Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.027848
Article

Tech Science Press

A Light-Weight Deep Learning-Based Architecture for Sign Language Classification

M. Daniel Nareshkumar[1,*] and B. Jaison[2]

[1]Department of Electronics and Communication Engineering, R.M.K. Engineering College, Kavaraipettai, 601206, India
[2]Department of Computer Science and Engineering, R.M.K. Engineering College, Kavaraipettai, 601206, India
*Corresponding Author: M. Daniel Nareshkumar. Email: mnr.ece@rmkec.ac.in
Received: 27 January 2022; Accepted: 10 May 2022

Abstract: With advancements in computing powers and the overall quality of images captured on everyday cameras, a much wider range of possibilities has opened in various scenarios. This fact has several implications for deaf and dumb people as they have a chance to communicate with a greater number of people much easier. More than ever before, there is a plethora of info about sign language usage in the real world. Sign languages, and by extension the datasets available, are of two forms, isolated sign language and continuous sign language. The main difference between the two types is that in isolated sign language, the hand signs cover individual letters of the alphabet. In continuous sign language, entire words' hand signs are used. This paper will explore a novel deep learning architecture that will use recently published large pre-trained image models to quickly and accurately recognize the alphabets in the American Sign Language (ASL). The study will focus on isolated sign language to demonstrate that it is possible to achieve a high level of classification accuracy on the data, thereby showing that interpreters can be implemented in the real world. The newly proposed Mobile-NetV2 architecture serves as the backbone of this study. It is designed to run on end devices like mobile phones and infer signals (what does it infer) from images in a relatively short amount of time. With the proposed architecture in this paper, the classification accuracy of 98.77% in the Indian Sign Language (ISL) and American Sign Language (ASL) is achieved, outperforming the existing state-of-the-art systems.

Keywords: Deep learning; machine learning; classification; filters; american sign language

1 Introduction

By virtue of the solution, all sign languages are heavily dependent on the motion of the body in specific ways. Communication for deaf and mute people is not limited solely to sign languages but also includes other forms of bodily expressions such as facial expressions and gestures. While most people can pick up facial expressions and gestures, the actual brunt of the conversation is typically conducted via the use of sign languages, which, as they imply, are unique set of languages.

Like other languages, Sign languages are not universal throughout the world instead in most cases, each country has its version of the language. And similarly to regular languages, sign languages are also affected by the fact that one must know the language in detail for effective communication,. While in the longer run, learning the actual sign languages will prove to be a better solution for effective communication. This work explores a computer vision-based solution that can quickly interpret sign language and help in communication in all this regard.

Unlike verbal communication, all sign languages can be split into two major types, isolated sign language and continuous sign language. Isolated sign language mainly encompasses individual hand motions that are usually used to spell out words letter by letter. On the other hand, continuous sign language hand motions are primarily used to describe word-level meanings. This form of sign language usually connects the hand motions to form sentences that are found in verbal languages.

This work will focus on the latter, isolated sign languages, and will attempt to build a system that can interpret the different alphabets in the American Sign Language (ASL) Alphabet System, Fig. 1.

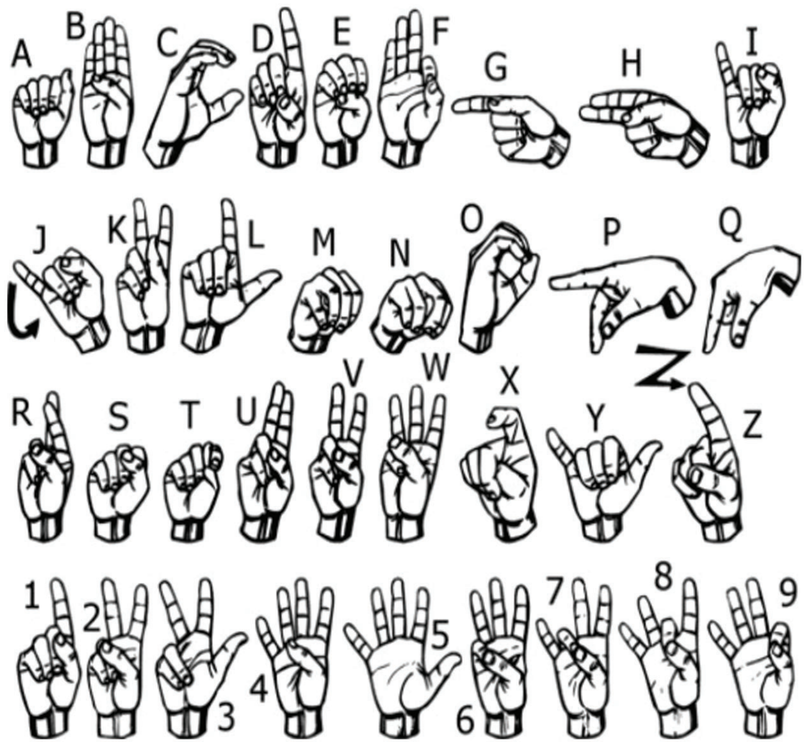


Figure 1: American sign language alphabet system [1]

1.1 Contributions of this Paper

a) Implemented a system specifically focused on the classification of isolated sign language, which focuses on alphabet hand signs. The proposed system can interpret different alphabets in the two-alphabet systems, namely, i)American Sign Language (ASL) Alphabet System and ii) Indian Sign Language alphabet system (Fig. 2).

b) Utilizing pre-trained models, we can obtain valuable results while maintaining low train time.

c) Utilized modified MobileNetV2 architecture to provide the shortest possible inference time on the end system.

d) Showcases modifications to the MobileNetV2 architecture that improve classification accuracy compared to other works that also utilize the MobileNetV2 architecture.
e) Developed an architecture that outperformed state-of-the-art techniques with a classification accuracy of 98.77 percent.

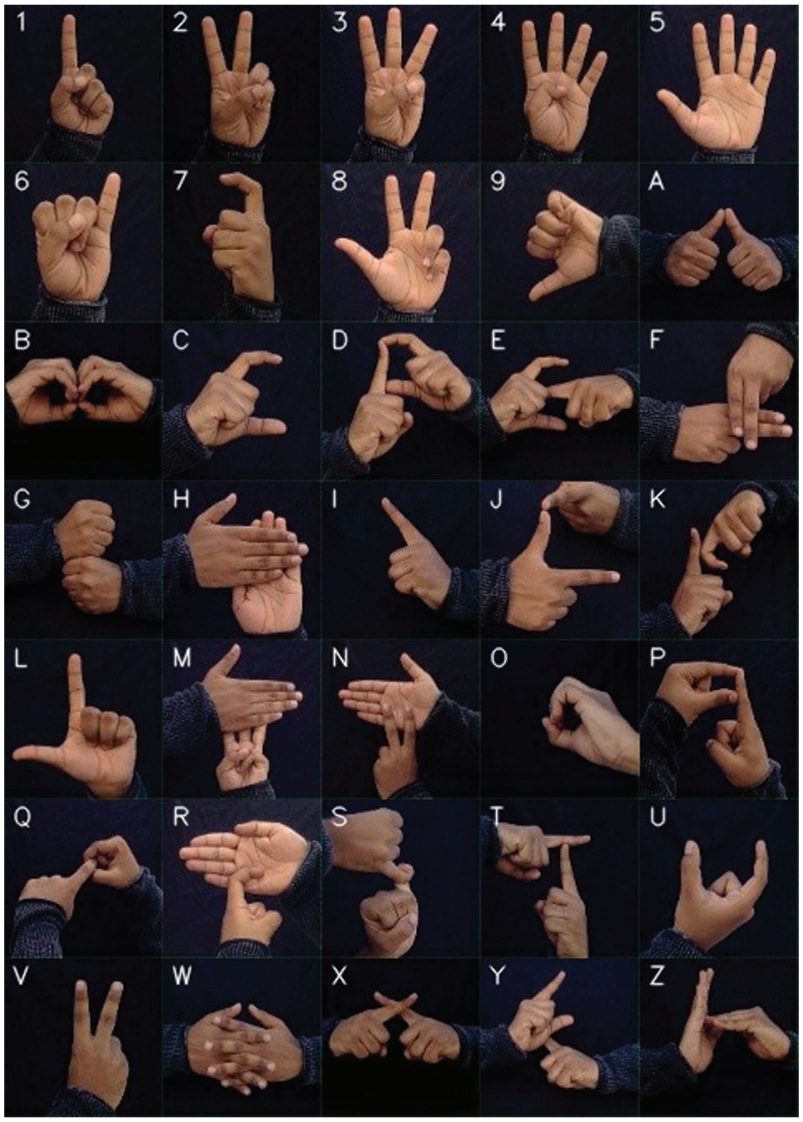


Figure 2: Indian sign language alphabet system [2]

2 Literature Survey

This section covers various existing state-of-the-art sign language interpretation architectures.

2.1 Problem Formulation

When it comes to problem formulation for isolated sign language interpretation, two ways are commonly employed. The first is to use a static image that is captured using a camera and then extract features from those images to be used as the input for the deep learning model. The next step is to develop a three-dimensional model using specialized equipment that allows creating an accurate representation of the hand. The second method of generating a hand model does not translate very efficiently to real-world use due to the extensive setup needed before the model can be given the input during inference. Thus, this work will only follow the first methodology of using a static image. However, this literature survey section will only cover architecture and methodologies specific to both methods.

2.2 *Different Architectures Used*

The works that revolve around the usage of static images can be further broken down into two categories-Machine learning-based and Deep learning-based. The Machine learning-based architectures usually involve manual feature extraction, which was then passed to machine learning classification architectures such as the k-NN architecture. On the other hand, the CNN-based models include both custom CNN models and fine-tuning large pre-trained models (Tripathy˙MS˙2021). This section will be covering both of these methodologies, even though the initial machine learning models reported lower results.

2.2.1 Machine Learning Strategies

One of the earliest approaches by Pugeault et al. [3] used Gabor filters to extract features, which were then used to train multi-class random forests to develop a classifier for 24 letters of ASL.

Another attempt was made with regard to sign language classification of alphabets in the ASL by Safaya et al. [4]. In this work, the main method for obtaining the features from the hand is the use of a specialized camera system-Dynamic Vision Sensor (DVS). The DVS camera system can detect the temporal pixel luminance difference, from which features regarding the hand shape are obtained. This methodology cannot provide any substantial results.

Wathugala et al. [5] pursued another approach using a new feature vector approach recognize two-dimensional hand configuration for interpreting Sinhala fingerspelling. This new feature vector was later used to input to a modified nearest neighbor algorithm. On unseen data, the system was able to yield 62 percent.

Similar to the work done by Wathugala et al. [5], Ewald et al. [6] also used the nearest neighbor algorithm for the classification. However, they deviated by creating a hand model from the images that had 20 degrees of freedom. This was then passed through an algorithm to extract the histogram of centroid distances (HOCD) and Gabor filters, Fig. 3. These extracted features were then passed to various classification algorithms like KNN and SVM. With this methodology, they obtained an accuracy of 80% on unseen hand data.

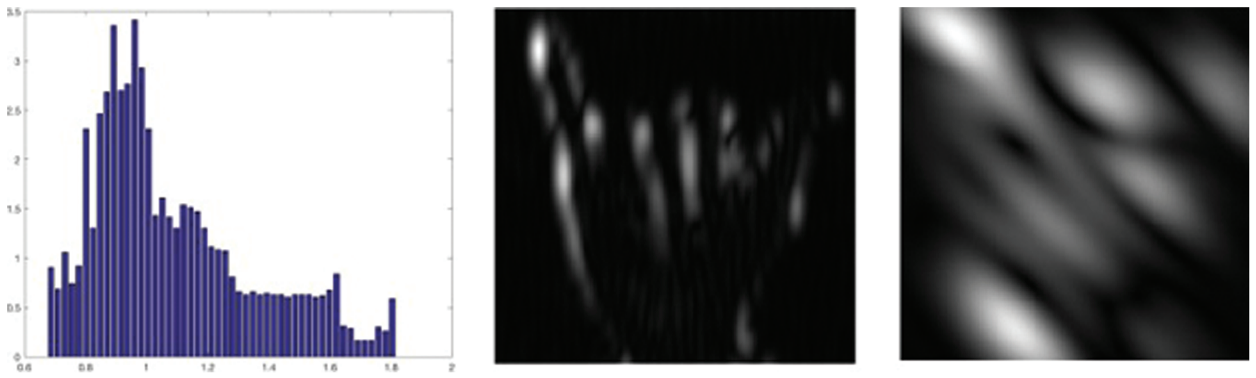


Figure 3: HOCD and gabor filters, sourced from [6]

While several other works make sole use of machine learning architectures, the overall results from these methodologies were not very impressive. The average results range from around 60% to 80%, leading to highly undependable classification in actual real-world use. Another issue with machine learning models was that the images of the hand obtained from the camera required several preprocessing stages to extract features that could be fed into the machine learning models. This preprocessing procedure was often quite time-intensive, causing the model's deployment in a real-world scenario extremely slow.

2.2.2 Custom CNN Models

Ameen et al. [7] extracted picture and depth information from the static image input and performed feature extraction on each of them using two convolution layers. After the feature extraction from the first stage, the feature maps were combined in the second step of convolution, which involved a pooling layer. Precision and recall were reported to be 82% and 80%, respectively.

Pigou et al. [8] utilized a similar framework to segregate hand and upper body information. They used the Microsoft Kinect framework to obtain more information than what could be obtained from using a static image such as the depth map and the 3D skeleton, Sethuraman et al. [9]. Like the previous work, this work also used a CNN-only network with a depth of 3 and the only pooling they utilized was max-pooling after the second and third layers of CNN, Fig. 4. With this approach, Pigou et al. [8] achieved an accuracy of 91.7%.

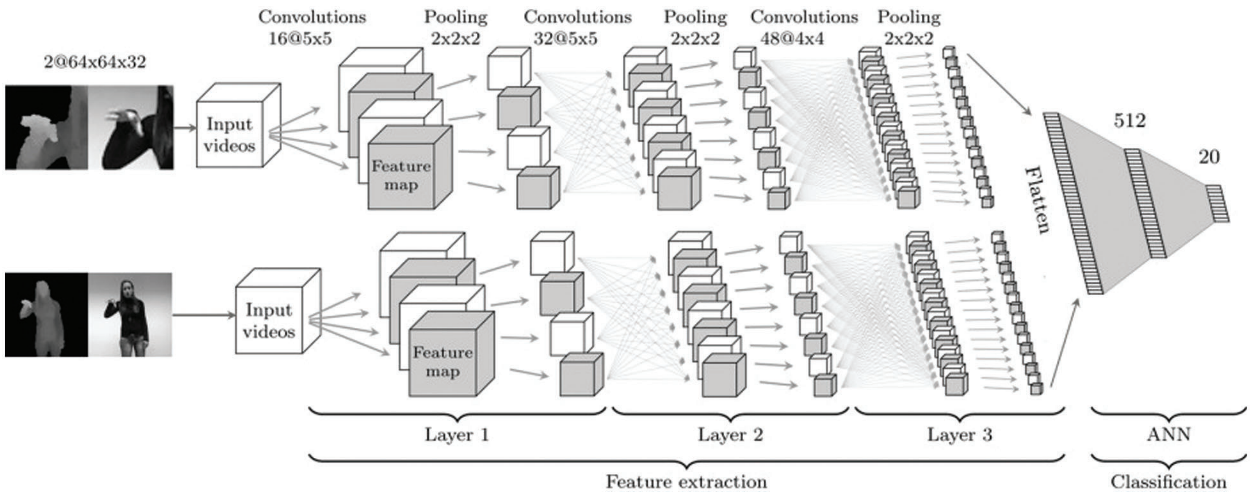


Figure 4: Three-layer architecture implemented, sourced from [8]

Bheda et al. [10] used a deep convolutional network to classify ASL with alphabets and digits. However, unlike the previous two approaches, this work used cascaded CNN rather than the traditional CNN. The main difference between a regular CNN and the cascaded CNN is that the architecture of cascaded CNN shares the hidden layer weights across several layers. This allows the model to get trained much more quickly while still not destabilized. Their proposed network had 3 cascaded convolutional layers and 1 max-pooling layer. There are two hidden layers with dropout before connecting to the output layer. Using this procedure, an accuracy of 82.5 percent was achieved.

2.2.3 Fine Tuned Models

Unlike the previous two types of models, transfer learning Tripathy et al. [11] and Kompally et al. [12] or fine-tuning pre-trained models come with the advantage that the model does not have to be trained from scratch. Any large pre-trained image model can be fine-tuned; these models are often exceptionally deep CNN models that have been trained on massive datasets like the ImageNet dataset or the

Places365 dataset. In the work by Alashhab et al. [13] several different large pre-trained models were used on the same dataset to test which model performed the best. The models used were-VGG16, VGG19, ResNet, Xception, InceptionV3, MobileNet and SqueezeNet. The input to all these models were 5 classes of hand gestures in the sign languages, and since only 5 classes were used instead of all the alphabets in the system, this work reported high scores for all the fine-tuned models. Another work carried out by Das et al. [14] was also based on the fine-tuning of the InceptionV3 model. However, in their work, they used custom-processed static images and achieved an average validation accuracy of 90%.

In most cases, fine-tuning a pre-trained model for sign language classification provides a better result, but in the work done by Bousbai et al. [15] their custom CNN model was able to outperform the fine-tuned model. Their custom CNN model could get ahead of the fine-tuned model by scoring 98.9% over the 97.06% of the fine-tuned model. The input data that was provided to both the models were from the same source and were preprocessed the same way.

Like in the previous work, Garcia et al. [16] were unable to get high scores using a pre-trained model. Using the GoogLeNet model for classifying the 24 classes of sign languages only resulted in 70% accuracy.

In general, when it comes to the sign language classification tasks, transfer learning will mostly outperform many custom CNN models [17–19]. Typically, a custom model is required to accommodate additional data, such as depth information from the Microsoft Kinect. However, transfer learning models built on pure image datasets, such as ImageNet, may not perform as well as specialized models with an extensive preprocessing pipeline [20–22]. Apart from that, transfer learning is a better method for developing a high-performing model [23,24].

3 Proposed Architecture

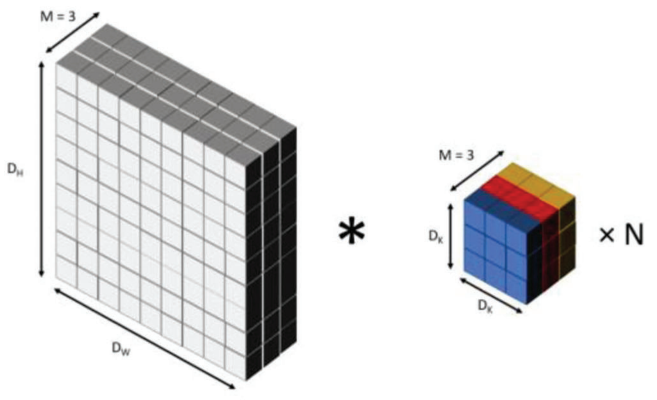
This section discusses the proposed architecture in terms of design and implementation. It will cover the motivation of using MobileNet over other large pretrained architectures, showcase the simple yet effective preprocessing strategy used in the pipeline, and discuss the proposed architecture in depth.

As we have seen from the previous section, several different methods can be employed to classifying sign language. However, while several methods exist, most of the methods used have some form of caveat attached to them. With custom models, you have the issue of having to manually preprocess in a specific manner to extract the features. On the other hand, certain large pre-trained models are not suitable for this task, either because the underlying architecture fails to detect hand features or because they are too large and take too long to infer. The pretrained model solution given by other works has produced extremely good results. However, they come at the cost of not being implemented in a real-life scenario due to the inference time.

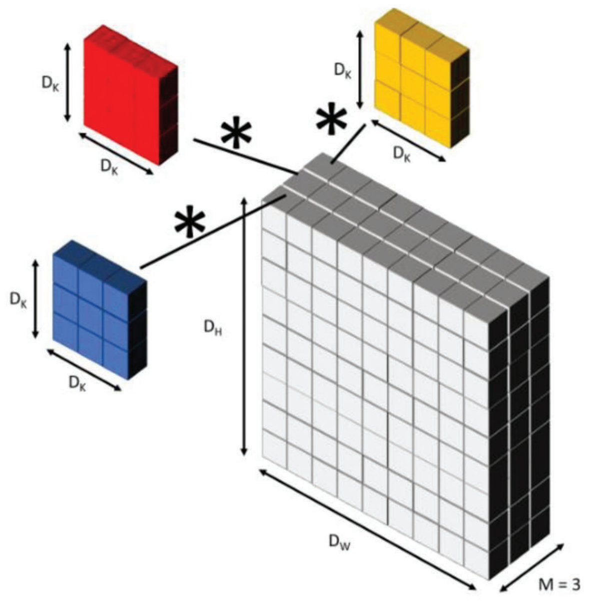
3.1 Motivation Behind MobileNetV2

The primary motivation for using MobileNetV2 architecture is to run the proposed model on mobile devices, robots, and other forms of self-driving cars [25,26]. And so, the MobileNetV2 architecture is optimized to have very low latency in processing the images while running on low hardware resources. In the use case of sign language classification, having this low latency for processing the images allows the entire architecture to produce inference quickly, making the whole architecture deployable in a real-world scenario [27,28].

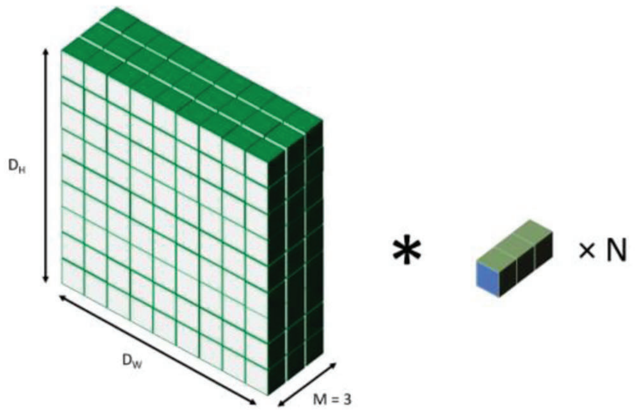
While other large pre-trained models that also try to reduce latency do so by teacher learning or by shrinking the overall depth of the model, MobileNetV2 works by modifying the underlying CNN architecture [29,30]. The CNN architecture in MobileNetV2 utilizes a rebuilt CNN architecture that uses separable depth-wise convolutions [31,32]. These depth-wise convolutions factorize the normal convolutions into two separate operations–depth-wise convolution and point-wise convolution. By breaking down the normal convolution into two different functions, MobileNetV2 reduces the computational cost and overall model size, Fig. 5.



(a) StandardConvolution



(b) Depth Wiseconvolution



(c) Point WiseConvolution

Figure 5: Comparison of standard convolution *vs.* mobilenetv2

The standard convolutional layers work using N square and K filters amounting to DK × DK × M (DW and DH denote the width and the height of the image and M denotes the depth of the image) when applied to the whole image I. So, with this setup, the standard convolutional layer will have the computation shown in the Eq. (1)

$$J_{standard} = N[(D_W \times D_H) \times (D_K \times D_K \times M)] \tag{1}$$

As mentioned above, MobileNetV2 splits this operation into two parts called separable depth-wise convolution. Instead of using the kernel on all the layers in M, the kernel is applied to a single layer at a time, as we can see in Eq. (2). Then the point-wise convolution is applied to the resulting output from the depth-wise convolution. The point-wise convolution uses a 1×1 kernel with a depth of M, N times as shown in Eq. (3).

$$J_{depth} = M[(D_W \times D_H) \times (D_K \times D_K)] \tag{2}$$

$$J_{point} = N(M \times D_W \times D_H) \tag{3}$$

The overall computational cost of MobileNetV2 ends up being the sum of the two individual operations,

$$JMobileNetV2 = Jdepth + Jpoint \tag{4}$$

When comparing to the cost of the standard convolution, a reduction factor of almost 8 to 9 times, Eq. (5), with only a marginal decrease in the overall accuracy, exists.

$$\frac{J_{MobileNetV2}}{j_{standard}} = \frac{1}{N} + \frac{1}{D_K^2} \tag{5}$$

MobileNetV2 also resolves the issue with MobileNetV1 in which values less than zero were ignored due to the ReLU activation algorithm. When the number of channels M was minimal, this resulted in a significant loss of data because all of the information obtained from the images was incredibly useful. This issue was rectified by implementing a new strategy called inverted residual block, which had the structure going from narrow to wide to narrow again, Fig. 6.

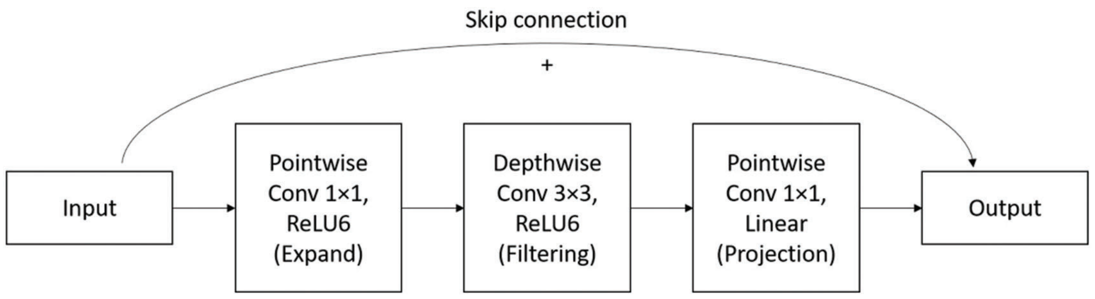

Skip connection
+
Input
Pointwise Conv 1×1, ReLU6 (Expand)
Depthwise Conv 3×3, ReLU6 (Filtering)
Pointwise Conv 1×1, Linear (Projection)
Output

Figure 6: Linear bottleneck architecture

Using the point-wise 1×1 convolution, the image's dimension was expanded before passing it to the depth-wise convolution having 3×3 filters. Since the dimensionality of the overall feature map to the ReLU in the depth-wise convolution was increased, it bypasses the information loss that could occur due to the low depth of the image. Once passed through the depth-wise filter, the dimensionality was brought back to the original by passing the output through a1×1 point-wise convolution again. To ensure that loss of data does not occur, a skip connection was also established between the first and the last layers in the block. This structure was called the "linear bottleneck" by the original authors of the MobileNetV2 architecture.

3.2 Proposed Architecture

Fig. 7 shows how the constructed environment work. The MobileNetV2 model, which was trained with the ImageNet dataset, was used for fine-tuning, and the MobileNetV2 model itself was frozen so that the weight would not change while training. Unlike the original architecture of MobileNetV2, this work utilizes MobileNetV2 with an average pooling layer rather than the max-pooling layer to obtain the MobileNetV2 outputs. After that, a dense layer with 128 neurons was added to allow room for fitting into the sign language classification data. This dense network used a standard ReLU as its activation function. The output from the dense layer was then passed on to a 0.2 dropout layer to avoid overfitting the data into the model. The last layer is a dense layer with a softmax activation function, responsible for classifying photos into 29 distinct groups.

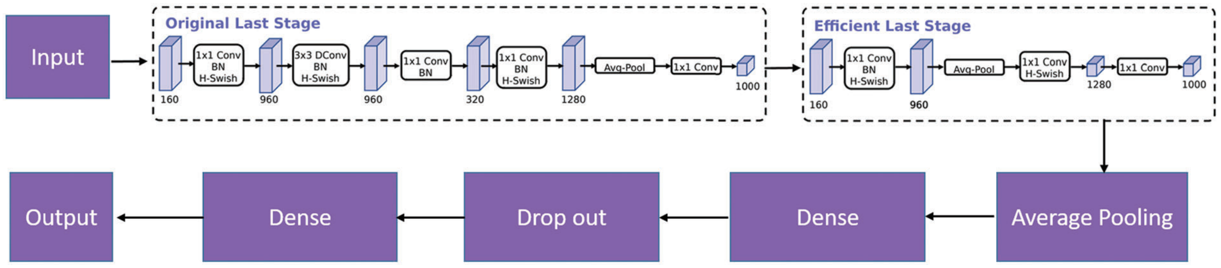


Figure 7: Proposed architecture-signlan-net (Modified MobileNetV2)

4 Experimental Setup and Performance Evaluation

4.1 Dataset

The dataset used in this work is the Indian Sign Language Dataset [2] and the ASL Alphabet data [1], which are publicly available in Kaggle repositories. The training data considered are 87,000 images, which are 200 × 200 pixels and 29 classes, of which 26 are for the letters A-Z and 3 classes for SPACE, DELETE, and NOTHING. Each of the 29 classes present in the dataset has 3060 images each, which allow the use of this dataset without any prior data balancing between classes.

4.2 Computational Setup

Even though such a vast stack will not be required for inference, the computational hardware used in this study is quite remarkable. In the training phase, the model was fitted using an Nvidia Tesla A100 GPU, which has 40GB of vRAM running in Ubuntu Server with 200GB of RAM and an Intel Xeon processor. To evaluate the efficacy of the proposed model to run in resource-constrained compatible devices, we validated the setup using JetsonNano GPU connected with Raspberry Pi running Raspbian OS. With this setup, the design at its peak load utilized about 85% of the RAM making the training a much simpler hardware system like that found on mobile phones. However, it is worth noting that the needs for the training are much higher than what will be required for inference in the final stage. Thus, these numbers are once again misleading when considering a real-life use case.

4.3 Preprocessing

The preprocessing was maintained minimal to demonstrate that it is indeed possible to build a system capable of sign language classification with a high level of accuracy retaining nominal inference time. The images present in the dataset were already of high enough quality; thus, no upscaling of the images was done before passing them into the model. To ensure that the hand itself was clear in the images, all the images in the training and the validation dataset were brightened to about 30% more than that of the original image. By brightening the image, it is ensured that there is enough contrast between the hand and the background so that the model can adequately pick out the hand shape. After this, the images were also randomly flipped horizontally to ensure that the model can classify images regardless of which hand is used while signing in sign language. An example of the preprocessed images is shown in Fig. 8.

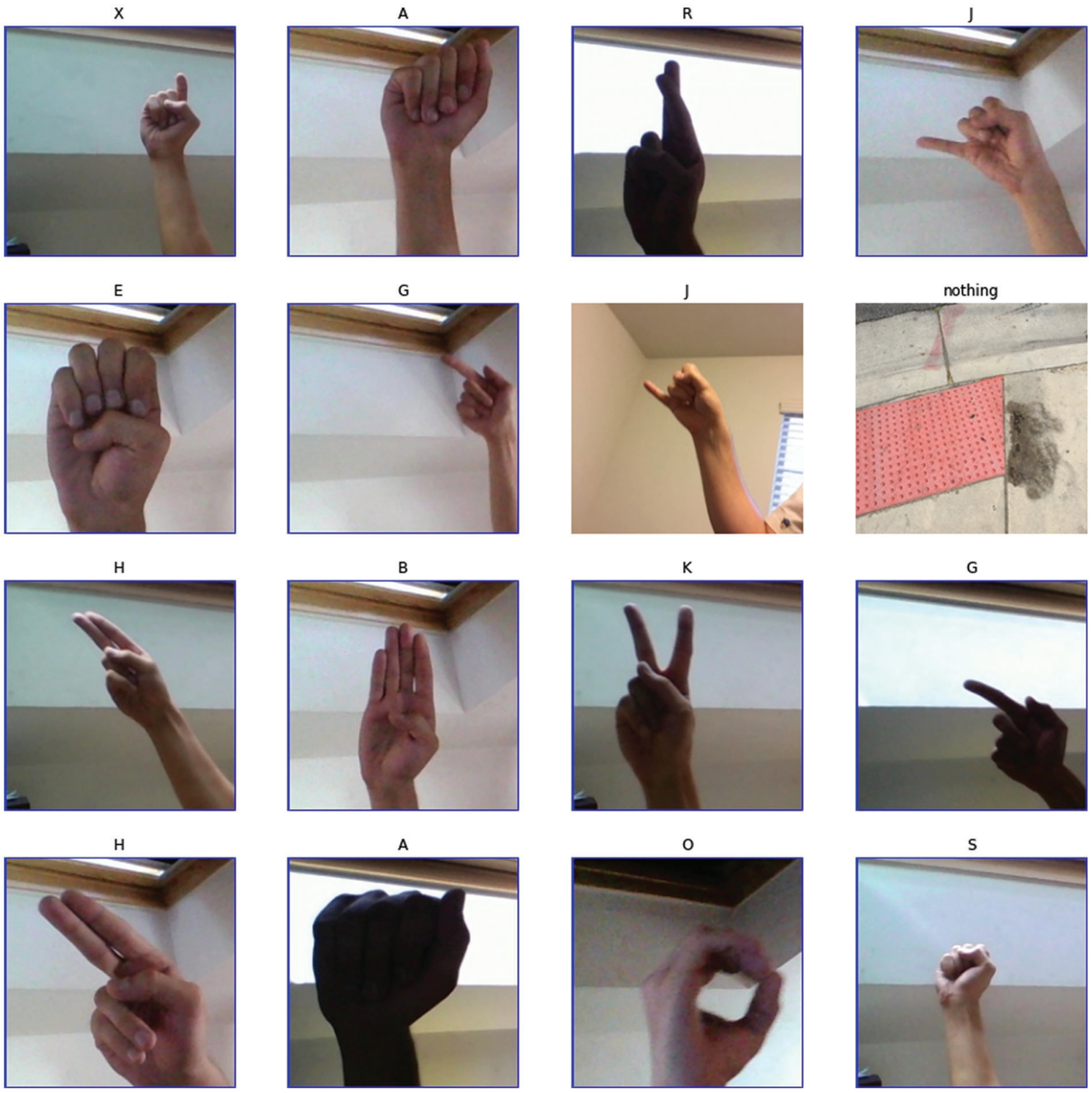
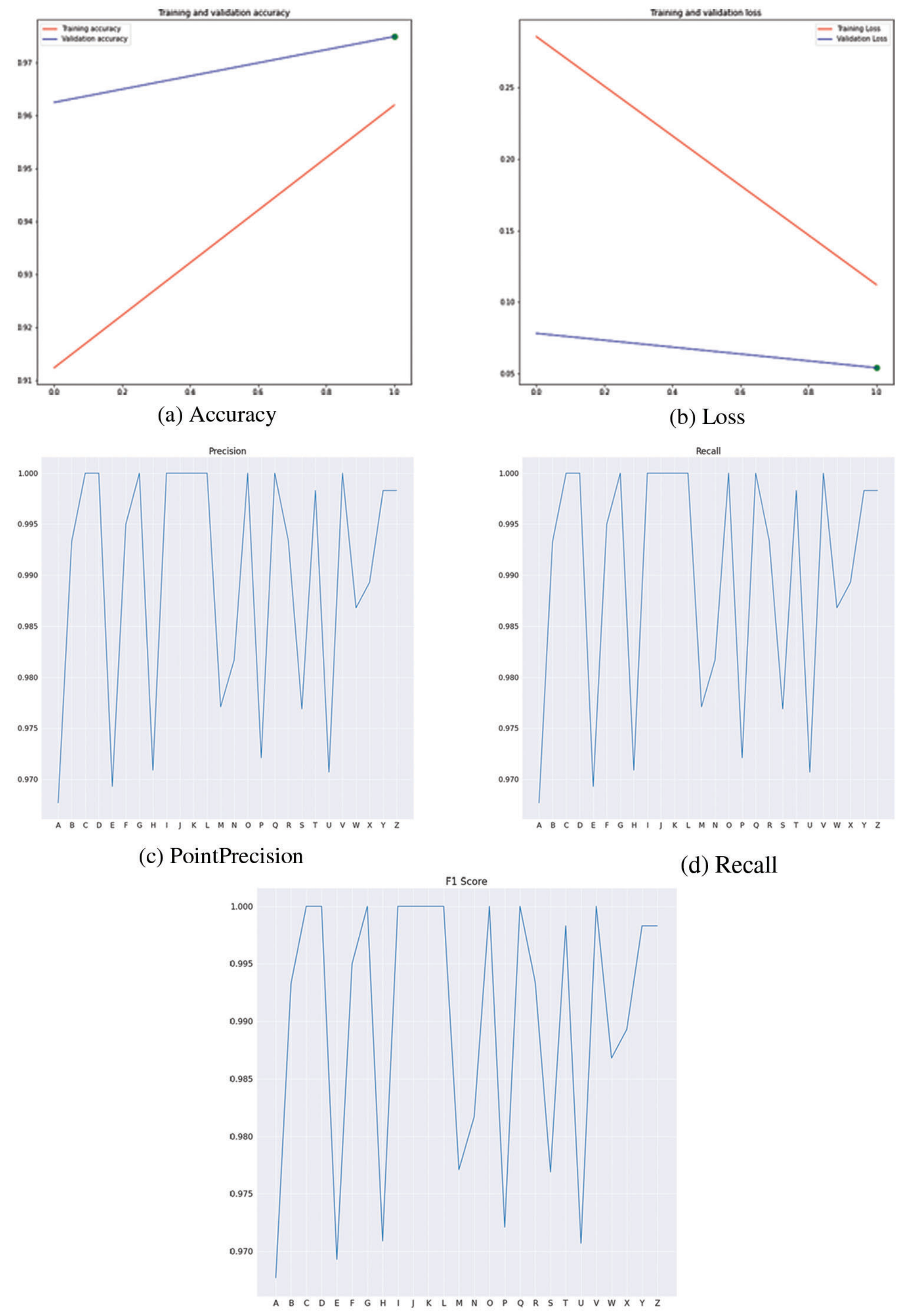


Figure 8: Preprocessed train images

The model was trained for 10 epochs in total and it was experimentally evident that training it any more caused severe overfitting even though the drop-out layer was included. The proposed model was able to achieve an overall accuracy of 98.77%, with the highest-class-wise accuracy of 100%, for multiple classes. The lowest class-wise accuracy was obtained for the class of P, where the accuracy was 84.89%. However, while looking at the loss and the accuracy graphs, Fig. 9, we can see that the fine-tuning of the model was quite stable. Hence it is possible to infer that the loss of accuracy in certain classes such as P is most likely due to the data from the low-quality datasets as compared to the other classes.



(a) Accuracy (b) Loss

(c) PointPrecision (d) Recall

(e) F1 Score

Figure 9: Metric graphs

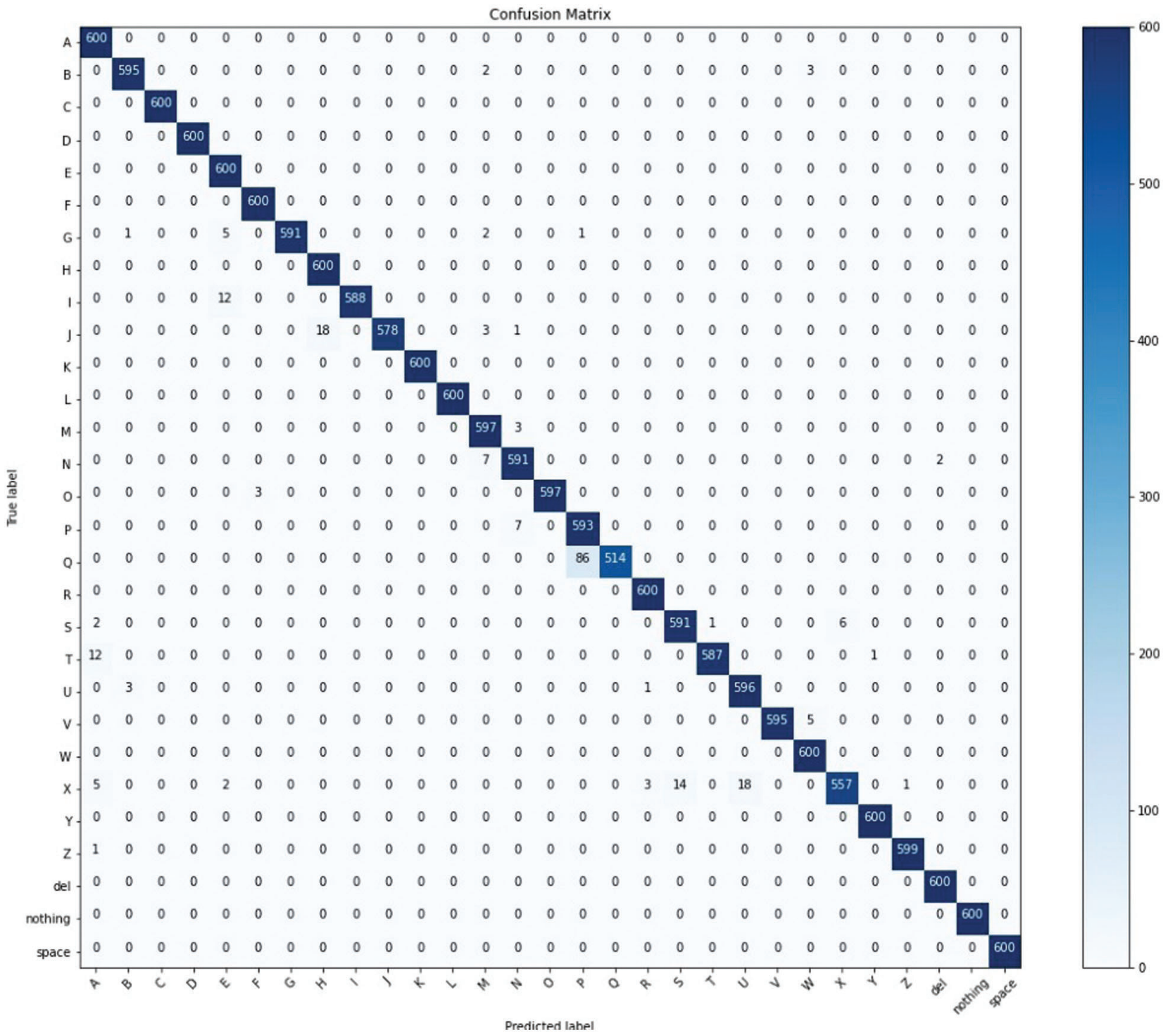
Tab. 1 shows the comparison of this work with a few other works that utilized pre-trained models. From this table, it is transparent that only the work conducted by Alashhab et al. [13] was able to obtain a higher accuracy as proposed in this paper. However, it is very likely that the cause for this is because they used a dataset that only had 5 classes for classification. We can also see from the table that the work done by Bousbai et al. [15] using MobileNetV2 performed poorer as compared to the proposed architecture. Since the pre-trained architecture is used for the majority of the implementation between their work and this work, the suggested architecture's higher accuracy is most likely due to the overfitting being minimized by the usage of the drop out layer.

Table 1: Comparison of different works against proposed architecture

Work	Model	Accuracy
Das et al. [14]	Inception V3	90.0%
Alashhab et al. [13]	MobileNetV1	94.5%
Garcia et al. [16]	GoogLeNet	70%
Bousbai et al. [15]	MobileNetV2	97.06%
Proposed architecture	SignLan-Net (Modified MobileNetV2)	98.77%

This model obtained the highest overall accuracy of 98.67%. Fig. 10 depicts the confusion matrix. For each sign, 28 out of 29 were recorded with a precision of greater than 95%. The best individual accuracy was 100%, while the lowest was 84.89%. The letter Q had the lowest level of accuracy. Only 514 photos (out of 600) were properly predicted. The rest of the numbers were all anticipated to be P. The cause was hypothesized to be the resemblance in these two signals, where the index finger and thumb are both pointing out.

Furthermore, the main direction of this work to provide an architecture that is capable of providing inference in a short time was achieved whilst maintaining a high level of accuracy. The architecture is capable of producing inference in under 2.5 s, allowing it to be much more usable when comparing it with other large fine-tuned models and even some custom models that have lower accuracy.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	del	nothing	space
A	600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	595	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
C	0	0	600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	0	0	5	0	591	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	12	0	0	0	588	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	18	0	578	0	0	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	597	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	7	591	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
O	0	0	0	0	0	3	0	0	0	0	0	0	0	0	597	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	593	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	514	0	0	0	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	600	0	0	0	0	0	0	0	0	0	0	0
S	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	591	1	0	0	0	6	0	0	0	0	0
T	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	587	0	0	0	0	1	0	0	0	0
U	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	596	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	595	5	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	600	0	0	0	0	0	0
X	5	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	3	14	0	18	0	0	557	0	1	0	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	600	0	0	0	0
Z	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	599	0	0	0
del	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	600	0	0
nothing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	600	0
space	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	600

Figure 10: Confusion matrix plotted by the proposed model

5 Conclusion

This paper presents a light weight deep learning architecture for the use cases of classifying sign language alphabets from the ISL and ASL systems. The proposed model is built on top of the MobileNetV2 architecture, which was specifically designed to have very low latency while performing any inference. By fine-tuning using the MobileNetV2 architecture, the architecture ends up being a very lightweight architecture. The suggested architecture retains high levels of accuracy (98.77 percent) and can provide inference outputs in under 2.5 s, making it a far more practical model for real-world application.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Akash Asl alphabet, Apr. 2018. [Online]. Available: https://www.kaggle.com/grassknoted/asl-alphabet.

[2] R. Elakkiya and B. Natarajan, “Isl-csltr: Indian sign language dataset for continuous sign language translation and recognition,” Jan. 2021. [Online]. Available: https://data.mendeley.com/datasets/kcmpdxky7p/1.

[3] N. Pugeault and R. Bowden, “Spelling it out: Real-time asl fingerspelling recognition,” in *2011 IEEE Int. Conf. on Computer Vision Workshops (ICCV Workshops)*, IEEE, Barcelona, Spain, pp. 1114–1119, 2011.

[4] K. Safaya and P. Bakal, “Real time-based bare hand gesture recognition,” *IPASJ International Journal of Information Technology*, vol. 1, no. 2, pp. 1–9, 2013.

[5] D. M. Wathugala and N. Kodikara, “A sinhala finger spelling interpretation system using nearest neighbor classification,” in *Proc. of 4th Int. Information Technology Conf. Envisioning an eNation*, Colombo, Sri Lanka, pp. 72–78, 2002.

[6] H. M. Ewald, I. Patil and S. Ranmuthu, “Asl fingerspelling interpretation,” University of Stanford, Reports, 2016.

[7] 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, pp. e12197, 2017.

[8] L. Pigou, S. Dieleman, P. J. Kindermans and B. Schrauwen, “Sign language recognition using convolutional neural networks,” in *European Conf. on Computer Vision*, Switzerland, Springer, pp. 572–578, 2014.

[9] S. C. Sethuraman, P. Kompally and S. Reddy, “Visu: A 3-d printed functional robot for crowd surveillance,” *IEEE Consumer Electronics Magazine*, vol. 10, no. 1, pp. 17–23, 2021.

[10] V. Bheda and D. Radpour, “Using deep convolutional networks for gesture recognition in American sign language,” arXiv preprint arXiv:1710.06836, 2017.

[11] J. K. Tripathy, S. C. Sethuraman and M. V. Cruz., “Comprehensive analysis of embed-dings and pre-training in nlp,” *Computer Science Review*, vol. 42, no. 1, pp. 100433, 2021.

[12] P. Kompally, S. C. Sethuraman, S. Walczak, S. Johnson and M. V. Cruz, “Malang: A decentralized deep learning approach for detecting abusive textual content,” *Applied Sciences*, vol. 11, no. 18, pp. 1–18, 2021.

[13] S. Alashhab, A.-J. Gallego and M. A. Lozano, “Hand gesture detection with convolutional neural networks,” in *Int. Symp. on Distributed Computing and Artificial Intelligence*, Spain, Springer, pp. 45–52, 2018.

[14] A. Das, S. Gawde, K. Suratwala and D. Kalbande, “Sign language recognition using deep learning on custom processed static gesture images,” in *2018 Int. Conf. on Smart City and Emerging Technology (ICSCET)*, IEEE, Mumbai, India, pp. 1–6, 2018.

[15] K. Bousbai and M. Merah, “A comparative study of hand gestures recognition based on mobilenetv2 and convnet models,” in *2019 6th IEEE Int. Conf. on Image and Signal Processing and their Applications (ISPA)*, Algeria, pp. 1–6, 2019.

[16] B. Garcia and S. A. Viesca, “Real-time American sign language recognition with convolutional neural networks,” *Convolutional Neural Networks for Visual Recognition*, vol. 2, no. 2, pp. 225–232, 2016.

[17] M. Prakash and T. Ravichandran, “An efficient resource selection and binding model for job scheduling in grid,” *European Journal of Scientific Research*, vol. 81, no. 4, pp. 450–458, 2012.

[18] P. Mohan and R. Thangavel, “Resource selection in grid environment based on trust evaluation using feedback and performance,” *American Journal of Applied Sciences*, vol. 10, no. 8, pp. 924–930, 2013.

[19] R. Farah Sayeed, S. Princey and S. Priyanka, “Deployment of multicloud environment with avoidance of ddos attack and secured data privacy,” *International Journal of Applied Engineering Research*, vol. 10, no. 9, pp. 8121–8124, 2015..

[20] U. Gowshika, D. Shaloom Immulicate and S. Sathiya Priya, “Analysis of defect in dental using image processing,” *International Journal of Applied Engineering Research*, vol. 10, no. 9, pp. 8125–8129, 2015.

[21] S. Satpathy, S. Debbarma, S. C. Sengupta Aditya and K. D. Bhattacaryya, “Design a fpga, fuzzy based, insolent method for prediction of multi-diseases in rural area,” *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 5, pp. 7039–7046, 2019.

[22] P. V. Rajaram and M. Prakash, “Intelligent deep learning based bidirectional long short term memory model for automated reply of e-mail client prototype,” *Pattern Recognition Letters*, vol. 152, no. 1, pp. 340–347, 2021.

[23] J. Deepak Kumar, B. Prasanthi and J. Venkatesh, “An intelligent cognitive-inspired computing with big data analytics framework for sentiment analysis and classification,” *Information Processing & Management*, vol. 59, no. 1, pp. 1–15, 2022.

[24] A. Arun, R. R. Bhukya, B. M. Hardas and T. Ch, “An automated word embedding with parameter tuned model for web crawling,” *Intelligent Automation & Soft Computing*, vol. 32, no. 3, pp. 1617–1632, 2022.

[25] D. Venu, A. V. R. Mayuri, G. L. N. Murthy, N. Arulkumar and S. Nilesh, “An efficient low complexity compression based optimal homomorphic encryption for secure fiber optic communication,” *Optik*, vol. 252, no. 1, pp. 168545, 2022.

[26] S. Harinder, D. Ramya, R. Saravanakumar, S. Nayani, R. Anand *et al.,* “Artificial intelligence based quality of transmission predictive model for cognitive optical networks,” *Optik*, vol. 257, no. 2, pp. 1–16, 2022.

[27] S. Gurram, K. Geetha, S. P. Aditya Kumar, S. Hemalatha and K. Vinay, “Intelligent deep learning based ethnicity recognition and classification using facial images,” *Image and Vision Computing*, vol. 121, no. 1, pp. 1–13, 2022.

[28] G. Anitha and S. B. Priya, “Vision based real time monitoring system for elderly fall event detection using deep learning,” *Computer Systems Science & Engineering*, vol. 42, no. 1, pp. 7–103, 2021.

[29] N. Kanagavalli and B. P. Sankaralingam, “Social networks fake account and fake news identification with reliable deep learning,” *Intelligent Automation & Soft Computing*, vol. 33, no. 1, pp. 191–205, 2022.

[30] S. Bhargava, M. Kumar, N. Robert and S. Upadhye, “Optimal stacked sparse autoencoder based traffic flow prediction in intelligent transportation systems,” *Studies in Systems, Decision and Control*, vol. 412,.no. 1, pp. 111–127, 2022.

[31] B. Jaishankar, S. Vishwakarma, A. K. Singh Pundir, I. Patel and N. Arulkumar, “Blockchain for securing healthcare data using squirrel search optimization algorithm,” *Intelligent Automation & Soft Computing*, vol. 32, no. 3, pp. 1815–1829, 2022.

[32] T. Satish Kumar, S. Jothilakshmi, B. C. James and N. Arulkumar, “Hho-based vector quantization technique for biomedical image compression in cloud computing,” *International Journal of Image and Graphics*, vol. 26, no. 1, pp. 1–15, 2022.