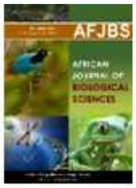
https://doi.org/10.33472/AFJBS.6.Si2.2024.2833-2838

AFJBS

African Journal of Biological Sciences

Journal homepage: http://www.afjbs.com



Research Paper Open Access

Dynamic Sign Language Recognition Through An Ensemble Of Deep Learning Techniques

[1]Rama Krishna Gandi, [2]Mattam Shaik Yaseen, [3]Syed Sahail, [4]Shaik Sameer, [5]Shaik Usman

1 Assistant Professor, 2, 3, 4,5 UG Students
Department of Computer Science & Engineering,
Madanapalle Institute of Technology and Science (MITS)
Madanapalle, Andhra Pradesh
[1]gandiramakrisha2@gmail.com
[2]yaseenshaik0987@gmail.com
[3]sahailsyed42@gmail.com
[4]shaiksameer89419@gmail.com
[5]shaikusman481@gmail.com

Volume 6, Issue Si2, 2024

Received: 09 March 2024

Accepted: 10 April 2024

Published: 20 May 2024
doi:10.33472/AFJBS.6.Si2.2024.2833-2838

Abstract— The recognition and interpretation of American Sign Language (ASL) using computational methods stands as a significant advancement in bridging communication gaps between the Deaf community and the hearing world. This project aims to develop an accurate and By employing deep learning techniques, we propose a model that interprets ASL signs from static images and video sequences, enabling seamless translation of sign language gestures into textual or spoken language.Our methodology encompasses the collection and preprocessing of a comprehensive dataset consisting of ASL signs, including both alphabet and commonly used phrases. Utilizing state-of-the-art CNN architectures, the system undergoes rigorous training and validation phases to ensure high levels of accuracy and efficiency. Robustness and dependability are assessed by gauging the model's capacity to correctly identify a broad spectrum of ASL signals in a variety of backgrounds and environments.

Keywords— Sign_Language_Detection, Deep_Learning, Cnn, ensemble learning

I. INTRODUCTION

One essential component of human contact is communication., enabling individuals to express thoughts, share ideas, and build relationships. serves as a primary mode of communication, embodying a rich linguistic structure that allows for the conveyance of complex information its widespread use and importance, barriers still exist in communication between ASL users and those unfamiliar with the language, leading to challenges in education, employment, and access to services.

new opportunities have emerged to bridge these communication gaps. Particularly, shown exceptional prowess in image and video recognition tasks,. This project aims to harness the potential of CNNs to develop a robust and accurate ASL detection system capable of translating ASL signs into textual or spoken real-time language use.The reason behind this endeavor is twofold: to enhance the accessibility of services and information for the Deaf community and to facilitate mutual understanding and interaction between ASL users and the hearing world. By leveraging the capabilities of CNNs, we propose a solution that not only recognizes individual ASL signs with high accuracy but also interprets continuous sign language gestures from video inputs.

Fig(a) Sign Language

II. Research Work

To facilitate the extraction of signs from video sequences, color segmentation is utilized, along with the isolate the region. In the realm of object detection, the implementation of Faster RCNN enhances the speed of detection by normalizing input to a uniform size before processing it through the convolutional block. Sign decoding at the sentence level involves extracting centroids of both facial and spatial elements, determined by fuzzy membership class functions. For inputs comprising sequences of gestures, an LSTM model is employed to decipher a continuous array of signs, which are then segmented into smaller components for (FBPNN). The SIFT descriptor is essential torecognizing both alphabets and numerals.

The suggested algorithm efficiently extracts signs and features from continuous video sequences by applying color segmentation for hand image identification, although it requires a minimally cluttered background for optimal results. Support Vector Machines are utilized to differentiate The Viola-Jones algorithm aids in facial region removal from video sequences, focusing solely on hand signs. Zernike moments serve as the shape descriptor for static signs, while curve features are analyzed for dynamic gestures.

The model is primarily based on hand movements, which are complemented by body placement and face expressions that were trained on large datasets.. Feature extraction is performed using the convex hull method, with classification via KNN, achieving an accuracy rate of 65%. The system compares favorably to popular methods, with Faster R-CNN models offering increased detection speed through RPN modules. It demonstrates superior gesture location detection accuracy compared to YOLO, utilizing a 3D CNN network with four convolution blocks and ReLU activation.

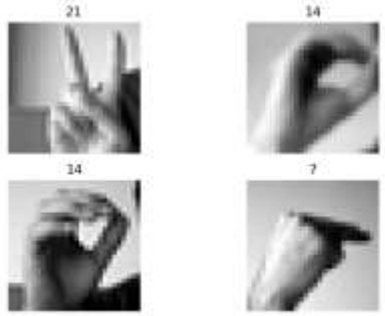
The system preprocesses input sequence images captured by a webcam using classifies hand postures using KNN.

This technique's potential extends to OCR (Optical Character Recognition), offering a significant advancement in understanding sentence-level signs by speech-impaired individuals through fuzzy membership functions. Continuous Sign Language Recognition is achieved with an LSTM model, focusing on Indian Sign Language with a test dataset comprising 942 signed sentences, achieving accuracies integrates MLP and autoencoders for feature globalization, classifying via the SoftMax algorithm. It was trained on 1080 videos of 10 dynamic ISL gestures using NVIDIA Tesla K80 GPUs. The VGG11 Model, Finally, the binary transformation of detected regions and Euclidean distance transformation are key post-processing steps, showcasing the versatility and comprehensive nature of this ASL detection project.

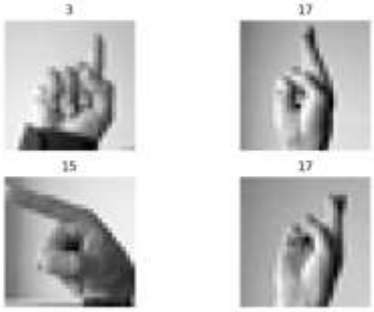
III. DATASET

The MNIST American Hand Sign Language dataset is a comprehensive collection designed to facilitate the development and testing of machine learning models capable of recognizing American Sign Language (ASL) signs. Inspired by the original MNIST dataset, which is widely used for benchmarking image processing systems and techniques, this specialized dataset focuses on the visual representation of ASL alphabets through hand gestures. It plays a crucial role in advancing technologies aimed at bridging communication gaps for the people cant hear and hard-of-hearing communities by enabling the development of automated ASL recognition systems.

The dataset typically consists of images representing the 26 letters of the English alphabet as expressed in ASL hand gestures. These images are often grayscale, normalized in size, and centered to ensure consistency and ease of use in training CNNs and other machine learning models. Each image in the dataset is labeled with its corresponding ASL alphabet letter, making it suitable for supervised learning tasks, including classification and recognition.



Fig(b) MNIST Hand Sign Language Dataset



Fig(c) MNIST Hand Sign Language Dataset

Feature Extraction:

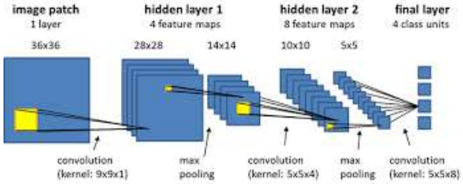
Convolutional layers act as filters for extracting out features from an image. After passing the image through a number of such layers and maxpooling layers which remove excess information, we end up with a flattened array of values which are essentially the features extracted from the image that will be passed to the dense layers for training..

Training and testing:

27855 images in the dataset were divided into two sets of training and testing that were 75% and 25%, respectively, using scikit-learn's train_test_split function.The network of neurons is fed a picture with dimensions of 28x28x1. The layers make use of maxpooling and the relu activation function.

The process for recognizing hand gestures involves: Data Collection: The well-known MNIST American sign language dataset, which includes 28x28 grayscale pictures of hand signals, was employed by us.

Data Preprocessing: There is no background in these images. The images are present in a csv format with 784 columns which has to be read into a pandas data frame. Every row represents one image with 784 pixel values which needs to be reshaped into 28x28 size.The pixel values are also normalized by dividing them by 255. This is done for faster convergence of the network.



Fig(d). Feature Extraction

Since we are dealing with grayscale images, the number of channels is one. For our work, the channel does not convey any extra information about the sign gesture. Hence, using RGB images does not have any advantage over using grayscale images. On the other hand, using grayscale images will lead to faster processing times.

Classification of Gestures

- Initially, The 1D vectors are reshaped and pixel values are normalized.
- This preprocessed image is then analyzed using a CNN model..
- The CNN then gives an inference based on its training.

IV. ALGORITHM USED

Convolution Neural Network

The conv2D layers operate as filters, extracting characteristics from the photos. We have utilized three separate layers of 32 filters in total, with a single unit size of 3x3, next to a a dropout layer and the maximum pooling level.There are then three further multivariate 2D layers. including sixteen 3x3 filters, succeeded by a maxpooling, dropout, and flatten layer. Next, we use a flatten layer, a dense for dropping to turn the feature list into a 1D array. There are 25 nodes in the final layer that use the activation of soft max for classification.

i) We use relu activation in all but the last layer and padding is kept as ‘same’ since the image sizes are small so we do not want to lose any information during the process of convolution.

ii) Layer for pooling: Each feature map is handled separately by the MaxPooling layer. The initial feature map is split into a group of non-producing zones based on the greatest value of each overlapping rectangle. This method reduces the spatial dimensions of the feature map so that the output reference map is less as the input option map.By constricting the spatial dimensions of the feature maps, MaxPooling layers lower the amount of computing and factors in the network. By giving the representation in an abstracted form and enabling the CNN to make judgments based on the most salient characteristics from the preceding layers, this helps to lessen overfitting..

MaxPooling helps the network to become invariant to small translations of the input image. Since it takes the maximum value in a local pool, slight shifts or translations in the input image will still likely result in the same maximum value, allowing the network to recognize features regardless of their exact location in the input.The MaxPooling layer has two key parameters: pool size and stride. The pool size determines the size of the region over which the maximum is computed. The stride determines the distance between consecutive pooling regions.

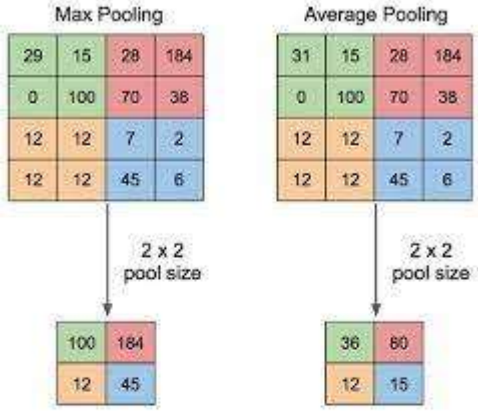


Fig. (e) Pooling

iii) Dense Layer: In neural networks, fully linked layers—also referred to as dense layers—are essential to acquire information non-linear mixtures of the high-level information that the network's preceding levels were able to extract. The reason these layers are called "fully connected" is that every neuron in a dense layer receives input from every other neuron in the layer above it. creating a densely connected structure.Dense layers integrate the features learned by the network so far to make predictions. They are typically placed towards the end of a CNN after convolutional and pooling layers have extracted spatial hierarchies of features.Each neuron in a fully connected layer holds a set of weights and a bias, applying a linear transformation (dot product) followed by a non-linear activation function like ReLU, sigmoid, or tanh to the input data.

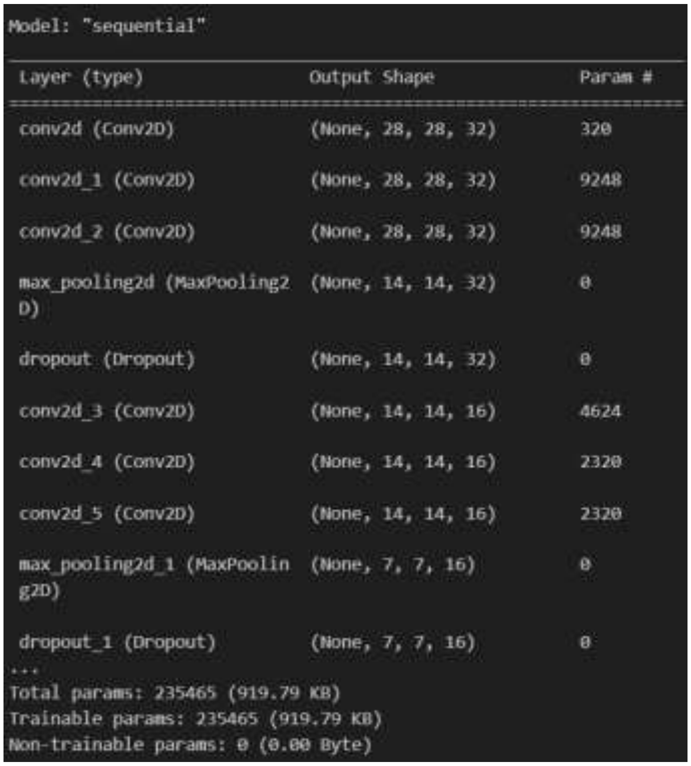
iv)Dropout layer:

The dropout layer is a regularization technique used by neural networks to prevent overfitting. When a model learns using training data overly well, it overfits and has poor generalization to intriguing, unseen data by collecting noise and random oscillations. Dropout solves this problem by arbitrarily "falling off" (i.e., set to zero) a percentage of the network's neurons at each training update.

Architecture used:

The following cnn architectures have been used, which have shown very high accuracy on the test dataset. An ensemble of CNN models have been used and the best model is taken into consideration.

Model 1: This model aimed at capturing hierarchical features in the input images. Dropout layers are strategically placed to prevent overfitting. The model concludes with dense layers for classification, producing probabilities across 25 output classes. For training, it uses categorical cross-entropy loss and the RMSprop optimizer..

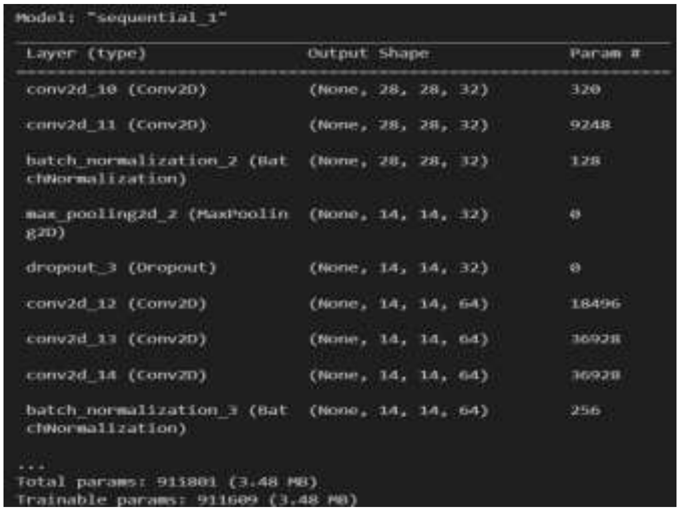


```
Model: "sequential"

 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320

 conv2d_1 (Conv2D)           (None, 28, 28, 32)        9248

 conv2d_2 (Conv2D)           (None, 28, 28, 32)        9248

 max_pooling2d (MaxPooling2  (None, 14, 14, 32)        0
 D)

 dropout (Dropout)           (None, 14, 14, 32)        0

 conv2d_3 (Conv2D)           (None, 14, 14, 16)        4624

 conv2d_4 (Conv2D)           (None, 14, 14, 16)        2320

 conv2d_5 (Conv2D)           (None, 14, 14, 16)        2320

 max_pooling2d_1 (MaxPoolin  (None, 7, 7, 16)          0
 g2D)

 dropout_1 (Dropout)         (None, 7, 7, 16)          0
...
Total params: 235465 (919.79 KB)
Trainable params: 235465 (919.79 KB)
Non-trainable params: 0 (0.00 Byte)
```

Fig. Architecture of model 1

Model 2: Model 2 adopts a similar convolutional architecture to Model 1 but integrates batch normalization layers after certain convolutional and pooling operations, enhancing training stability and convergence speed. Dropout layers are incorporated for regularization, and the model's structure culminates in dense layers for classification. It also uses the multi-class classification tasks' softmax activation function

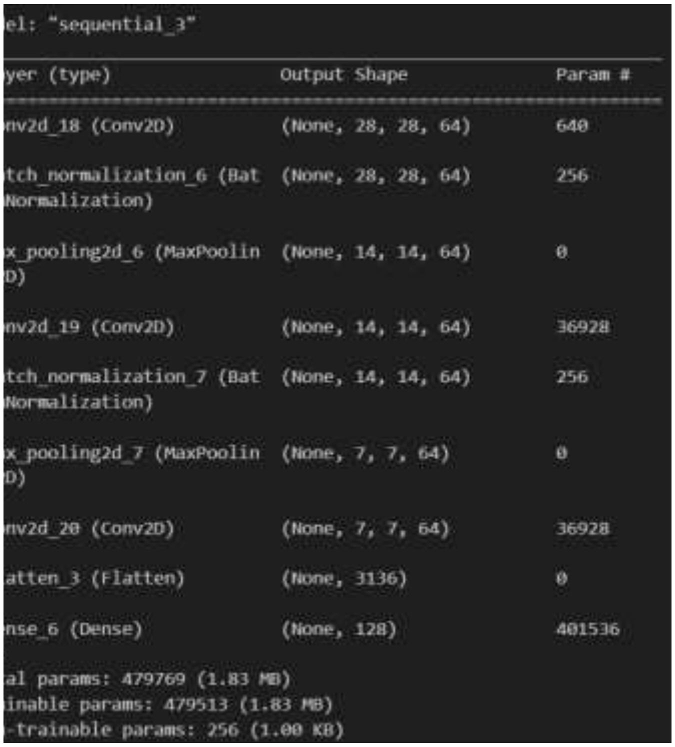
.



```
Model: "sequential_1"

 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 28, 28, 32)        320

 conv2d_11 (Conv2D)          (None, 28, 28, 32)        9248

 batch_normalization_2 (Bat  (None, 28, 28, 32)        128
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 14, 14, 32)        0
 g2D)

 dropout_3 (Dropout)         (None, 14, 14, 32)        0

 conv2d_12 (Conv2D)          (None, 14, 14, 64)        18496

 conv2d_13 (Conv2D)          (None, 14, 14, 64)        36928

 conv2d_14 (Conv2D)          (None, 14, 14, 64)        36928

 batch_normalization_3 (Bat  (None, 14, 14, 64)        256
 chNormalization)

...
Total params: 913801 (3.48 MB)
Trainable params: 911609 (3.48 MB)
```

Fig. Architecture of model 2

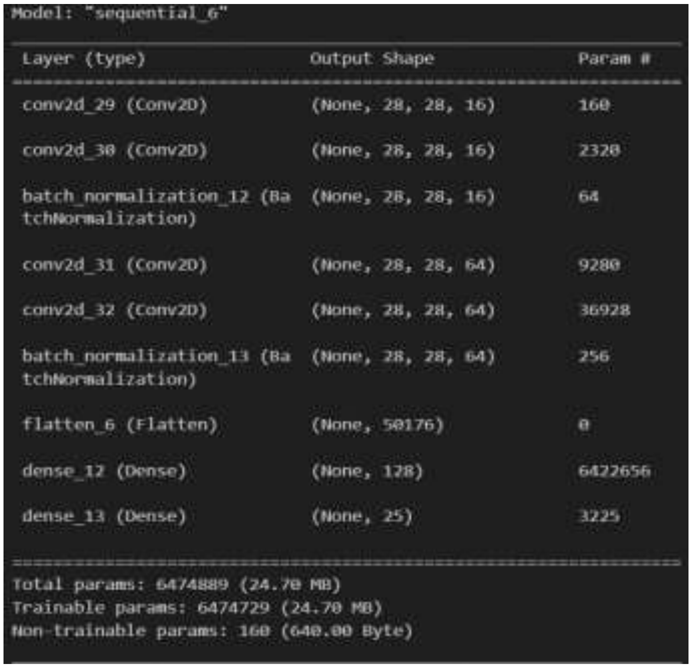
Model 3: This model features a simplified architecture compared to the previous ones, with fewer convolutional layers and no dropout. Batch normalization is applied after selected convolutional to aid in training. aims to strike a balance between complexity and performance, with a focus on capturing essential features for accurate classification.



```
el: "sequential_3"

yer (type)                Output Shape              Param #
===============================================================
nv2d_18 (Conv2D)          (None, 28, 28, 64)        640

tch_normalization_6 (Bat  (None, 28, 28, 64)        256
Normalization)

x_pooling2d_6 (MaxPoolin  (None, 14, 14, 64)        0
D)

nv2d_19 (Conv2D)          (None, 14, 14, 64)        36928

tch_normalization_7 (Bat  (None, 14, 14, 64)        256
Normalization)

x_pooling2d_7 (MaxPoolin  (None, 7, 7, 64)          0
D)

nv2d_20 (Conv2D)          (None, 7, 7, 64)          36928

atten_3 (Flatten)         (None, 3136)              0

nse_6 (Dense)             (None, 128)               401536

al params: 479769 (1.83 MB)
inable params: 479513 (1.83 MB)
-trainable params: 256 (1.00 KB)
```

Fig. Architecture of model 3

Model 4: Model 4 experiments with a different architectural approach, starting with a reduced number of convolutional layers compared to Models 1 and 2. It incorporates batch normalization early in the network to stabilize training. However, unlike the other models, it uses two consecutive convolutional layers with smaller filter sizes before introducing batch normalization. The model concludes with dense layers for classification into 25 output classes.



```
Model: "sequential_6"

 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_29 (Conv2D)          (None, 28, 28, 16)        160

 conv2d_30 (Conv2D)          (None, 28, 28, 16)        2320

 batch_normalization_12 (Ba  (None, 28, 28, 16)        64
 tchNormalization)

 conv2d_31 (Conv2D)          (None, 28, 28, 64)        9280

 conv2d_32 (Conv2D)          (None, 28, 28, 64)        36928

 batch_normalization_13 (Ba  (None, 28, 28, 64)        256
 tchNormalization)

 flatten_6 (Flatten)         (None, 50176)             0

 dense_12 (Dense)            (None, 128)               6422656

 dense_13 (Dense)            (None, 25)                3225

=================================================================
Total params: 6474889 (24.70 MB)
Trainable params: 6474729 (24.70 MB)
Non-trainable params: 160 (640.00 Byte)
```

Fig. Architecture of Model 4

Activation function:

One of the most popular activated functions in neural networks, particularly CNNs, is ReLU, or Rectified Linear Unit. The formula for the function is f(x)=max(0,x).

ReLU adds the non-linear to the model despite its simplicity, which enables it to recognize intricate patterns in the data.

For multi-class classification tasks, the softmax function is usually utilized in the final results layer of a neural network. By taking the increasing of each output and normalizing it by dividing by the total of all the exponentials, it turns the raw output scores (commonly referred to as logits) via the network into probabilities. The definition of the function is:

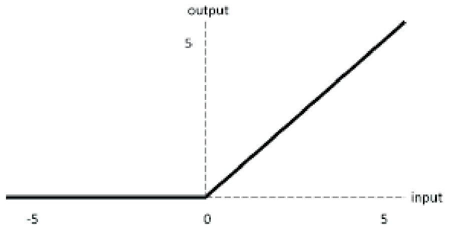


Fig. (f) Relu Activation Function [16]

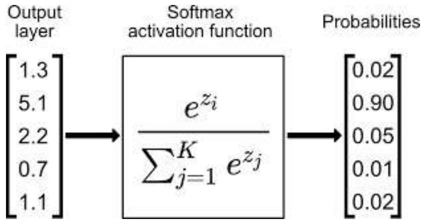

Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

Fig. (g) Softmax Activation Function

V. Procedure

- The MNIST american hand sign language dataset is acquired and preprocessed.
- During preprocessing we convert the pixel values into a 2 dimensional numpy array and perform normalization by dividing them by 255.
- After that we build the CNN and train it using the training set obtained from a 0.75-0.25 random split of the dataset.
- After training we evaluate the model on the testing set.
- We can take a picture from a camera and preprocess it and then send it to the network for inference.

VI. Model comparison

The CNN outperformed the model trained using transfer learning with vgg16 as its base. Moreover, the training was brisk since image sizes were small and all images are grayscale. The model had 99.3% validation accuracy during training and 99.07% accuracy on the testing dataset.

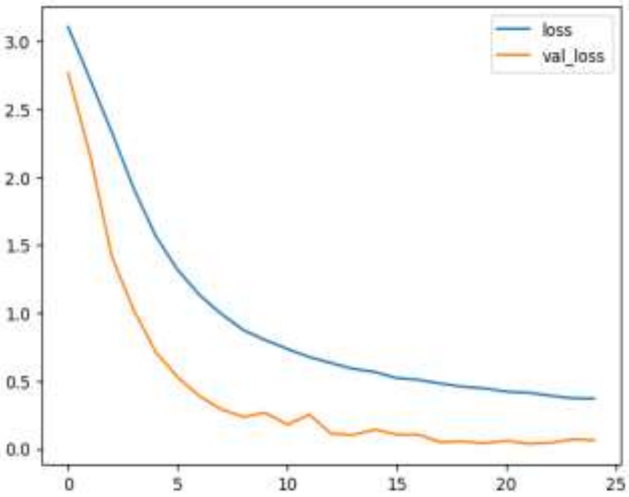


Fig. (h) Loss Plots versus epoch

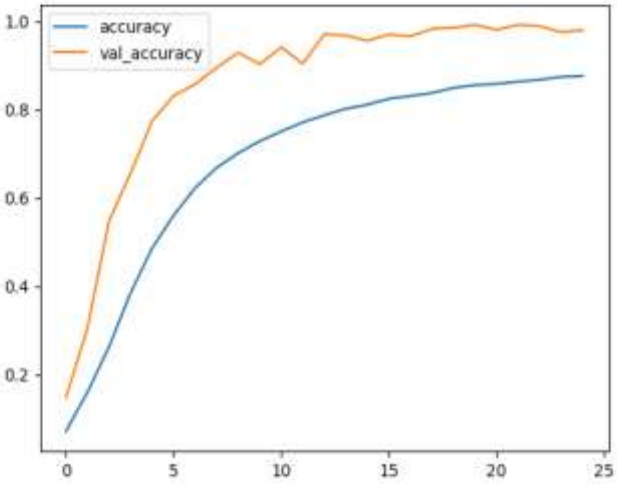
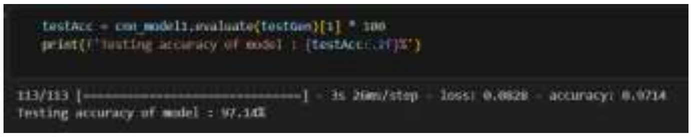


Fig. (i) Accuracy Plots versus epochs

VII. Observations

In the first eleven epochs out of 30, we have a validation accuracy of 94%. In the last ten epochs the validation accuracy increased from 95% to 97%. The accuracy on the test dataset is 97.14%.



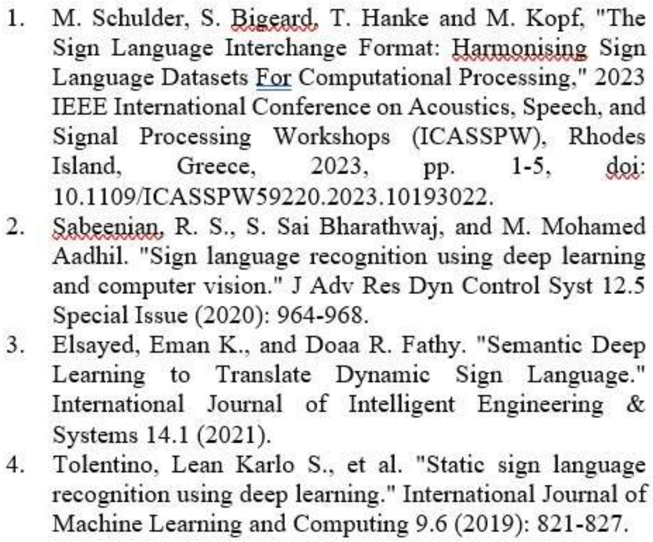
```
testAcc = cnn_model1.evaluate(testGen)[1] * 100
print(f'Testing accuracy of model : {testAcc:.2f}%')

113/113 [==============================] - 3s 26ms/step - loss: 0.0828 - accuracy: 0.9714
Testing accuracy of model : 97.14%
```

Fig. (j) Result

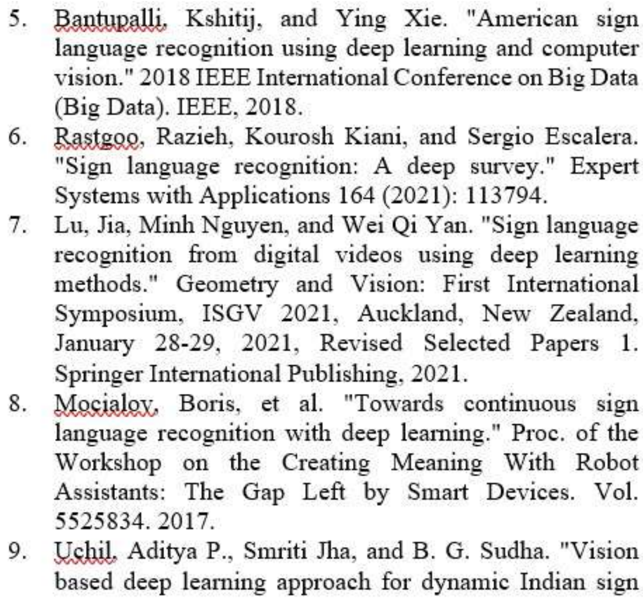
VIII. Conclusion

We created a light convolutional model for american sign language detection and it produced an accuracy of over 97.14%. This method can be extended to include numbers and Indian sign language as well. After that a real-time system can be created that will take the image feed from a video capture device and decipher the hand sign upon inspection. Further experimentation with different types of pre-trained models can be performed. Initial use of vgg16 did not yield satisfactory results, mostly due to the low resolution of the images. On the other hand, while using real-time images these pre-trained models may come in hand

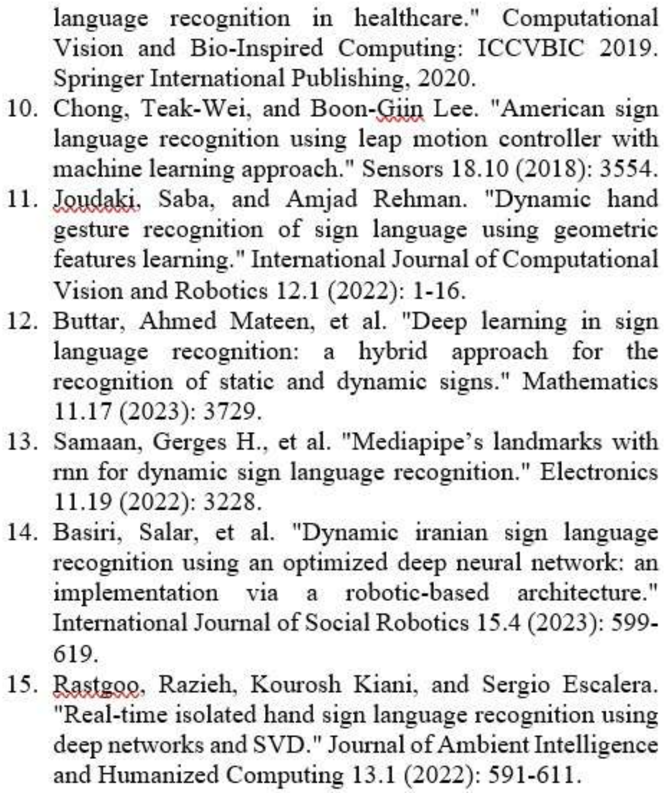
IX REFERENCES



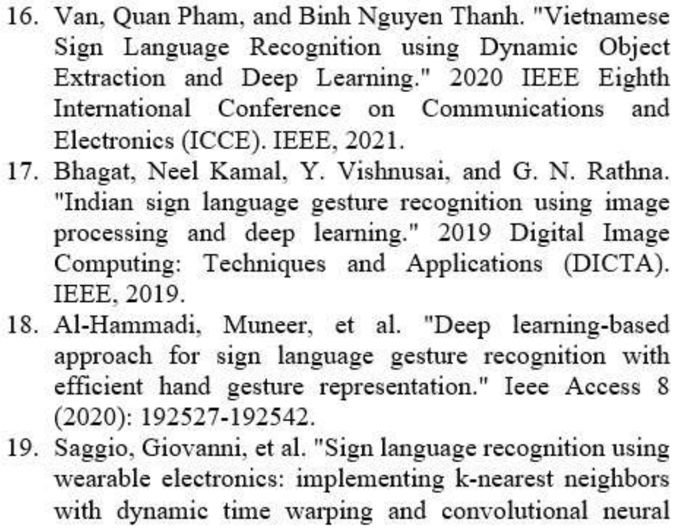
1. M. Schulder, S. Bigeard, T. Hanke and M. Kopf, "The Sign Language Interchange Format: Harmonising Sign Language Datasets For Computational Processing," 2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW), Rhodes Island, Greece, 2023, pp. 1-5, doi: 10.1109/ICASSPW59220.2023.10193022.
2. Sabeenian, R. S., S. Sai Bharathwaj, and M. Mohamed Aadhil. "Sign language recognition using deep learning and computer vision." J Adv Res Dyn Control Syst 12.5 Special Issue (2020): 964-968.
3. Elsayed, Eman K., and Doaa R. Fathy. "Semantic Deep Learning to Translate Dynamic Sign Language." International Journal of Intelligent Engineering & Systems 14.1 (2021).
4. Tolentino, Lean Karlo S., et al. "Static sign language recognition using deep learning." International Journal of Machine Learning and Computing 9.6 (2019): 821-827.



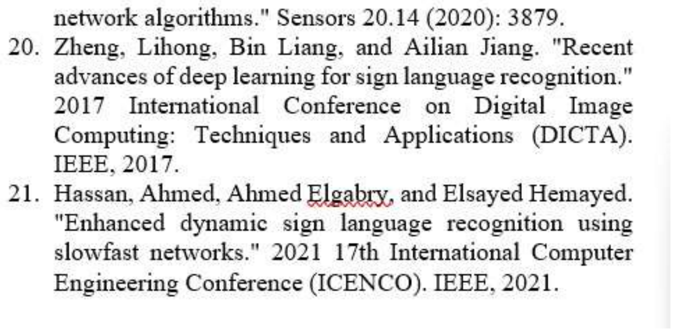
5. Bantupalli, Kshitij, and Ying Xie. "American sign language recognition using deep learning and computer vision." 2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018.
6. Rastgoo, Razieh, Kourosh Kiani, and Sergio Escalera. "Sign language recognition: A deep survey." Expert Systems with Applications 164 (2021): 113794.
7. Lu, Jia, Minh Nguyen, and Wei Qi Yan. "Sign language recognition from digital videos using deep learning methods." Geometry and Vision: First International Symposium, ISGV 2021, Auckland, New Zealand, January 28-29, 2021, Revised Selected Papers 1. Springer International Publishing, 2021.
8. Mocialov, Boris, et al. "Towards continuous sign language recognition with deep learning." Proc. of the Workshop on the Creating Meaning With Robot Assistants: The Gap Left by Smart Devices. Vol. 5525834. 2017.
9. Uchil, Aditya P., Smriti Jha, and B. G. Sudha. "Vision based deep learning approach for dynamic Indian sign



language recognition in healthcare." Computational Vision and Bio-Inspired Computing: ICCVBIC 2019. Springer International Publishing, 2020.
10. Chong, Teak-Wei, and Boon-Giin Lee. "American sign language recognition using leap motion controller with machine learning approach." Sensors 18.10 (2018): 3554.
11. Joudaki, Saba, and Amjad Rehman. "Dynamic hand gesture recognition of sign language using geometric features learning." International Journal of Computational Vision and Robotics 12.1 (2022): 1-16.
12. Buttar, Ahmed Mateen, et al. "Deep learning in sign language recognition: a hybrid approach for the recognition of static and dynamic signs." Mathematics 11.17 (2023): 3729.
13. Samaan, Gerges H., et al. "Mediapipe's landmarks with rnn for dynamic sign language recognition." Electronics 11.19 (2022): 3228.
14. Basiri, Salar, et al. "Dynamic iranian sign language recognition using an optimized deep neural network: an implementation via a robotic-based architecture." International Journal of Social Robotics 15.4 (2023): 599-619.
15. Rastgoo, Razieh, Kourosh Kiani, and Sergio Escalera. "Real-time isolated hand sign language recognition using deep networks and SVD." Journal of Ambient Intelligence and Humanized Computing 13.1 (2022): 591-611.



16. Van, Quan Pham, and Binh Nguyen Thanh. "Vietnamese Sign Language Recognition using Dynamic Object Extraction and Deep Learning." 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE). IEEE, 2021.
17. Bhagat, Neel Kamal, Y. Vishnusai, and G. N. Rathna. "Indian sign language gesture recognition using image processing and deep learning." 2019 Digital Image Computing: Techniques and Applications (DICTA). IEEE, 2019.
18. Al-Hammadi, Muneer, et al. "Deep learning-based approach for sign language gesture recognition with efficient hand gesture representation." Ieee Access 8 (2020): 192527-192542.
19. Saggio, Giovanni, et al. "Sign language recognition using wearable electronics: implementing k-nearest neighbors with dynamic time warping and convolutional neural



network algorithms." Sensors 20.14 (2020): 3879.
20. Zheng, Lihong, Bin Liang, and Ailian Jiang. "Recent advances of deep learning for sign language recognition." 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA). IEEE, 2017.
21. Hassan, Ahmed, Ahmed Elgabry, and Elsayed Hemayed. "Enhanced dynamic sign language recognition using slowfast networks." 2021 17th International Computer Engineering Conference (ICENCO). IEEE, 2021.