Convolutional Neural Network Hand Gesture Recognition for American Sign Language

Shruti Chavan, Xinrui Yu and Jafar Saniie
Embedded Computing and Signal Processing Research Laboratory (http://ecasp.ece.iit.edu/)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, IL, U.S.A.

2021 IEEE International Conference on Electro Information Technology (EIT) | 978-1-6654-1846-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/EIT51626.2021.9491897

***Abstract*—With the advancements in the computer vision technology, learning and using sign languages to communicate with deaf and mute people has become easier. Exciting research is ongoing for providing a global platform for communication in different sign languages. In this paper, we present a Deep Learning based approach to recognize a sign performed in American Sign Language by capturing an image as input. The system can predict the signs of 0 to 9 digits performed by the user. By utilizing image processing to convert RGB data to grayscale images, efficient reduction is achieved in the storage requirements and training time of the Convolutional Neural Network. The objective of the experiment is to find a mix of Image Processing and Deep Learning Architecture with lesser complexity to deploy the system in mobile applications or embedded single board computers. The database is trained from scratch using smaller networks as LeNet-5 and AlexNet as well as deeper network such as Vgg16 and MobileNet v2. The comparison of the recognition accuracies is discussed in the paper. The final selected architecture has only 10 layers including a dropout layer which boosted the training accuracy to 91.37% and testing accuracy to 87.5%.**

Keywords—convolutional neural network, hand gesture recognition, sign language

I. INTRODUCTION

A sign language is a way of communicating by using the hands and other parts of the body. According to the World Health Organization, there are around 466 million people worldwide have disabling hearing loss, who often use sign language for communication[1]. Sign languages differ from country to country, and we mainly examine American Sign Language (ASL) in this paper. ASL is expressed be movements of the hands and face, and is the primary language of North Americans who have disabling hearing loss [2]. Still, there will be problems when these people try to communicate with ordinary people who are unfamiliar with ASL. As a result, an automatic and real-time sign language interpreter is not only necessary, but also in high demand for the people with and without hearing loss alike [3].

The expressions in ASL are difficult to recognize with conventional algorithms. However, with the help of convolutional neural networks, we can recognize and translate the sign language more efficiently [4]. In this paper, we analyzed the performance of different neural network models in this area to obtain a balanced neural network models, which does not only recognize sign languages with good accuracy but also runs on lower end embedded systems and smartphones with acceptable speed. This would be helpful in developing a real-time and portable sign language recognition system.

II. RELATED WORK

Gesture recognition is a competitive area of research. It may include different approaches to recognize the gesture such as usage of sensory hardware. But using a hardware is costly and less convenient in real life. Thus, using computer vision techniques, researchers are trying to get the best recognition accuracy. Performance of different glove-based and vision-based sign language detection was compared by Sruthi C. J. and A. Lijiya [5].

Deep Learning techniques are easier to use and advantageous as they are capable of extracting the features from the raw database. The depths of the Convolutional Neural Networks and the combination of CNN layers alter the performance of the system. Gesture recognition modules have been designed to recognize the sign from a selfie-video taken on mobile phones [6]. A new dataset is created for Indian Sign Language with five different subjects performing 200 signs in 5 different viewing angles. The CNN consists of 4 convolution layers. Each convolution layer has a different filter size which improved speed and accuracy. Stochastic pooling layer is introduced for better feature extraction as it sums up the advantages of max and mean pooling. Surely this approach provides a platform for a visual interaction to deaf and mute users but may put limitations on performing the signs as user has to hold the phone in one hand.

Skin detection algorithms help remove unnecessary background and this increases the model accuracy [7]. Though, skin detection process eliminates the noise in the image, it also elongates the recognition time. Use of fast GPUs can help overcome this issue, but will result in a costly system.

Skin models are designed to eliminate the effects of non-uniform illuminance [8]. Also, this skin model followed by the rectification of the hand orientation and obtaining a binary image with Gaussian Mixture Model, further increases the validation accuracy of the CNN.

As the research in sign language recognition started encouraging scientists and students to implement their own CNN architecture or to use Transfer Learning approach, there is a great demand for American Sign Language database. Some Computer Vision Scientists and laboratories have made their database public for the world so that more novel solutions can be brought up [9].

978-1-6654-1846-1/21/$31.00 ©2021 IEEE

III. Proposed Method

As researchers prefer to use Convolutional Neural Networks for feature extraction from gesture datasets, after investigating the existing related designs and studying the available alternatives given several constraints, we can implement a design that is based on a combination of image pre-processing and deep learning to achieve our sought-after objectives. It is easier for users to simply capture a photo of a hand sign and feed it to the system. The task of feature extraction is performed by the CNNs only. Real time prediction accuracies can be improved by trying transfer learning, CNN fine-tuning methods.

A. System Description

The model of the neural networks used in this paper can be described as a multilayer perceptron. A block diagram of the system is shown in Fig. 1. Description for each block is given below.

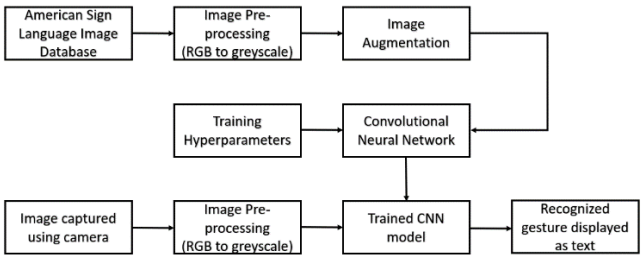

Figure 1. System Block Diagram

- *Image Database*: The database contains the images of different hand signs. These images are taken from different users with multiple repetitions. The resolution of the images may be varying. Different datasets are available for American Sign Language.
- *Image Pre-processing*: Training the raw images as it is might lead to poor performance. Thus, simple image processing algorithms can be implemented to achieve maximum accuracy. Image processing algorithms such as RGB to gray conversion reduce the training time and power consumption. The noise from the images can be eliminated.
- *Image Augmentation*: Data augmentation helps in case of a small database. Image augmentation is achieved by doing various operations, including: Mirroring – Flip the image horizontally; Cropping – Cutting out a certain portion of an image; Rotating, shearing, local warping; Color shifting – For RGB dataset, the pixel values can be modified.
- *CNN Training & Training Options*: Deep learning is used for the project. Training options are set accordingly before training the database using any CNN architecture. The training options are Maximum batch size, number of epoch and learning rate.
- *Image Acquisition*: Any camera, even a laptop webcam can be used to acquire the image to be recognized. Because in the end the image captured will be reduced to input size of the CNN. Hence the camera need not be high-resolution.
- *Display Output*: The recognized sign can is displayed in text format or can be also conveyed with audio description.

B. Hardware Components

As the input to the system is in the form of an image, a camera is required to capture the photograph of the plant. There are no specific requirements regarding the camera. Any good quality and resolution camera can be used to take input. The resolution of the camera can be kept as low as possible.

C. Software Libraries

- *OpenCV*: to capture and process the images.
- *TensorFlow and Keras*: to train the plant dataset.
- *NumPy*: to hold a group of images to test the model.

IV. Implementation

A. Database

The biggest challenge we faced during the experiments was to find a suitable and sufficient sign language dataset. We used American Sign Language dataset made available publicly by Turkey Ankara Ayrancı Anadolu High School which comprises approximately 205 images per class (see Fig. 2) [10]. The dataset consists of hand gestures for 0 to 9 digits. The images are in RGB format with resolution of 100x100.

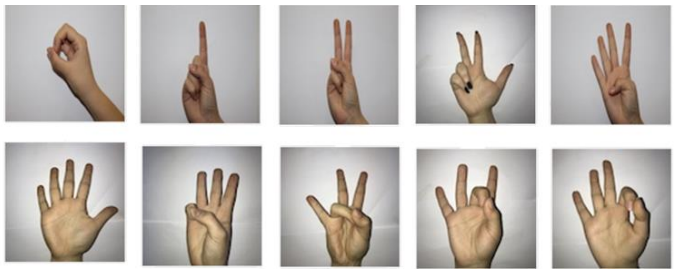

Figure 2. Sample Images from the American Sign Language Dataset

B. Image Preprocessing

After converting the RGB image dataset to grayscale, three image processing algorithms were implemented and compared for better results. First method is simple thresholding, for which the source image should be a grayscale image. The threshold value is used to classify the pixel values in only two categories- 0 and 1. The pixel values are compared with pre-defined threshold value by programmer. The outputs of simple thresholding are binary images.

The second method is Otsu's thresholding, which automatically calculates a threshold value from an image histogram. It is used to perform automatic image thresholding. In the simplest form, the algorithm returns a single intensity threshold that separate pixels into two classes, foreground and background. The result is an image with bimodal distribution. The bimodal distribution has two peaks, which are generated by combining two normal distribution curves.

The last method is Canny edge detection. It is a five stage algorithm [11]: Noise Reduction (use gaussian filter to smooth the image and remove the Noise), Gradient calculation (find the

intensity gradients of the image), Non-maximum Suppression (apply Non-Maximum suppression to get rid of spurious response to edge detection), Double threshold (determine the potential edges) and Hysteresis Thresholding (finalize the detection of the edges by suppressing all the other edges that are weak or not connected to strong edges). As these discussed methods didn't yield correct output images for entire data, we decided not to make any changes to obtained grayscale image database.

C. Convolutional Neural Networks

Convolutional neural networks are specifically designed to make inference from visual data such as images and videos. The features are extracted and learnt to tarin the model, which gives better recognition accuracy compared to conventional Machine Learning algorithms. CNNs have numerous applications in the field of Signal Processing, robotics, medical imaging, data analysis, Business Intelligence, etc. The learning from unaltered and smaller dataset with CNNs yields surprisingly better results.

CNN architecture is built with combination of several layers which can be referred as functional units of deep learning. The architecture takes input data, with predefined hyperparameters, learns from the data to decide the values of weights and biases. The accuracy or loss is checked after each iteration of learning process against a part of original data set aside before training, called as validation dataset. Descriptions for layers used in the experiments is given below:

Convolution layer- Convolution layers extract feature maps from the fed images. The feature maps are obtained by applying filters on the image termed as convolution filters. The number of feature maps are equal to the number of filters. These filters are in the form of 3x3, 5x5, etc. matrices. The feature maps go through the activation function before feeding to the next layer. ReLU is one of the widely used activation functions.

- *Pooling Layer*: Pooling layer reduces the size of the images by taking maximum pixel value or the average pixel value from a group of pixels. Here the pooling areas or windows do not overlap on image regions. Pooling layers are useful for reducing computational loads.
- *Flatten Layer*: Output nodes from previous layer are taken and are separated or flattened and weights are assigned to these individual nodes. Thus, the array or tensor of nodes is reshaped by flatten layer.
- *Dense Layer*: The output shape of the dense layer is affected by the number of units specified in the code. It is a regular NN layer which simply applies activation and gives output.
- *Dropout Layer*: CNN architecture might undergo overfitting in which the model gets trained specifically to given training data and thus fails on any new data. Dropout is the solution to avoid overfitting in which involves elimination of randomly selected nodes from each iteration learning process.

V. Results

A. Comparison of results obtained for existing CNN architectures

Table I shows the results of the transfer learning approach performed with 10 class dataset. The raw image dataset was fed to CNNs. The weights taken were learnt on ImageNet database. The last output layer of the MobileNetV2 was modified for our data. The massive difference between training and validation accuracy implies the overfitting. Fig. 3 shows the training graph for the MobileNetV2.

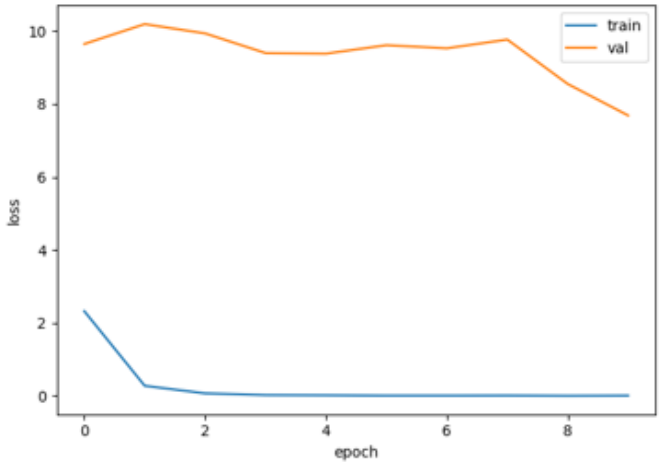

Figure 3. Overfitting with MobileNetV2

Overfitting refers to the model trained perfectly on the 'training' subset of the data. It fails to accurately generalize the test data. Thus, though the training accuracy has reached almost unity, the model fails on the validation dataset.

Thus, after seeing the failure with the above approach, we decided to train some CNN architectures on our dataset from scratch. The weights were randomly initialized. Table I also shows the results of training Vgg16 and LeNet-5.

Features of Vgg16 [12]:

- It uses only small 3x3 filters with stride of 1.
- There are a total of 16 layers with a stack of convolution and max-pooling layers and 3 fully-connected layers and softmax-layer at the end.
- There are approximately 138 parameters to be trained and a huge amount of memory is required (96 MB/image).

On the other hand, LeNet-5 has only 5 layers [13]. Thus, as seen in Table I, both Vgg16 and LeNet-5 architecture is not suitable for our data. Thus, a conclusion can be drawn regarding the depth of required CNN architecture. It should have moderate depth in between Vgg16 and LeNet-5.

TABLE I. RESULTS FOR OTHER CNN MODELS

Architecture	MobileNetV2			Vgg16	LeNet-5
Validation split	0.2	0.2	0.3	0.2	0.2
Optimizer	Adam	SGD(LR =0.01)	SGD(LR =0.01)	SGD(LR =0.01)	SGD(LR =0.01)
Batch size	128	128	32	64	64
Number of epochs	5	5	5	5	5
Training Accuracy	99.64%	98.97%	88.78%	10.55%	10.18%
Validation Accuracy	23.30%	8.98%	9.22%	11.65%	8.98%

B. *Architecture of designed CNN Model*

A model having moderate number of layers is designed. The final architecture that worked better on the data is shown in Fig. 4. The CNN has 10 layers with 3 convolution layers, 3 max-pooling layers, flatten, dropout and 2 dense layers. The use of dropout layer eliminates or skips some randomly selected nodes to avoid overfitting. Generally, a small dropout value of 20%-50% of neurons is used for experimental purposes. Table II shows the training and validation accuracies with different dropout values.

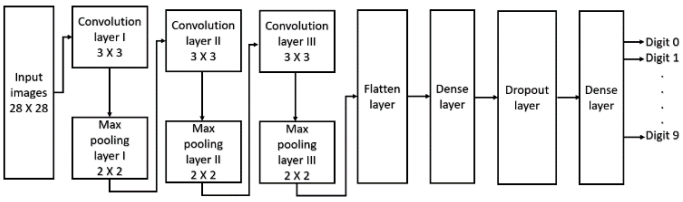

Figure 4. Selected CNN Architecture with 10 Layers

TABLE II. RESULTS FOR OUR CNN MODEL[a]

Dropout value	0.2	0.3	0.4	0.5
Training Accuracy (%)	93.4	95.82	94.19	95.1
Validation Accuracy (%)	88.14	90.31	89.1	91.28

Settings: Validation split = 0.2, optimizer = SGD(LR=0.01), batch size = 64, the number of epochs = 20

Thus, we see the validation accuracy remains in the range of 90 (±2%).

As of now, we have not kept out the test dataset and have not evaluated the test accuracy for any of the models. The training data is a group of images actually sent through the CNN to train the weights and biases of the model. Validation data is another section of the whole dataset used to verify the accuracy of the model. Thus, after each iteration of training, the trained model is evaluated against this validation dataset. Test data is used to evaluate the entire model and it is never touched during the training process. This data gives the final accuracy of the entire CNN architecture.

Thus, we took out the test dataset from the whole data. 20 images per class, i.e. total 200 images were set out for testing the model. This gives a training accuracy of 91.37%, a validation accuracy of 86.30%, and a testing accuracy of 87.50% as shown in Fig. 5. The other settings are the same as above.

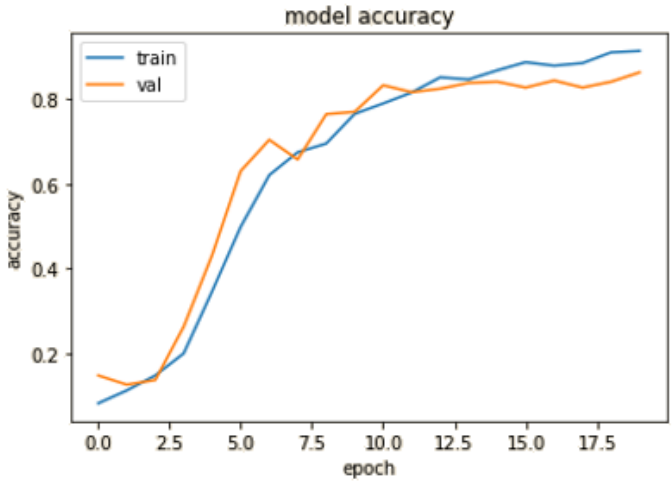

Figure 5. Model Accuracy and Model Loss for Selected CNN Architecture

Thus, a training accuracy of 91.37% and validation accuracy of 86.30% was achieved. Table III shows the obtained confusion matrix.

TABLE III. OBTAINED CONFUSION MATRIX

		Predicted labels									
		0	1	2	3	4	5	6	7	8	9
True labels	0	20	-	-	-	-	-	-	-	-	-
	1	1	19	-	-	-	-	-	-	-	-
	2	-	2	17	-	-	-	-	1	-	-
	3	-	1	2	17	-	-	-	-	-	-
	4	1	-	-	-	14	3	1	-	-	1
	5	-	-	-	-	2	18	-	-	-	-
	6	-	-	-	-	1	-	19	-	-	-
	7	-	1	-	-	1	-	1	16	1	-
	8	-	-	-	-	-	-	-	1	18	1
	9	-	-	-	-	1	-	-	-	2	17

VI. CONCLUSION

The transfer learning approach with ImageNet weights is not suitable for the American Sign Language (ASL) data used in the experiments. Deeper as well as smaller networks do not give satisfying results and thus a CNN architecture with moderate depth needs to be implemented foe this specific dataset. Selected CNN architecture gives recognition test accuracy of 87.5%. The image preprocessing on the database helps reduce the computational and storage complexity. A good set of training hyperparameters is found by manually checking the performance of the model. Such a model will be useful in mobile

applications and embedded systems, increasing recognition speed while keeping the size and complexity of the neural network model at an acceptable level.

REFERENCES

[1] "Deafness and hearing loss," *World Health Organization*. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss. [Accessed: 12-Feb-2021].

[2] "American Sign Language," *National Institute of Deafness and Other Communication Disorders*, 14-Dec-2020. [Online]. Available: https://www.nidcd.nih.gov/health/american-sign-language. [Accessed: 12-Feb-2021].

[3] M. M. Islam, S. Siddiqua and J. Afnan, "Real time Hand Gesture Recognition using different algorithms based on American Sign Language," *2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, Dhaka, Bangladesh, pp. 1-6, 2017.

[4] K. Bantupalli and Y. Xie, "American Sign Language Recognition using Deep Learning and Computer Vision," *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, pp. 4896-4899, 2018.

[5] Sruthi, C. J., and A. Lijiya. "Signet: A deep learning based indian sign language recognition system." *2019 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0596-0600. IEEE, 2019.

[6] Rao, G. Anantha, K. Syamala, P. V. V. Kishore, and A. S. C. S. Sastry. "Deep convolutional neural networks for sign language recognition." *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)*, pp. 194-197. IEEE, 2018.

[7] Shahriar, Shadman, Ashraf Siddiquee, Tanveerul Islam, Abesh Ghosh, Rajat Chakraborty, Asir Intisar Khan, Celia Shahnaz, and Shaikh Anowarul Fattah. "Real-time american sign language recognition using skin segmentation and image category classification with convolutional neural network and deep learning." *TENCON 2018-2018 IEEE Region 10 Conference*, pp. 1168-1171. IEEE, 2018.

[8] Lin, Hsien-I., Ming-Hsiang Hsu, and Wei-Kai Chen. "Human hand gesture recognition using a convolution neural network." *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1038-1043. IEEE, 2014.

[9] 'ASL-100-RGBD Dataset, Media Lab, The City College, City University of New York': http://media-lab.ccny.cuny.edu/wordpress/datecode/#Dataset

[10] 'American Sign Language dataset, Turkey Ankara Ayrancı Anadolu High School', https://github.com/ardamavi/Sign-Language-Digits-Dataset

[11] 'Canny Edge Detection using OpenCV', https://docs.opencv.org/master/da/d22/tutorial_py_canny.html

[12] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[13] LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.