School of Engineering and Digital Sciences

Electrical & Computers Engineering Department

ELCE 455 Machine Learning with Python

Group Project Report

Sign Language Classification using MNIST dataset

Daniyar Zhakyp

201774605

Almat Duisembayev

201756513

ABSTRACT

Understanding one another through the use of a common language has always been a valuable ability of adaptable humans. However, due to the minority status of certain groups, non-verbal languages like sign languages were not on the list of languages that humans aimed to learn, thus creating a social distance between the minorities and other people. This paper attempts to design a robust and accurate sign language classification algorithm using convolutional neural networks (CNNs) and classical machine learning models like k-nearest neighbors (k-NN), logistic regression (LogReg), and random forest (RF). The MNIST American Sign Language (ASL) dataset used in this paper has been divided into training (60%), validation (20%), and test (20%) sets. The data augmentation has been applied only to the training set. Among the four test models, CNN has achieved a 100% score on accuracy, precision, recall, f1, and ROC AUC metrics.

INTRODUCTION

Sign language is a form of communication for people who are deaf, hard of hearing, or dumb. According to the World Federation of the Deaf, there are over 300 sign languages used by 70 million deaf people worldwide [1]. Despite such a large number of sign languages, some sign systems might be complex to learn or hard to remember without daily use, which creates a strong social barrier and communication gap between ordinary people and people with disabilities. The latest knowledge in artificial intelligence (AI) and machine learning (ML) allows scientists and researchers to develop real-time sign and pattern recognition systems that are able to distinguish hand gestures of a particular letter or word. This work is focused on designing an offline classification model based on deep learning (DL) methods like convolutional neural networks (CNN) and ML methods like k-nearest neighbors (k-NN) to predict letters shown in images using American Sign Language (ASL) in the MNIST dataset. As the image classification problem is a major part of different deep learning and computer vision (CV) tasks, previous researchers have been deploying various CNN architectures on many distinct datasets [2]. A CNN model with two convolutional layers has received two types of inputs – depth and color images of signs – and concatenated their features to achieve an accuracy of 80.34% [3]. In other work, researchers have used transfer learning to build a CNN on a pre-trained Inception V3 model and achieved 92% accuracy on a validation set of digit hand signs [4]. Another proposed CNN model with stochastic pooling layers has been trained on a self-designed dataset consisting of 200 signs from five subjects and performed with 92.88% accuracy on test data [5]. As one of the preprocessing steps, multi-view (3D) data augmentation and inference fusion have been used on 50% of the training data of the ASL images, which yielded 92.7% accuracy on the remaining 50% corresponding to the test data [6].

In this project, we have used four augmentation techniques – random zooming, rotating, vertical and horizontal shifting - on the large MNIST dataset as one of the sources of innovation. Random search using a validation set has also been used to identify the optimal number of convolutional filters, dense units, and dropout rate. Lastly, three statistical ML models like k-NN, logistic regression (LogReg), and random forest (RF) have been trained on the same dataset, and their performance has been evaluated and compared to that of CNN.

METHODS

Dataset description

The dataset contains labels from 0 to 25, which are mapped one-to-one with alphabetic letters A-Z. For example, "0" represents the letter "A," "1" represents the letter "B," and so on for other labels and letters. However, exceptions are "J" = "9" and "Z" = "25," because the signs of these letters are dynamic and need motion gestures. Therefore, the dataset contains 24 labels (0–24). Fig. 1 represents the training and test data that have been used in this project. There are 27455 images (80%) in the training data and 7172 (20%) images in the test data. These images are grayscale, with values ranging from 0 to 255 and a size of 28x28 pixels.

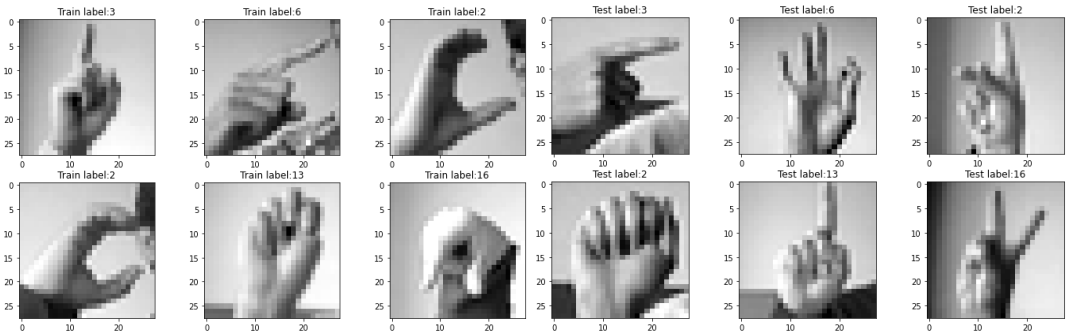

Fig. 1. The images of training and test data

The graphs with the number of images per class for training and test data are presented in Fig. 2. It is clearly seen that the training set of data is balanced; therefore, the training set does not require any technique to balance it.

Fig. 2. The number of images per label of training and test data

Data preprocessing

The first step in data preprocessing is scaling the training and test sets by dividing them by 255; therefore, images are rescaled and values are in the range of 0–1, which makes the training and prediction processes faster. The next step is splitting training data into training and validation data (75% and 25%, respectively). After this process, the number of training data images decreased; therefore, it is considered to be data augmentation, to artificially expand the size of training data, and to improve the accuracy of model prediction. The data augmentation of the input image is shown in Fig. 3. Data augmentation contains zooming, rotation, vertical shift, and horizontal shift of input images.

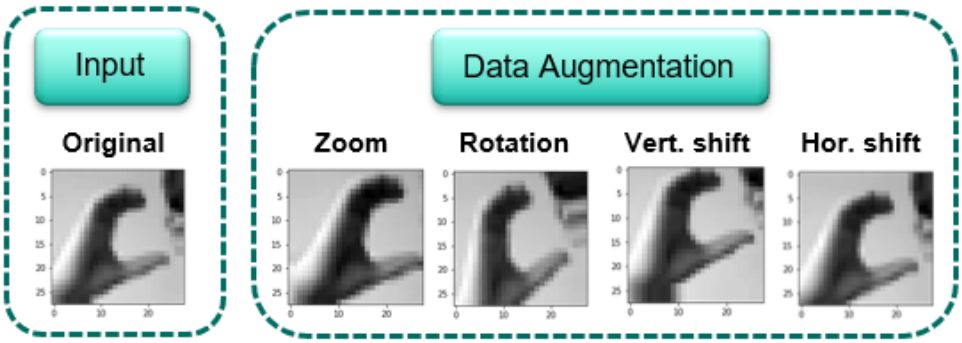

Fig. 3. The data augmentation of original image

CNN Model Architecture

The CNN architecture that we have built is shown in Fig. 4. The network consists of three convolutional layers alternated with three max pooling layers; at the end, there are two fully connected (dense) layers, where the last maps the values (i.e., probabilities) to 24 different classes (i.e., letters).

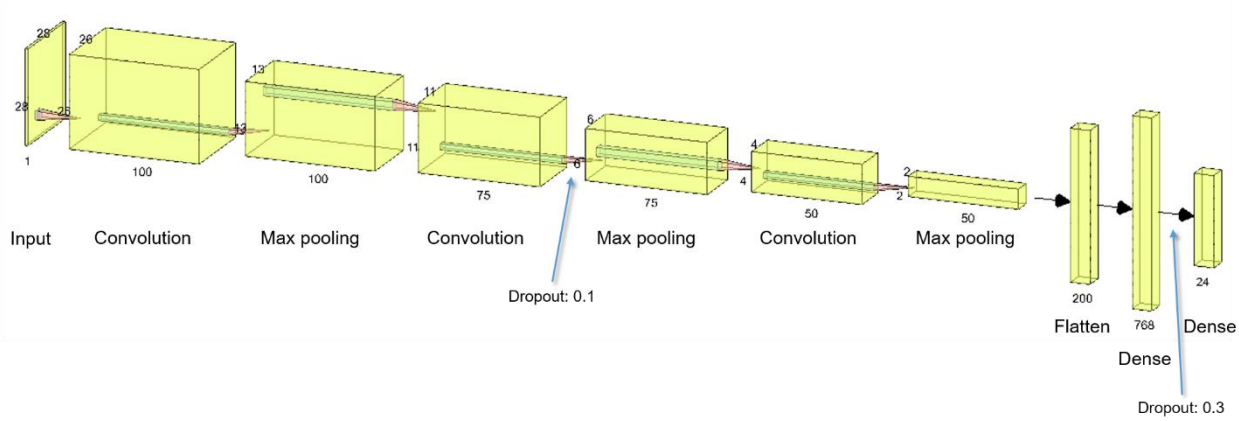

Fig. 4. The CNN architecture with three convolutional and max pooling layers, two dense units, and two dropouts.

The random search optimization technique has been used to find optimal parameters for our network in five epochs based on validation accuracy. As a result, the training data has been divided 75% for the training set and 25% for the validation set in advance – before the data augmentation process. The parameters included in the search space of the random search algorithm are the number of filters in each convolutional layer, the

number of dense units, dropout rates, and learning rates. Table I shows the optimal values of the parameters picked by random search. We have specified the range within which the optimization algorithm should search: there, the number of filters in the next convolutional layer as well as the dropout rate are meant to decrease. The algorithm has chosen the values for the first, second, and third layers to be 100, 75, and 50, respectively; the dropout rate has increased from 0.1 to 0.3; the number of dense units has been set to 768; and the learning rate has been chosen to be 0.001.

Table I. The optimal values of various parameters picked by the random search algorithm within the specified range between min and max numbers

	Min value	Max value	Step	Optimal value
conv_1_filter	75	200	25	100
conv_2_filter	50	125	25	75
conv_3_filter	25	75	25	50
drop_1_rate	0.1	0.5	–	0.1
drop_2_rate	0.1	0.3	–	0.3
dense_1_units	128	1024	32	768
learning_rate	10^{-3}	10^{-2}	–	10^{-2}

CNN Model Training

The step-wise approach for training the CNN model is shown in Fig. 5. Having found the optimal parameter values, the training and validation sets have been merged to train the model. The designed model with the optimal parameter values retrieved from random search has been set up for training with a batch of 128 images and 20 epochs. We have created a callback that reduces the learning rate by a factor of 0.5 once it stops improving after three consecutive epochs. The model has been compiled using the Adam optimizer, categorical cross-entropy as the loss function, and accuracy for the metrics.

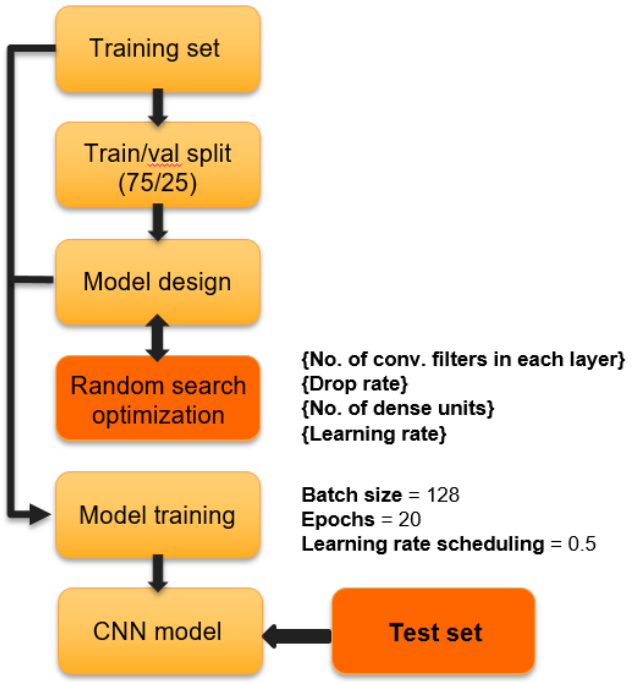

Fig. 5. The flow diagram consisting of splitting the set intro training and validation sets, designing a model, random searching for optimal parameter values, training the model on the entire training set, and evaluating the performance with the test set.

Machine Learning (ML) Models

To observe how less or more effective ML models manage the image classification task than CNNs, k-NN, LogReg, and RF classifiers have been introduced in this project. The data augmentation has not been applied for the ML model design. The manual tuning based on a single hyperparameter has been performed for each model; for k-NN, it is the number of neighbors ranging from 3 to 11 in a step of 2; for LogReg, it is the C hyperparameter ranging from 0.01 to 10 in a step of 10; and for RF, it is the number of estimators ranging from 50 to 200 in a step of 50.

RESULTS AND EVALUATION

Table II shows the results of manual tuning of hyperparameters for three ML models. The training set has been divided into training and validation sets with the same random states as in the CNN model optimization. As a result, the k-NN classifier with the number of neighbors of 3, the logistic regression model with C = 10, and the random forest classifier with the number of estimators of 150 have shown the best accuracy on the training and validation sets; therefore, these three optimal models have been chosen for the final comparison with CNN's performance.

Table II. Training and validation accuracies for three different ML models based on the corresponding changes of their hyperparameters

	k-NN (no. of neighbors)					**LogReg (value of C)**				**RF (no. of trees)**			
Parameters	**3**	5	7	9	11	0.01	0.1	1	**10**	50	100	**150**	200
Training accuracy (%)	**100**	99.8	99.7	99.4	99.2	81.6	96.1	99.9	**100**	100	100	**100**	100
Validation accuracy (%)	**99.8**	99.7	99.3	99.0	98.7	80.7	95.2	99.8	**99.9**	99.5	99.7	**99.8**	99.8

Table III summarizes the performance of all four models chosen to classify images from the MNIST dataset. The CNN model, which has a total of 276,009 parameters, performed the best on the test set, with a score of 1 on every metric. The second best performance based on the test accuracy has been shown by the RF classifier, with 82.0%. The k-NN model has performed slightly worse on the test set than RF, with 80.4% accuracy. Lastly, logistic regression has demonstrated the worst performance with 66.0% test accuracy.

Table III. Comparison of the performance of CNN, k-NN, LogReg, and RF based on the set of classification metrics

	CNN	**k-NN**	**LogReg**	**RF**
	Total params: 276,009	**No. of neighbors** = 3; **Distance weight:** Equal	**Penalty:** L_2; **Solver:** Liblinear; **C** = 10	**No. of estimators** = 150; **Split criterion:** Gini
Training accuracy	**1.000**	1.000	1.000	1.000
Test accuracy	**1.000**	0.804	0.660	0.820
Precision	**1.000**	0.810	0.664	0.817
Recall	**1.000**	0.798	0.648	0.814
F1 score	**1.000**	0.793	0.641	0.806
ROC AUC	**1.000**	0.923	0.935	0.984

Fig. 6 shows the change in both training/test loss and training/test accuracy of the best model (i.e., CNN) on each epoch. Fig. 7 depicts its confusion matrix. The accuracy/loss graphs demonstrate the convergence of the model's performance to the ideal values of 1 for accuracy and 0 for loss, despite the presence of several “negative” spikes. The confusion matrix demonstrates that all samples are on its diagonal, justifying that all the samples are true positives (TP) and have been correctly classified.

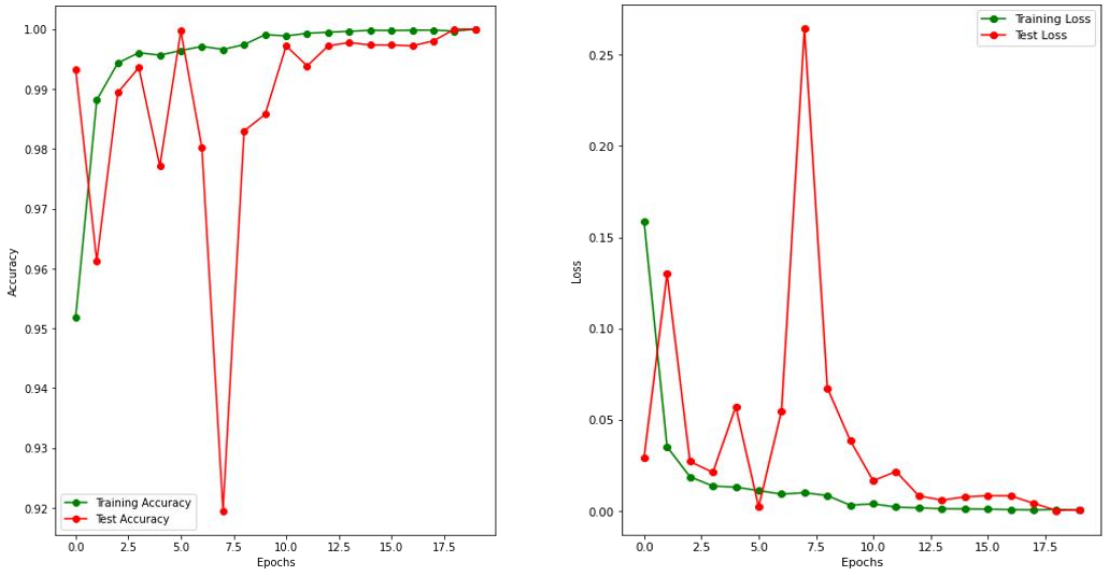

Fig. 6. The training and test accuracy (on the left) and the training and test loss (on the right) of the CNN model on each epoch.

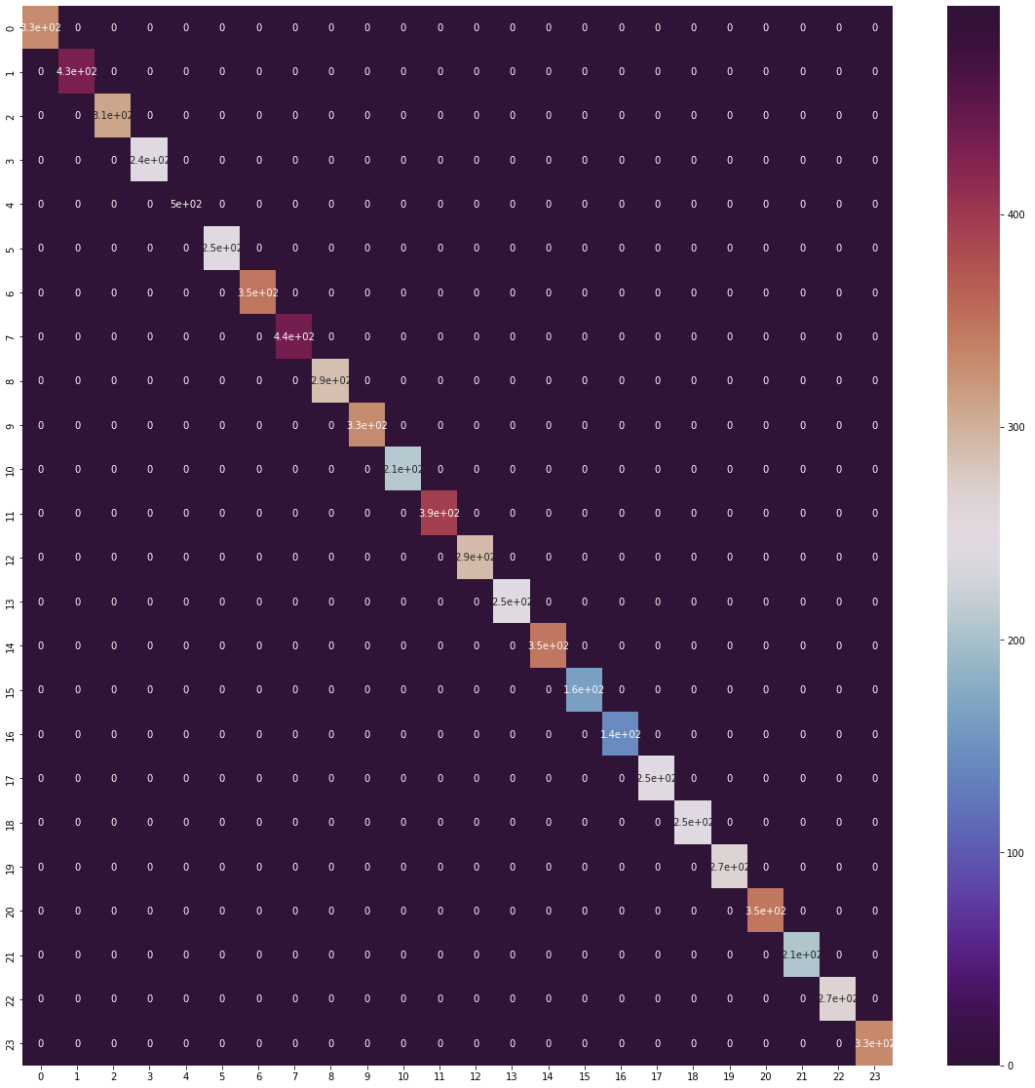

Fig. 7. The confusion matrix of the CNN model.

DISCUSSION

The designed CNN model has demonstrated the behavior of an "ideal" classifier, which predicted all the classes correctly. The reason for that, in the first place, might be the size of the MNIST dataset, which is quite formidable and has been even enlarged via data augmentation. Secondly, the model parameters have been optimized based on the validation accuracy, which has shown a score of 1. Lastly, the dataset has been fairly split into the training and test sets, with a considerate amount for each class. The MNIST dataset can be considered a benchmark dataset for deep learning models; however, feeding it as an array to machine learning models is not as effective. RF and k-NN classifiers have shown moderate overfitting of around 20% on the test set, whereas the LogReg model has performed inferiorly with a 34% accuracy drop on the holdout set.

CONCLUSION

In this project, we developed and compared machine learning models that predict real-time sign and pattern recognition and can distinguish hand gestures of American Sign Language letters. The American Sign Language MNIST dataset has been used to build models. The dataset was preprocessed, divided into three sets (train, test, and validation), augmented, and rescaled. Three classical machine learning models were used to train

and predict the preprocessed dataset: Random Forest, K-Nearest Neighbors, and Logistic Regression. Moreover, a deep learning model called Convolutional Neural Networks (CNN) has been used; among all of these models, CNN is the best model with 100% prediction accuracy on test data.

REFERENCES

[1] Disabled World, “Deaf communication: Sign language and assistive hearing devices - document list,” *Disabled World*, 07-Apr-2022. [Online]. Available: https://www.disabled-world.com/disability/types/hearing/communication-2/.

[2] R. Rastgoo, K. Kiani, and S. Escalera, “Sign language recognition: A deep survey,” *Expert Systems with Applications*, vol. 164, p. 113794, 2021.

[3] S. Ameen and S. Vadera, “A convolutional neural network to classify American Sign Language fingerspelling from depth and colour images,” *Expert Systems*, vol. 34, no. 3, 2017.

[4] M. B. Hossain, A. Adhikary, and S. J. Soheli, “Sign language digit recognition using different convolutional neural network model,” Asian Journal of Research in Computer Science, pp. 16–24, 2020.

[5] G. A. Rao, K. Syamala, P. V. Kishore, and A. S. Sastry, “Deep Convolutional Neural Networks for Sign Language recognition,” 2018 Conference on Signal Processing And Communication Engineering Systems (SPACES), 2018.

[6] W. Tao, M. C. Leu, and Z. Yin, “American sign language alphabet recognition using convolutional neural networks with Multiview Augmentation and inference fusion,” Engineering Applications of Artificial Intelligence, vol. 76, pp. 202–213, 2018.