, 128 GB RAM, and Tesla P100 16 GB GPU. The model was designed and tested in Colab-pro, keeping in mind the factor of easy reproducibility by the research community, as Colab-pro has built-in support for GPU-enabled TensorFlow and the necessary support for CUDA acceleration.

RESULTS AND ANALYSIS

We used two datasets for the handwritten recognition process: the Kaggle dataset for our English letter (A–Z) and MNIST for our numeric characters (0–9). Two optimizers were used for each of the datasets, 'ADAM' and 'RMSprop', as well as three different learning rates (LRs) of 0.001, 0.0001, and 0.00001 for each of the optimizers. This gives us six CNN models for each of the datasets and twelve models overall. To avoid confusion and repetition, we named our models. The models were named as follows for the Kaggle dataset: with a learning rate of 0.001, the model under the 'ADAM' optimizer is K1 and the one under the 'RMSprop' is K2; with a learning rate of 0.0001, the model under the 'ADAM' optimizer is K3 and the one under the 'RMSprop' is K4; with a learning rate of 0.00001, the model under the 'ADAM' optimizer is K5 and the one under the 'RMSprop' is K6. The models were named similarly for the MNIST dataset from model M1 to model M6. Our results indicated that we obtained the best result under the 'ADAM' optimizer with a learning rate of 0.00001 under the Kaggle dataset (model K5), and under 'RMSprop' with a learning rate of 0.001 for the MNIST dataset (model M2). We then calculated the *F*1 score (micro, macro, and weighted average) and obtained confusion matrices and two classification reports for the two models that give us the best accuracy for the each datasets. Figure 7 simplifies the selection of the best models for each dataset.

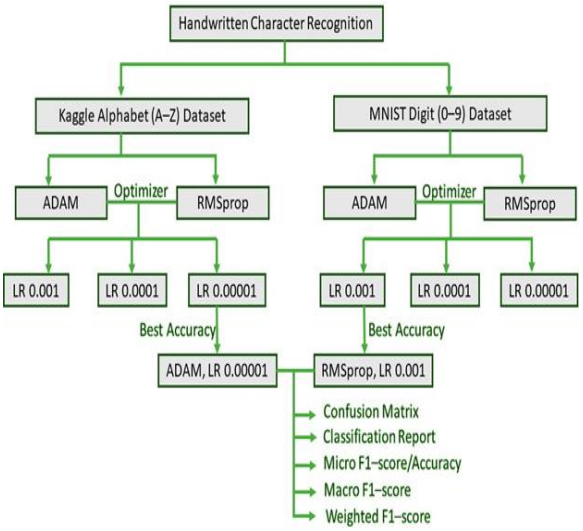

Figure 7. Best model selection from the Kaggle and MNIST datasets.

For the alphabet dataset, the overall accuracies using the 'ADAM' optimizer in the proposed CNN model for handwritten English alphabet recognition were 99.516%, 99.511%, and 99.563% for LR 0.001, LR 0.0001, and LR 0.00001, respectively. The same model using 'RMSprop' achieved the accuracy of 99.292%, 99.108%, and 99.191%, respectively, by LR 0.001, LR 0.0001, and LR 0.00001. These results clearly show that, in terms of accuracy, the model using the 'ADAM' optimizer with LR 0.00001, named as model K5, performs better than the other proposed models. It is clear that all the proposed six models for character recognition achieved above 99.00% overall accuracy.

For the digit dataset, the overall accuracies using 'RMSprop' for handwritten digit recognition were 99.642%, 99.452%, and 98.142% for LR 0.001, LR 0.0001, and LR 0.00001, respectively. The same model using the 'ADAM' optimizer achieved accuracies of 99.571%, 99.309%, and 98.142% for LR 0.001, LR 0.0001, and LR 0.00001, respectively. Figures 8 and 9 depict validation accuracies and Figures 10 and 11 show the validation losses of all the twelve models with the Kaggle and MNIST dataset, respectively. It is clear that overall accuracy decreases with the decrease in learning rate (LR). This confirms that the model using 'RMSprop' with LR 0.001, named as model M2, outperformed the other proposed models in terms of accuracy. From Figures 9 and 11, it can be clearly observed that no overfitting happens for the digit recognition or for alphabet recognition; overfitting occurs when 'RMSprop' is used, which is depicted in Figures 8d–f and 10d–f. Overfitting occurs when the model performs fine on the training data but does not perform exactly in the testing set. Here, the model learns the unnecessary information within the dataset as it trains for a long time on the training data.

The performance evaluation of the models is more obvious and explicit from the matrices of specificity, recall, precision, *F*1 score, and support. The possible outcomes obtained by the confusion matrix (CM) calculate the performance of these matrices. This CM has four different outcomes: total false positive (TFP), total false negative (TFN), total true positive (TTP), and total true negative (TTN). The CM sets up nicely to compute the per-class values of recall, precision, specificity, and *F*1 score for each of the datasets.

Let us consider the scenario where we want the model to detect the letter 'A'. For simplification, let us also assume that each of the 26 letters in the alphabet (A–Z) has 100 images for each of the letters, totaling 2600 images altogether. If we assume that the model accurately identifies the images of the letter 'A' in 97 out of 100 images, then we say that the accuracy of the model is 97%. Thus, we can also conclude that the total number of true positives (TTPs) is 97. Under the same assumptions as above, if the letter 'O' is incorrectly identified as 'A', then this would tell us that the number of total false positives (TFPs) in this case would be 1. If the letter 'A' has been misidentified as 'O' three times in the model, then the total number of false negatives (TFNs) for this model is 3. The rest of the 2499 images of the 2600 images are then considered as the total true negative (TTN). Figures 12 and 13 show the confusion matrices for the best two models (model K5 for letter recognition and model M2 for digit recognition) established in terms of overall performance that were trained and validated with the Kaggle and MNIST datasets, respectively.

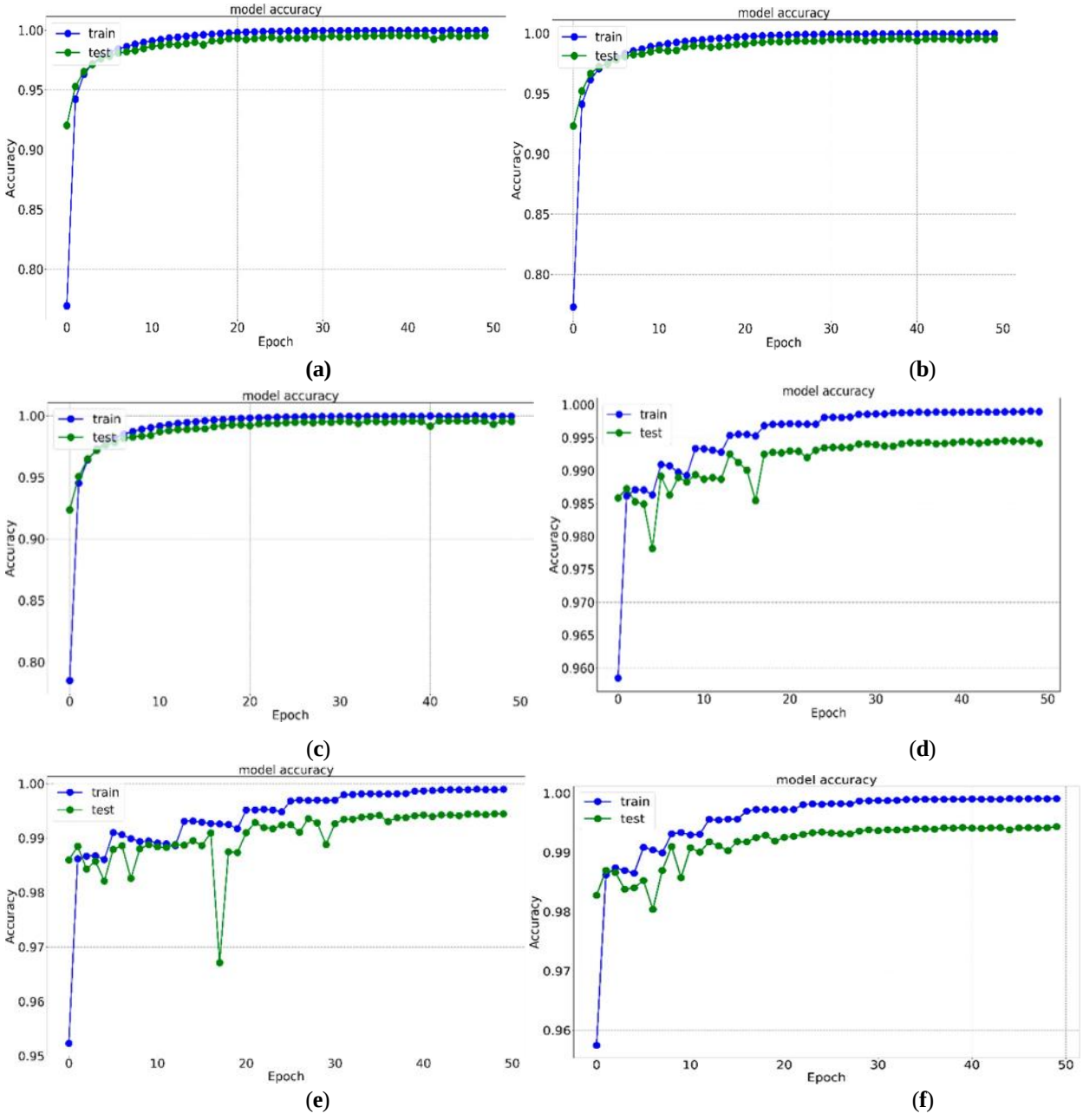

Figure 8. Validation accuracy of the six models for English alphabet recognition. (**a**) Optimizer—'ADAM'; learning rate—0.001. (**b**) Optimizer—'ADAM'; learning rate—0.0001. (**c**) Optimizer—'ADAM'; learning rate—0.00001. (**d**) Optimizer—'RMSprop'; learning rate—0.001. (**e**) Optimizer—'RMSprop'; learning rate—0.0001. (**f**) Optimizer—'RMSprop'; learning rate—0.00001.

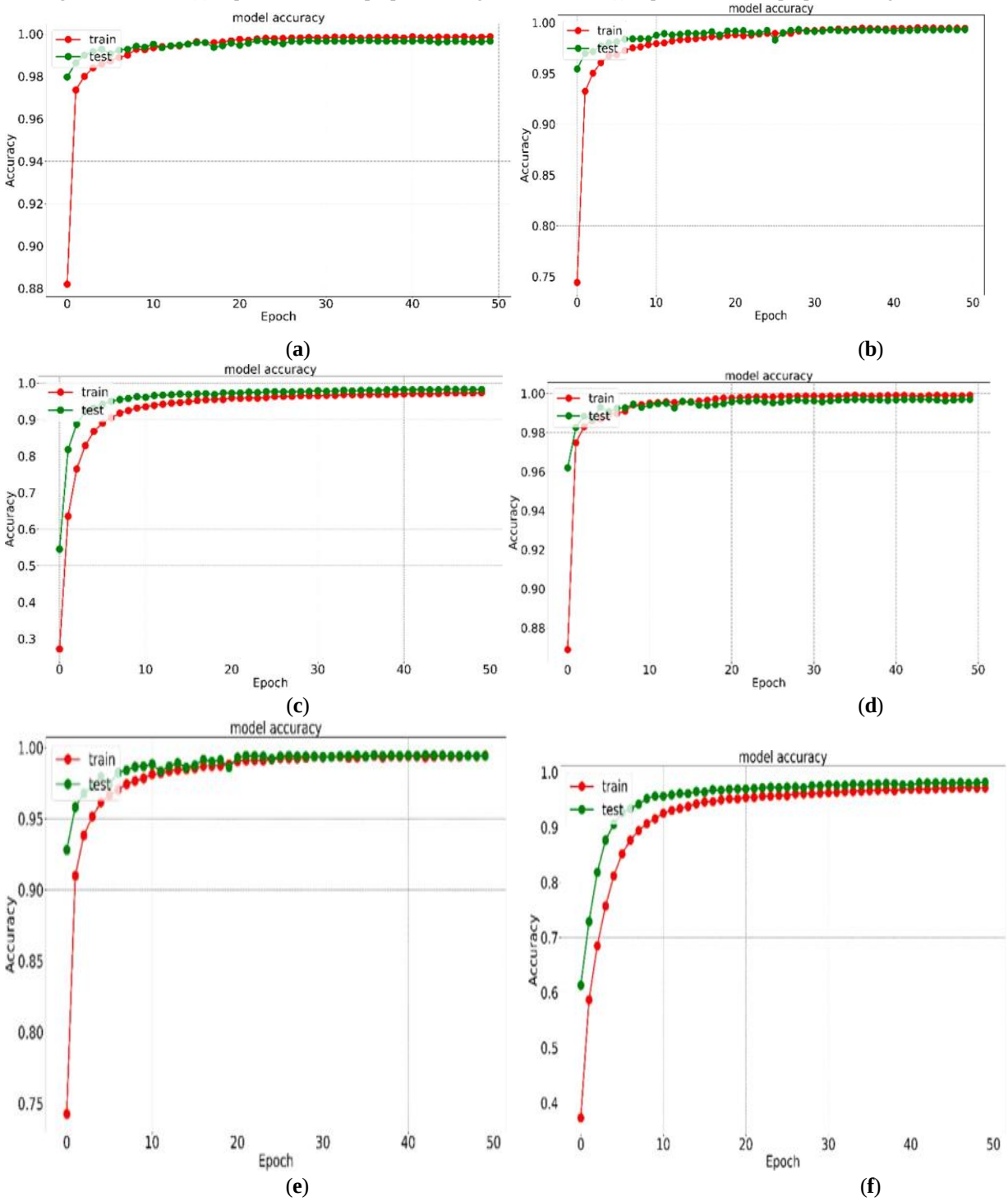

Figure 9. Validation accuracy of the six models for digit (0–9) recognition. (**a**) Optimizer—'ADAM'; learning rate—0.001. (**b**) Optimizer—'ADAM'; learning rate—0.0001. (**c**) Optimizer—'ADAM'; learning rate—0.00001. (**d**) Optimizer—'RMSprop'; learning rate—0.001. (**e**) Optimizer—'RMSprop'; learning rate—0.0001. (**f**) Optimizer—'RMSprop'; learning rate—0.00001.

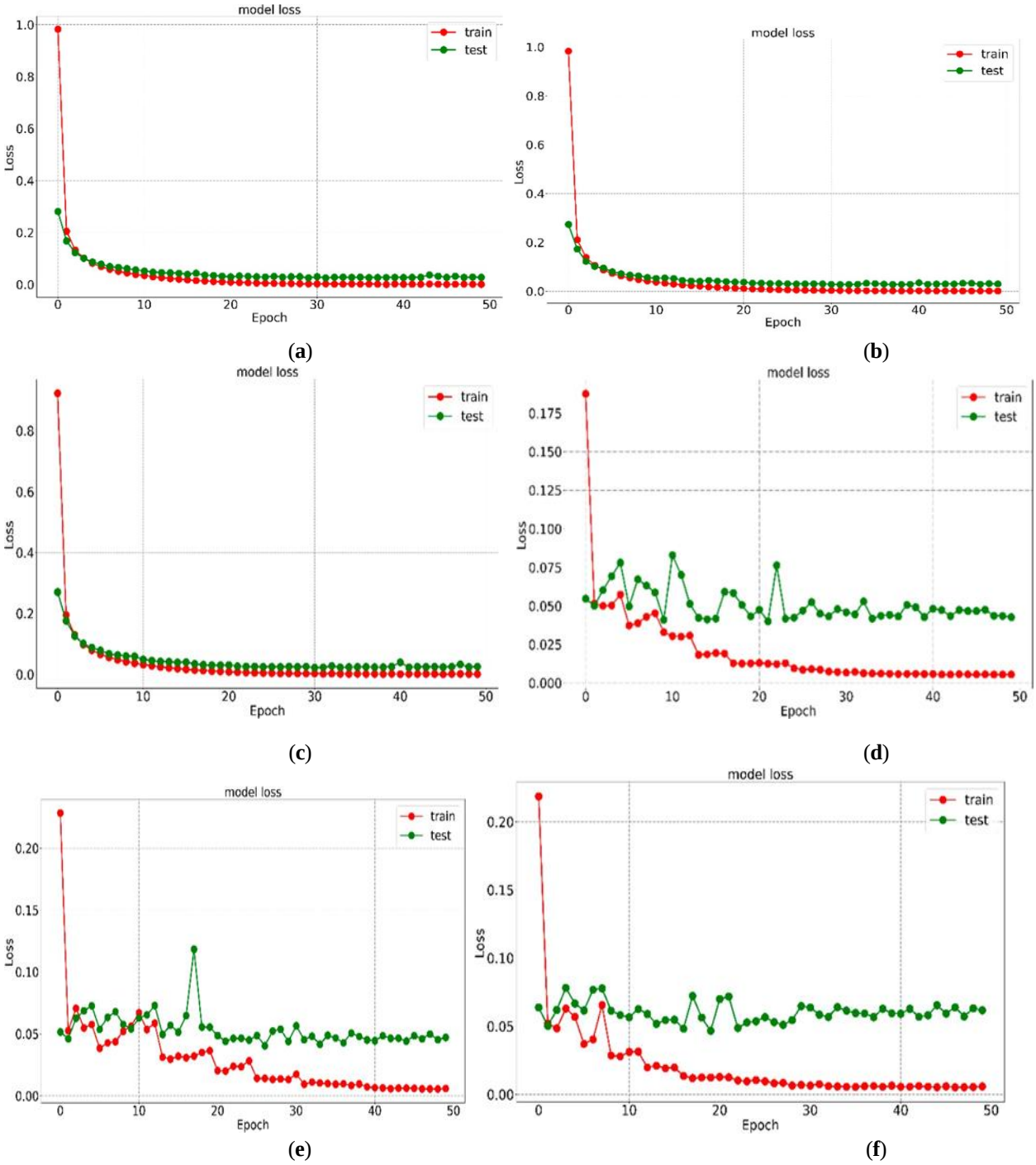

Figure 10. Validation loss of the six models for English alphabet recognition. (**a**) Optimizer—'ADAM'; learning rate—0.001. (**b**) Optimizer—'ADAM'; learning rate—0.0001. (**c**) Optimizer—'ADAM'; learning rate—0.00001. (**d**) Optimizer—'RMSprop'; learning rate—0.001. (**e**) Optimizer—'RMSprop'; learning rate—0.0001. (**f**) Optimizer—'RMSprop'; learning rate—0.00001.

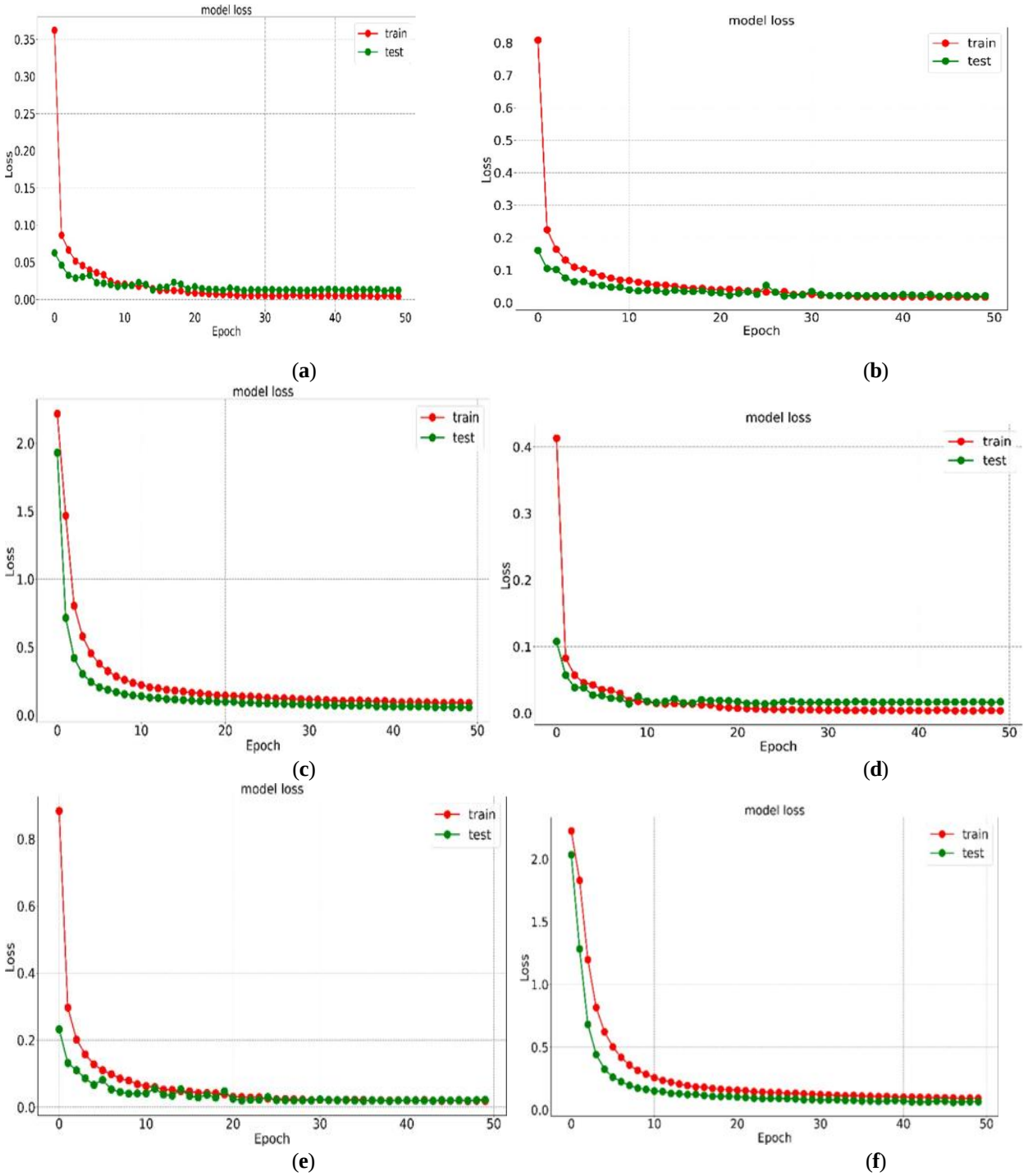

Figure 11. Validation loss of the proposed six models for digit (0–9) recognition. (**a**) Optimizer—'ADAM'; learning rate—0.001. (**b**) Optimizer—'ADAM'; learning rate—0.0001. (**c**) Optimizer—'ADAM'; learning rate—0.00001. (**d**) Optimizer—'RMSprop'; learning rate—0.001. (**e**) Optimizer—'RMSprop'; learning rate—0.0001. (**f**) Optimizer—'RMSprop'; learning rate—0.00001.

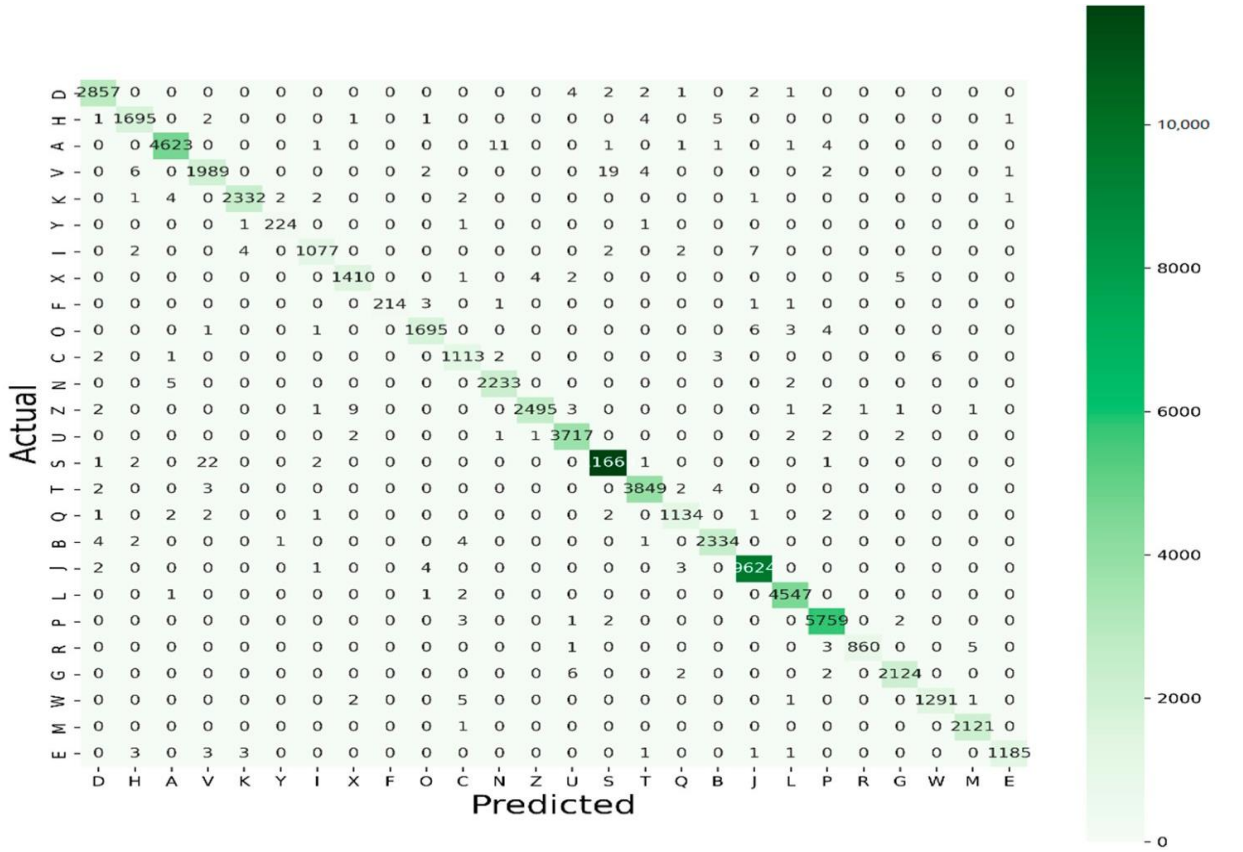

Figure 12. Confusion matrix of model K5 for A–Z recognition.

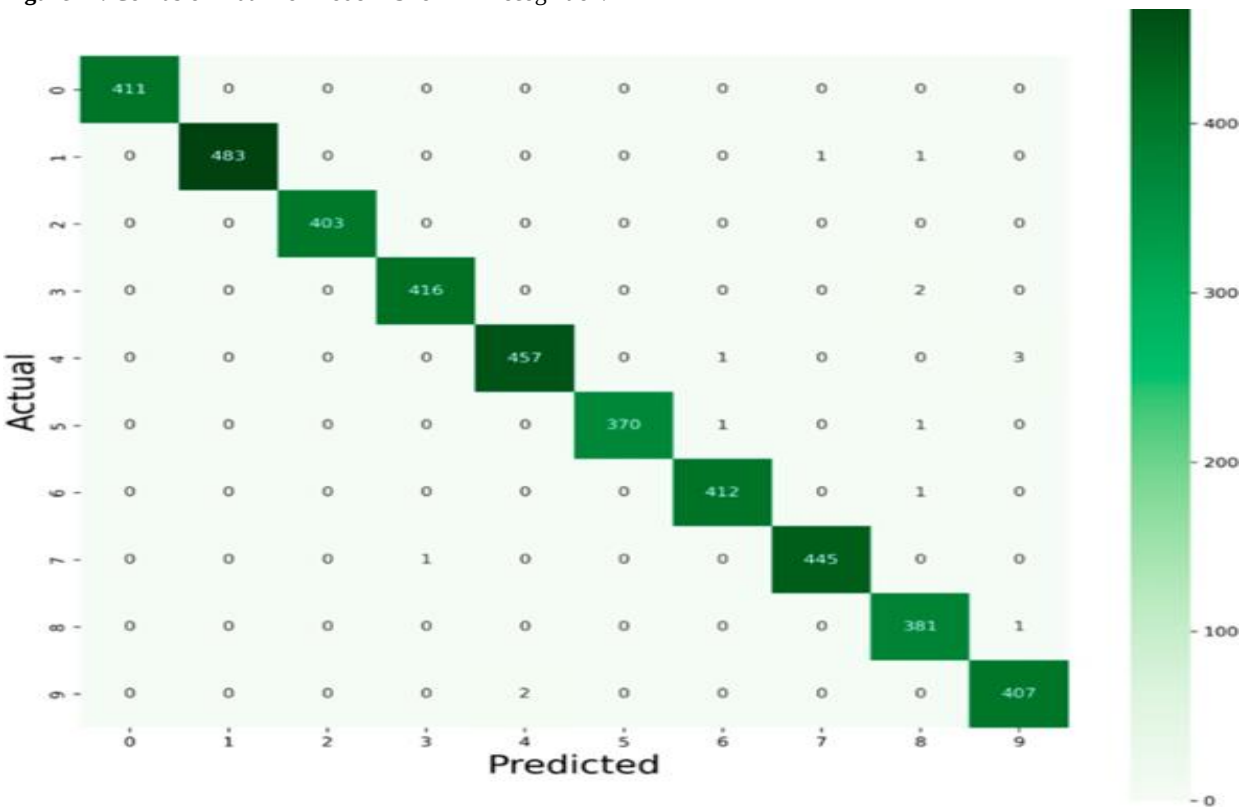

Figure 13. Confusion matrix of model M2 for 0–9 recognition.

CONCLUSIONS

In modern days, applications of handwritten character recognition (HRC) systems are flourishing. In this paper, to address HCR systems with multiclass classification, a CNN-based model is proposed that achieved exceptionally good results with this multiclass classification. The CNN models were trained with the MNIST digit dataset, which is shaped with 60,000 training and 10,000 testing images. They were also trained with the substantially larger Kaggle alphabet dataset, which comprises over 297,000 training images and a test set which is shaped on testing over 74,490 images. For the Kaggle dataset, the overall accuracies using the 'ADAM' optimizer were 99.516%, 99.511%, and 99.563% for learning rate (LR) 0.001, LR 0.0001, and LR 0.00001, respectively. Meanwhile, the same model using 'RMSprop' achieved accuracies of 99.292%, 99.108%, and 99.191%, respectively, by LR 0.001, LR 0.0001, and LR 0.00001. For the MNIST dataset, the overall accuracies using 'RMSprop' were 99.642%, 99.452%, and 98.142% for LR 0.001, LR 0.0001, and LR 0.00001, respectively. Meanwhile, the same model using the 'ADAM' optimizer achieved accuracies of 99.571%, 99.309%, and 98.142% with LR 0.001, LR 0.0001, and LR 0.00001, respectively. It can be easily understood that, for alphabet recognition, accuracy decreases with the increase in learning rate (LR); contrarily, overall accuracy is proportionately related to LR for digit recognition. In addition, precision, recall, specificity, and *F*1 score were measured from confusion matrices. Of all the discussed twelve models, the model using the 'ADAM' optimizer with LR 0.00001 obtained a recall value of 99.56%, and the model with LR 0.001 with the 'RMSprop' optimizer obtained the recall value of 99.64%; therefore, these two models excel other models for the Kaggle and MNIST datasets, respectively. As the distribution of the datasets is imbalanced, only the accuracy would be ineffective in evaluating the models; therefore, classification reports (CR) indicating the *F*1 score for every 10 classes for digits (0–9) and every 26 classes for alphabet (A–Z) were included for the predictions of the best two proposed models. From the CR, we achieved micro, macro, and weighted *F*1 scores of 0.996 and 0.995, 0.998 and 0.992, and 0.997 and 0.996 for the MNIST and Kaggle datasets, respectively. Furthermore, the obtained results of best two models presented here were compared with the results of other noticeable works in this arena. Considering future work, we intend to include several feature extraction methods by applying a similar framework to that proposed here to more complex languages, such as Korean, Chinese, Finnish, and Japanese.

REFERENCES

[1] Priya, A.; Mishra, S.; Raj, S.; Mandal, S.; Datta, S. Online and offline character recognition: A survey. In Proceedings of the International Conference on Communication and Signal Processing, (ICCSP), Melmaruvathur, Tamilnadu, India, 6–8 April 2016; pp. 967–970.

[2] Gunawan, T.S.; Noor, A.F.R.M.; Kartiwi, M. Development of english handwritten recognition using deep neural network. *Indones. J. Electr. Eng. Comput. Sci.* 2018.

[3] Vinh, T.Q.; Duy, L.H.; Nhan, N.T. Vietnamese handwritten character recognition using convolutional neural network. *IAES Int. J. Artif. Intell.* 2020.

[4] Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893.

[5] Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* 2004.