Hand Gesture Recognition using CNN and DNN Algorithms

Ritika Giridhar

ABSTRACT

As the one of the oldest methods of communication amongst humans, gestures are a form of non-verbal communication that employ various body parts, mostly particularly the hands and face. Hand gestures are often a useful tool for adding meaning or emphasis to speech. Computer recognition of hand gestures may provide a more natural human-computer interface, allowing people to point, interact more naturally in the metaverse, or rotate a CAD model by rotating their hands. This paper provides two methods to train and classify different hand gestures, from a dataset of more than 2,000 images, using attention-based convolution neural networks (CNN) and deep neural networks (DNN). Training the data using an attention-based CNN model gave an accuracy of 87.12%, while training it with a DNN model gave an accuracy of 76.49%. Covered issues include choice of the paper subject, background and literature review, method, implementation, results and further discussion.

Keywords—CNN; DNN; hand gesture recognition

Abbreviations—CNN (convolutional neural network); DNN (deep neural network); AI (artificial intelligence)

I. INTRODUCTION

With the advancement of human civilization, the difficulty of interpersonal communication, not only in terms of language, but also in terms of communication between common people and hearing-impaired people, is gradually ending. If the development of sign language was the first step, then the development of hand recognition systems using computer vision is the second step. Several works have been carried out worldwide using AI for different sign languages [17]. Research papers based on hand gestures have adopted many different techniques, including those based on instrumented sensor technology and computer vision. In other words, the hand sign can be classified under many headings, such as posture and gesture, as well as dynamic and static, or a hybrid of the two [15]. Interactive computer games would be enhanced if the computer could understand the players' hand gestures. Gesture recognition may even be useful to control household appliances [20].

A dataset[1] with more than 2,000 images was classified under 8 different classes—open palm, closed fist, semi-open fist, semi-open palm, finger circles, single finger bend, multi finger bend and finger symbols. Various computer vision algorithms have employed colour and depth cameras for hand gesture recognition, but robust classification of gestures from different subjects performed under widely varying lighting conditions, background, orientations, and hand coverings is still challenging [13]. For this reason, images were also taken while wearing gloves (rubber and woollen) worn on the hand in order to test if the algorithm would produce different results or a lesser accuracy because of this additional barrier.

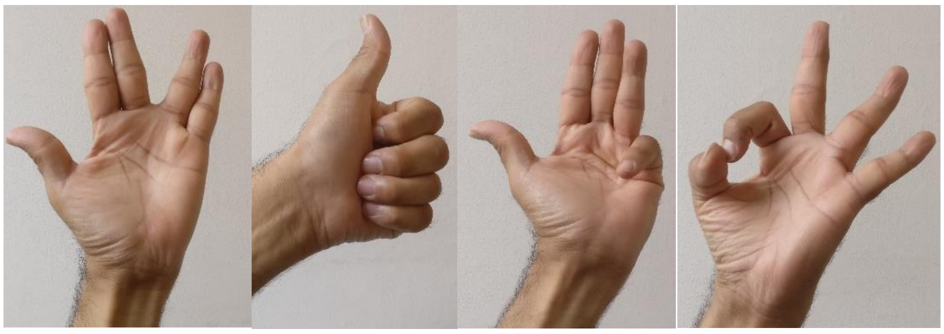


Figure 1. Different hand gestures present in the dataset.

[1] Dataset can be found at https://www.kaggle.com/datasets/ritikagiridhar/2000-hand-gestures

This project was chosen as a learning opportunity to figure out how convolution and deep neural network algorithms work, and how to implement them in specific cases. The project is divided into two main parts—one is the code[2] and the other is the report.

II. BACKGROUND AND LITERATURE REVIEW

A. Convolution Neural Networks (CNN)

The field of machine learning has taken a historic twist in recent times, with the rise of the Artificial Neural Network (ANN). These biologically inspired computational models are able to far exceed the performance of previous forms of AI in common machine learning tasks. One of the most impressive forms of ANN architecture is that of the Convolutional Neural Network (CNN). CNNs are primarily used to solve difficult image-driven pattern recognition tasks and, with their precise yet simple architecture (shown in figure 2 [10, 12]), offer a simplified method of getting started with ANNs [14]. A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

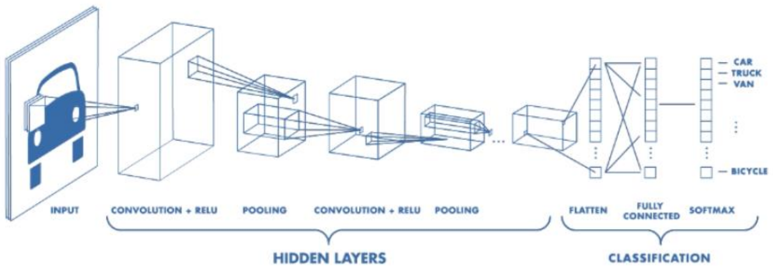


Figure 2. Architecture of a CNN

CNNs take their name from a mathematical linear operation between matrices called convolution. Convolution provides a way of "multiplying together" two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality [4], and can be performed using equation 1-

$$\boldsymbol{(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\tau'}$$

$$(f * g)(t) = \text{functions that are bring convoluted}$$
$$t = \text{real number variable of functions f and g}$$
$$g(\tau) = \text{convolution of the first f(t)}$$
$$\tau' = \text{first derivative of g}(\tau)\text{ function}$$

Equation 1. The convolution function

CNNs have excellent performance accuracies in machine learning problems, especially in those applications that deal with image data, such as the largest image classification data set (Image Net), computer vision, and in natural language processing (NLP), producing results of high significance [1]. By using a CNN, one can essentially "enable sight to computers".

B. Attention-based Convolution Neural Networks

An attention-based CNN is an "attention" operation that, for every decoder output step, produces a distinct vector representing all encoder hidden states but giving different weights to different encoder hidden state [19]. This can be seen in figure 3 [2]-

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

$$p(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

It should be noted that unlike the existing encoder–decoder approach, here the probability is conditioned on a distinct context vector c_i for each target word y_i

Figure 3. The attention operation

[2] Code can be found at https://github.com/trashsock/hand-gesture-recognition

The distinct context vector for an output step is a sum-product of attention weights and all input hidden states. The attention weights for every single output will be different and therefore the sum of the weighted hidden vectors is distinct for each output step, as shown in figure 4 [2].

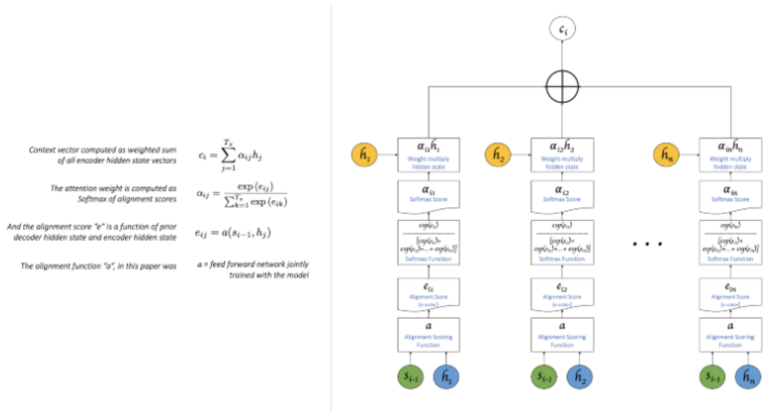

Context vector computed as weighted sum of all encoder hidden state vectors
$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$
The attention weight is computed as Softmax of alignment scores
$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$
And the alignment score "e" is a function of prior decoder hidden state and encoder hidden state
$e_{ij} = a(s_{i-1}, h_j)$
The alignment function "a", in this paper was
a = feed forward network jointly trained with the model
c_i
$\alpha_{i1} h_1$
Weight multiply hidden state
α_{i1}
Softmax Score
Softmax Function
e_{i1}
Alignment Score
a
Alignment Scoring Function
s_{i-1}
h_1
h_2
h_n

Figure 4. The weights and outputs for an attention operation

The spirit of "attention" is more about the ability to attend to various inputs for every output step and is less about other aspects like alignment function used, nature of the recurrent neural network (RNN) involved etc. While this solution seems to have addressed the problem of single context vector, it has made the model really big. There are a lot of computations involved when one tries to prepare a separate context vector for every output step.

C. Deep Neural Networks (DNN)

The term Deep Learning or Deep Neural Network (DNN) refers to ANNs with multiple layers, as shown in figure 5 [6].

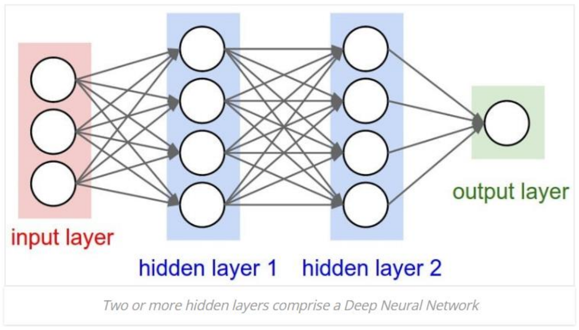


Figure 5. Different layers in a DNN

Over the last few decades, DNNs have been considered to be one of the most powerful tools, and have become very popular as they are able to handle huge amounts of data. The interest in having deeper hidden layers has recently begun to surpass classical methods performance in different fields, especially in pattern recognition [1]. DNNs excel at finding hierarchical representations that solve complex tasks over large datasets, and have become a promising solution to inject AI in our daily lives, from self-driving cars to smartphones, games, drones, etc. In most cases, DNNs were accelerated by server equipped with numerous computing engines, e.g., GPU. However, the recent technology advances require energy-efficient acceleration of DNNs as the modern applications moved down to mobile computing nodes [9].

III. METHOD

A. Using an attention-based CNN

First, the dataset was loaded onto Kaggle's Python environment, and the different classes were listed, along with the number of images in each class. To offset the misbalanced data, functions *trim()* and *balance()* were used. Test samples were created and shown using *showSampleImg()*.

The CNN "EfficientNetB5" was defined and used to train the data. EfficientNets are a new baseline network which achieve much better accuracy and efficiency than previous CNNs [18]. EfficientNetB5 returns a Keras image classification model, optionally loaded with weights pre-trained on ImageNet. In simpler words, EffieicntNetB5 is an attention-based CNN that produces an output sequence given an input sequence which are, in general, of different lengths.

Finally, the results, accuracy, F1-score and confusion matrix were printed.

B. Using a DNN

The deep learning model was built using Keras, which is a powerful and easy-to-use free open-source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows the user to define and train neural network models in just a few lines of code [3].

The dataset was loaded and the Keras model was defined as *Sequential()*. The Sequential model is a linear stack of layers. Sequential groups a linear stack of layers into a tf.keras.Model. Sequential provides training and inference features on this model [7, 8]. A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. The model was then compiled, fit onto the dataset, and then evaluated.

Finally, the results, accuracy, F1-score and confusion matrix were printed.

IV. IMPLEMENTATION

A. Dataset

Creating and using a good dataset is a way of ensuring high trainability and accuracy of the data. For this reason, a dataset of more than 2,000 images, called “Hand gestures dataset”, was created by taking pictures of multiple participants making different hand gestures, and classifying them under 8 different classes. The dataset was uploaded to Kaggle, due to its large size, and it has already been downloaded over 10 times. A detailed description of the creation of the dataset, and its classes, can be found in table 1.

Class	**Originally Calculated Number of Images (per hand for 1 participant)**	**Final Number of Images (both hands for 15 participants)**
Open palm	5	184
Closed fist	7	212
Semi-open fist	15	361
Semi-open palm	12	340
Finger circles	15	233
Single finger bend	15	239
Multi finger bend	12	69
Finger symbols	36	529
TOTAL	**111**	**2,146**

Table 1. Different classes and number of images in each class in the ‘2000+ Hand Gestures’ dataset

Due to lack of dexterity amongst some of the participants, as some of were of older age, the original number of 111*2*15 = 3,330 images could not be taken. Two special cases of participants wearing gloves (one made of rubber and the other made of wool) was also taken into consideration, to see if the AI could detect the gestures with some layer over the bare hand. In the end, 2,146 images were taken from 15 participants, 6 male and 9 female, aged 9 to 82. The file containing the compressed images is 20.4MB in size, and each image has dimensions of 207x207 pixels.

B. Kaggle

Kaggle is an online community of data scientists and machine learning enthusiasts and experts. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges [20]. The dataset was uploaded to Kaggle because of two reasons-

- It was easy to store 20.4MB of data in one place, and
- Kaggle has its own built-in Python environment

This made calling and using the dataset much easier. It should be noted that if the code is run on an environment outside Kaggle, the kaggle.json file must be downloaded from the GitHub repository and run as per the instructions provided in the detailed README file.

C. GitHub

GitHub is a provider of internet hosting for software development and version control using Git. It offers the distributed version control and source code management functionality of Git, plus its own features. Initially, the idea was to host all files- the zipped images, Python scripts, and README files- all in one repository. However, due to the large file size of the zipped images, the file was stored as a blob in GitHub, meaning that it was stored in a remote server, and could only be accessed via downloading the file.

All project files are stored in GitHub, with instructions on how to import the dataset from Kaggle and how to run the program in the detailed README file.

V. RESULTS AND DISCUSSIONS

Both methods (CNN and DNN) were tested on sample images of different hand gestures with different orientations, skin colours and lightings. Figures 6 (a) and 6 (b) show the different classes present in the dataset.

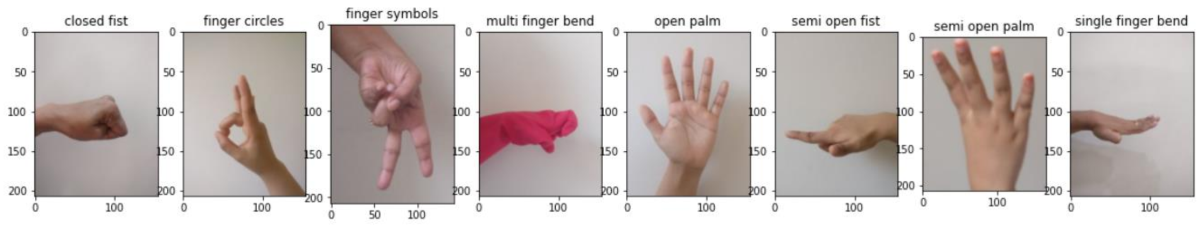


Figure 6 (a). Different images and the class they belong to

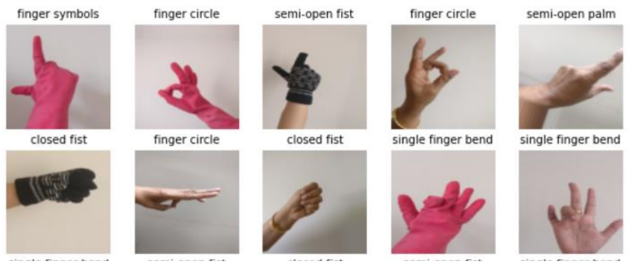


Figure 6 (b). Different images and the class they belong to

Figures 7 and 8 show the training and validation loss, as well as the training and validation accuracy for both methods. The attention-based CNN method produced a better graph than the DNN method.

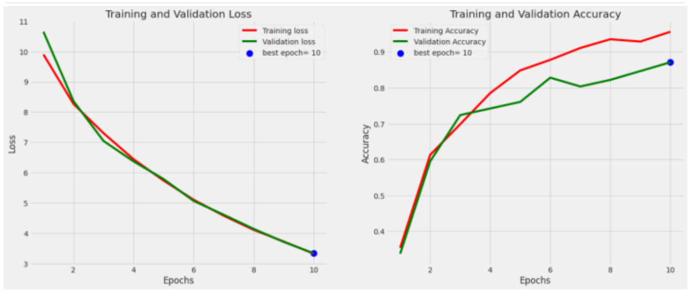


Figure 7. Loss and accuracy graphs for the attention-based CNN EfficientNetB5

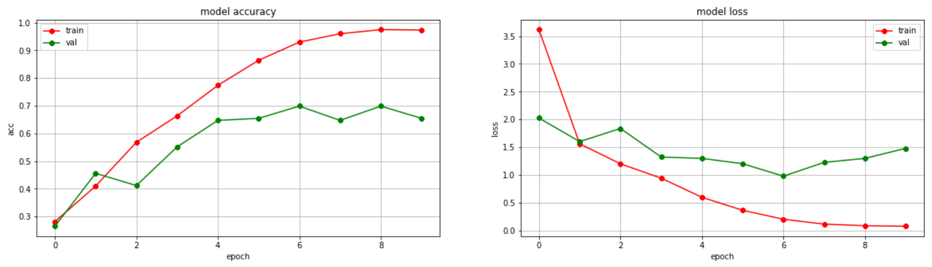


Figure 8. Loss and accuracy graphs for the DNN

In figure 7, the training and validation loss for the attention-based CNN are almost similar, while the training and validation accuracy produced better training accuracy than validation accuracy. In figure 8, due to the specific conditions of the pictures in the dataset (different orientations, skin colours and lightings), model accuracy and loss graphs for the DNN produced a very large discrepancy between the training and validation accuracy and loss, due to overfitting. This could be fixed by training with more data, regularisation algorithms, or stopping the training process before the learner passes a specific point.

Figures 9 and 10 show the confusion matrices for both methods. Based on the matrices, it can be concluded that both methods provided a higher accuracy than randomly guessing, and are thus successful in what they attempt to accomplish.

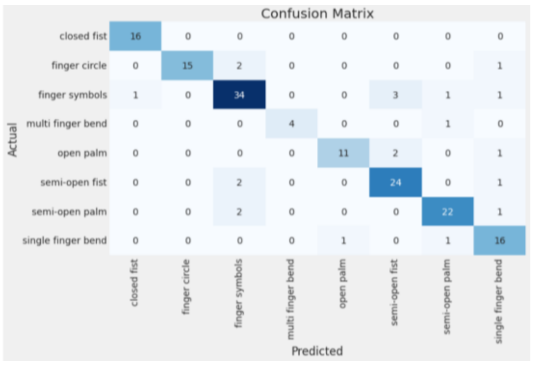


Figure 9. Confusion matrix for the attention-based CNN

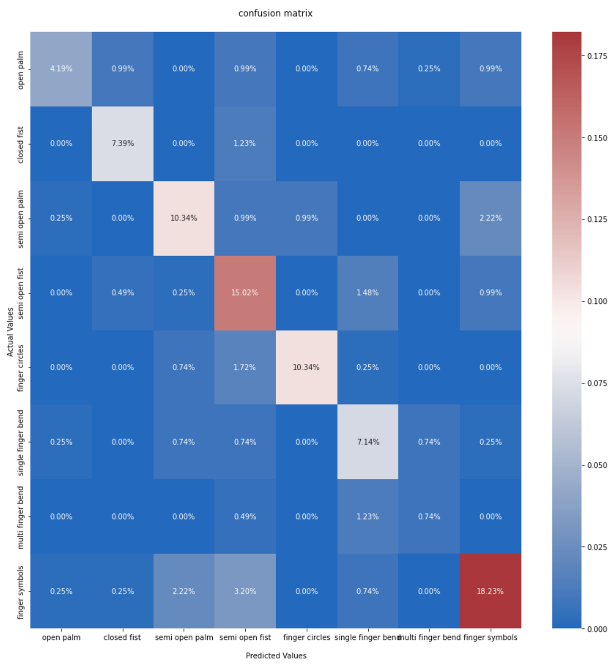
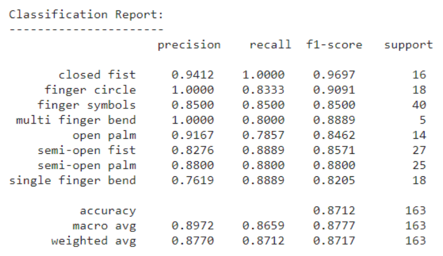


Figure 10. Confusion matrix for the DNN

In both figures 9 and 10, finger symbols gave the highest accuracy (34% for CNN and 18.23% for DNN) when it came to predicted vs. actual values in the confusion matrix. Multi finger bend gave the lowest accuracy (4% for CNN and 0.74% for DNN) for both of the methods used.

The attention-based CNN approach provided a noticeably better accuracy than the DNN approach, which may also be discerned when looking at the F1-scores of the methods, which are shown in figures 11 and 12.

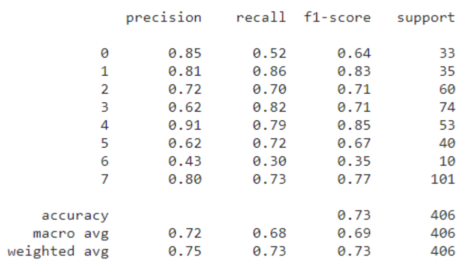


```
Classification Report:
----------------------
                    precision    recall  f1-score   support

       closed fist     0.9412    1.0000    0.9697        16
     finger circle     1.0000    0.8333    0.9091        18
    finger symbols     0.8500    0.8500    0.8500        40
 multi finger bend     1.0000    0.8000    0.8889         5
         open palm     0.9167    0.7857    0.8462        14
    semi-open fist     0.8276    0.8889    0.8571        27
    semi-open palm     0.8800    0.8800    0.8800        25
single finger bend     0.7619    0.8889    0.8205        18

          accuracy                         0.8712       163
         macro avg     0.8972    0.8659    0.8777       163
      weighted avg     0.8770    0.8712    0.8717       163
```

Figure 11. F1-score for the attention-based CNN



```
              precision    recall  f1-score   support

           0       0.85      0.52      0.64        33
           1       0.81      0.86      0.83        35
           2       0.72      0.70      0.71        60
           3       0.62      0.82      0.71        74
           4       0.91      0.79      0.85        53
           5       0.62      0.72      0.67        40
           6       0.43      0.30      0.35        10
           7       0.80      0.73      0.77       101

    accuracy                           0.73       406
   macro avg       0.72      0.68      0.69       406
weighted avg       0.75      0.73      0.73       406
```

Figure 12. F1-score for the DNN

The attention-based CNN approach needed 630.03 seconds to check 2,146 images, whereas the DNN approach needed 21.46 seconds to check the same number of images. Both methods were timed using Python's time library and were run on the same computer.

Further studies could be done using a pattern recognition technique developed by McConnell [11], which employs the histograms of local orientation by using the orientation histogram as a feature vector for gesture classification and interpolation. Freeman and Roth [5] propose that for static hand gestures, the histogram of local orientations is used as a feature vector for recognition. The authors conclude that this technique works well to identify hand gestures from a training vocabulary of gestures for close-up images of the hand. The real-time system lets the user control a computer graphic crane by hand gestures, monitor hand orientation, and play games such as rock-paper-scissors.

While the dataset is new, and still quite untrained when compared to more popular datasets (like CIFAR-10), the accuracy results were mostly in line with other works using the same algorithms [3, 16]. Another factor to be considered is background noise. The images in the dataset have either a plain white or a plain off-white background. Further testing could also be done to see how well the models perform on hand gestures taken in front of traffic, plants, and other materials that would have to undergo prior image pre-processing. Shadows are another category to consider, as the AI could consider them to be 2 different hands in the same picture.

Based on the results from running the CNN and DNN models on the dataset and the confusion matrices (figures 11 and 12), one can conclude that orientation does not produce much difference for certain classes of the dataset (mainly finger symbols, open palm, closed fist), but makes a huge difference in other classes (mainly multi-finger bend). Thus, orientation of the gestures, to some extent, does affect the results and accuracy of the models. The attention-based CNN model also produced a higher accuracy of 87.12%, when compared to the DNN model's accuracy of 76.49%.

VI. CONCLUSION

In this paper, two methods of performing image gesture classification have been proposed. The methods are using attention-based CNNs and DNNs, and provide an accurate prediction as to which class a hand gesture belongs to. The speed of the attention-based CNN approach leaves something to be desired, whereas the DNN approach runs well. This could be because the data was untrained prior to using. The classification of hand gestures, which have high application values in the fields of AI and robotics, are the problems that would be considered for future work.

REFERENCES

1. Albawi S., Mohammed T., Al-Zawi S. (2017), Understanding of a convolutional neural network, *International Conference on Engineering and Technology (ICET), 2017, pp. 1-6*, doi: 10.1109/ICEngTechnol.2017.8308186
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8308186

2. Bahdanau D., Cho K., Bengio Y. (2015), Neural Machine Translation by Jointly Learning to Align and Translate, *Published as a conference paper at ICLR 2015*
https://arxiv.org/pdf/1409.0473.pdf

3. Brownlee J. (2019), Your First Deep Learning Project in Python with Keras Step-By-Step, Accessed 28.05.2022
https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/

4. Fisher R., Perkins S., Walker A., Wolfart E. (2003), Convolution, Accessed 27.05.2022
https://homepages.inf.ed.ac.uk/rbf/HIPR2/convolve.htm

5. Freeman W., Roth M. (1994), Orientation Histograms for Hand Gesture Recognition, *IEEE Intl. Wkshp. on Automatic Face and Gesture Recognition, 1995*
https://john.cs.olemiss.edu/heroes/papers/hand_gesture.pdf

6. Johnson J. (2020), What's a Deep Neural Network? Deep Nets Explained, *Machine Learning & Big Data Blog*, Accessed 29.05.2022
https://www.bmc.com/blogs/deep-neural-network

7. Keras, The Sequential class, Accessed 28.05.2022
https://keras.io/api/models/sequential/

8. Keras, The Sequential model, Accessed 28.05.2022
https://keras.io/guides/sequential_model/

9. Lee K. (2021), Hardware Accelerator Systems for Artificial Intelligence and Machine Learning, *Advances in Computers Volume 122, 2021, Pages 217-245*
https://doi.org/10.1016/bs.adcom.2020.11.001

10. MATLAB® Tech Talk, Introduction to Deep Learning: What Are Convolutional Neural Networks?, Accessed 29.05.2022
https://in.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

11. McConnell R. (1986), Method of and apparatus for pattern recognition, *U. S. Patent No. 4,567,610, 1986*

12. Mishra M. (2020), Convolutional Neural Networks, Explained, *Published in Towards Data Science*, Accessed 29.05.2022
https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

13. Molchanov P., Gupta S., Kim K., Kautz J. (2015), Hand Gesture Recognition With 3D Convolutional Neural Networks, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2015, pp. 1-7*
https://www.cv-foundation.org/openaccess/content_cvpr_workshops_2015/W15/papers/Molchanov_Hand_Gesture_Recognition_2015_CVPR_paper.pdf

14. O'Shea K., Nash R. (2015), An Introduction to Convolutional Neural Networks
https://doi.org/10.48550/arXiv.1511.08458

15. Oudah M., Al-Naji A., Chahl J. (2020), Hand Gesture Recognition Based on Computer Vision: A Review of Techniques, *Journal of Imaging, 2020; 6(8):73*
https://doi.org/10.3390/jimaging6080073

16. Poisenka G. (2022), Watches F1 score=82%, Accessed 27.05.2022
https://www.kaggle.com/code/gpiosenka/watches-f1-score-82

17. Sarkar A., Sanyal G., Majumder S. (2013), Hand Gesture Recognition Systems: A Survey, *International Journal of Computer Applications (0975 – 8887) Volume 71– No.15*
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.8284&rep=rep1&type=pdf

18. Tan M., Le Q. (2020), EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, *International Conference on Machine Learning, 2019*
https://arxiv.org/abs/1905.11946

19. Venkatachalam M. (2019), An introduction to Attention, Published in Towards Data Science, Accessed 29.05.2022
https://towardsdatascience.com/an-introduction-to-attention-transformers-and-bert-part-1-da0e838c7cda

20. Wikipedia, Kaggle, Accessed 27.05.2022
https://en.wikipedia.org/wiki/Kaggle