National Conference on Engineering Innovations in Emerging Technologies
In association with International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET219430

A Deep Convolutional Neural Network Approach to Sign Alphabet Recognition

Uday Kumar Adusumilli[1], Sanjana M S[2], Teja S[2], Yashawanth K M[2], Raghavendra R[2], Dr. B. Udayabalan[3]

[1]Product Support Analyst, Associate, Infor, Bangalore, Karnataka, India

[2]Students, Department of Information Science and Engineering, East Point College of Engineering and Technology, Bangalore, Karnataka, India

[3]Professor, Department of Information Science and Engineering, East Point College of Engineering and Technology, Bangalore, Karnataka, India

ABSTRACT

In this paper, we present an application that has been developed to be used as a tool for the purposes of learning sign language for beginners that utilizes hand detection as part of the process. It uses a skin-color modelling technique, such as explicit thresholding in the skin-color space, which is based on modeling skin-color spaces. This predetermined range of skin-colors is used to determine how pixels (hand) will be extracted from non-pixels (background). To classify the images, convolutional neural networks (CNN) were fed the images for the creation of the classifier. The training of the images was done using Keras. A uniform background and proper lighting conditions enabled the system to achieve a test accuracy of 93.67%, of which 90.04% was attributed to ASL alphabet recognition, 93.44% for number recognition and 97.52% recognition of static words, surpassing other studies of the type. An approach which is based on this technique is used for fast computation as well as real-time processing. Deaf-dumb people face a number of social challenges as the communication barrier prevents them from accessing basic and essential services of the life that they are entitled to as members of the hearing community. In spite of the fact that a number of factors have been incorporated into the innovations in the automatic recognition of sign language, an adequate solution has yet to be reached because of a number of challenges. As far as I know, the vast majority of existing works focus on developing vision based recognizers by deriving complex feature descriptors from captured images of the gestures and applying a classical pattern analysis technique. Although utilizing these methods can be effective when dealing with small sign vocabulary captures with a complex and uncontrolled background, they are very limited when dealing with large sign vocabulary. This paper proposes a method for analyzing and representing hand gestures, which acts as the core component of the vocabulary for signing languages, using a deep convolutional neural networks (CNN) architecture. On two publicly accessible datasets (the NUS hand posture dataset and the American fingerspelling A dataset), the method was demonstrated to be more accurate in recognizing hand postures.

Keywords: ASL Alphabet Recognition, Sign Language, Recognition, Static Gesture, Deep Convolutional Neural Network (CNN).

I. INTRODUCTION

The use of sign language can be extremely useful for those who have speech and hearing problems (deaf and dumb). It is a common communication method for people to communicate with one another using hand gestures, facial expressions, and body movements. Since sign language isn't an international language, very few people are familiar with the gestures associated with it. Hearing people who do not know sign language interacting with people who are deaf and dumb can cause a communication breakdown that is of great concern in society. The spoken language is translated to sign language by interpreters who bridge the gap between the spoken language and the signed language. As this system is expensive, a deaf person may never be able to access it. Having automatic recognition of sign language gestures will alleviate the existing communication barrier for the deaf and dumb community. Sign language's vocabulary is primarily made up of hand gestures, while facial expressions and body movements emphasize the meaning of the gestures. There are two types of hand gestures: static and dynamic.

The static hand gesture, also called hand postures, involves a variety of hand positions and shapes without conveying any movement of the hands. Dynamic hand gestures are formed by combining varied hand postures and motion signals as a result of a variety of hand movements. There are many uses of fingerspelling, notably used for letter-by-letter names, place names, dates, ages, numbers, and words whose sign language vocabulary does not include predefined signs. The use of hand postures as a way of visual input in many fields (human computer interaction (HCI), human robot interaction (HRI), virtual reality systems, and medical procedures) has also become a popular choice as the hand positions reduce the necessity of direct physical contact with the conventional devices. In recent years, estimating hand postures automatically has been a hot research area based on both vision-based approaches, as well as signals-based approaches.

Taking into account the complexity of the data collection process, it appears that the vision-based approaches are more user friendly and convenient than that of other methods. Preprocessing and extraction of features are typically handled using classical methods of pattern analysis in most existing vision-based facial recognition work. This work presented a multilayer perceptron based upon features derived from discrete wavelet transforms (DTWs) for the recognition of letters and numbers in Persian Sign Language (PSL). The accuracy of their classification was 94.06 percent. Using discrete cosine transform (DCT) features and a hidden Markov model (HMM) Al-Rousan et al. implemented a user independent Arabic Sign Language Recognition system. It is claimed that 87% of the results are accurate. Based on the fusion of heterogeneous features, our researchers developed a method to recognize hand postures based on publicly available Triesh data that was 99.16% accurate. The researchers tested a support vector machine classifier that uses multiple kernels to recognize static hand gestures. They generated the feature vectors by combining shape context features with pyramidal HOG features as well as bags of features based on SIFT techniques.

With the aid of multiclass support vector machines which are constructed from scale invariant feature transforms (SIFT) descriptors, Nasser et al. propose a novel method for recognition of hand gestures. The percentage of correctly classified exams in their exams was 96.23%. A multiclass random forest classifier was proposed by Pugeault et al., using the features

extracted from Gabor filters in four levels, to recognize 24 static signs in the ASL (American Sign Language) alphabet. They estimate that 49% of their candidates will be accepted into their programs. A method has been proposed by Pramod et al to identify hand postures from complex backgrounds by using a human eye. As a result of combining the descriptors extracted from images that describe shape, color, texture, and density as well as the feature-based features, they were able to achieve an accuracy of 94.36%.

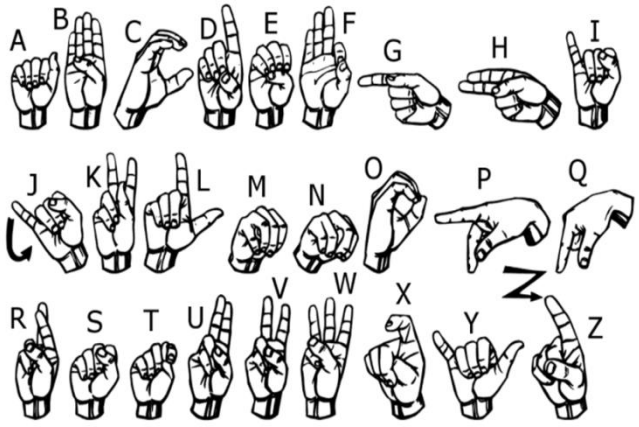

Fig. 1: ASL (American Sign Language).

It is still becoming increasingly important for gesture communication to include, along with hand gestures, facial expressions and body postures. Currently, there is also a research effort underway to develop a technique for recognizing all three different types of gestures using a gesture recognition system. Yang and Lee proposed that facial expressions and hand gestures can be used to automatically recognize British Sign Language (BSL) by means of automatic recognition. In spite of the fact that traditional methods have generated excellent results, they are unable to develop consistent feature descriptions for hand posture recognition in real-time scenarios because of a variety of challenging factors. A typical disadvantage of conventional machine learning techniques is their incapacity to accurately identify distinguishable patterns presented by natural raw data sets. A number of challenges stand in the way of detecting and segmenting hands in images with complex backgrounds when applying hand posture recognition approaches.

There is also a problem in analyzing the structural characteristics that account for geometrical variations in hand postures displayed by individuals displaying the same hand posture. The recognition of sign language in automatic systems is also difficult when there are many types of gestures, but their intraclass variations are relatively small. The detection and differentiation of the patterns in the images and videos is a complex process that involves computationally demanding steps and a great deal of domain knowledge. It has also been noted that public datasets with enough sample images do not exist which makes it difficult for researchers to study sign language recognition. Having difficulty communicating in sign language could be due to a variety of reasons, one of which is the difference in terminology between regions and countries. As a result, there are many different types of ASL, BSL, CSL (Chinese Sign Language), ISL (Indian Sign Language), and PSL.

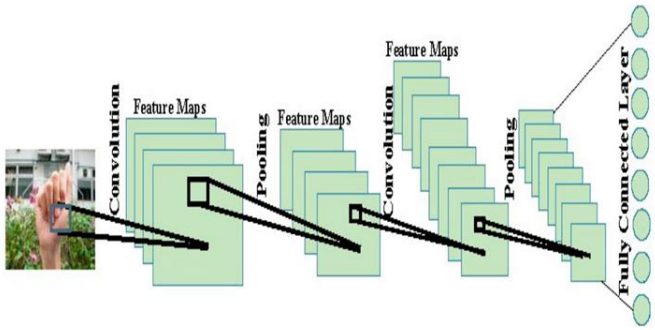

Fig. 2: Architecture of a typical CNN.

Recent advances in deep learning techniques and advances in convolutional neural networks (CNNs) have enabled advanced techniques to recognize hand gestures far more successfully than traditional systems, since they remove the need to develop complex feature descriptors from images after conventional pre-processing and segmentation steps. CNNs learn high level abstractions from images by using hierarchical architecture, resulting in good features being extracted from the images they process. As a result, a large number of gesture classes are no longer

generated with very slight variations within each one. This means that the problem of getting inconsistent feature descriptors can be avoided. A possible solution to identify ASL letters has been proposed by Ameen et al. by using CNN-based models. As part of their work, the researchers used two types of images of gestures in the ASL fingerspelling benchmark dataset and were able to achieve a prediction accuracy of 80.34 %. Using RGB and depth images Rastgoo et al. used a restricted Boltzmann machine (RBM) to recognize ASL's fingerspelling with RGB. This model uses the CNNs to detect hands, after which the images that have been detected are fed into a RBM to identify the signs.

Four public datasets were used to evaluate the model (Massey University Gesture Dataset, American Sign Language (ASL), Fingerspelling Dataset from the Center for Vision, Speech, and Signal Processing at University of Surrey, NYU, and ASL Fingerspelling A). The results of a recent study by Mohanty et al. shows using deep learning and CNN approach, and in the presence of a complex background and varying illumination conditions, it is possible to recognize static hand gestures. A good recognition result has been achieved with their proposed model using three publicly accessible benchmark datasets, mostly cluttered backgrounds, as well as the Triesh hand posture dataset with uniform dark background and the Marcel hand posture dataset with uniform dark background.

The aim of this paper is to propose an automated hand posture detection method that uses convolutional neural networks to compute deep parallel architectures. When the background is cluttered, it is difficult to segment by hand, and this model eliminates that process whenever the background is cluttered. It is possible to segment images using many parameters, such as skin color, hand shape, and many others, but none of them produce good results when applied to images with background colors. This will allow for the creation of more detailed feature descriptors that can be used as a basis for recognizing different gestures without the need to perform tedious calculations. It has been proven that the algorithm worked with datasets with both uniform and complex backgrounds, and that the results have been both promising and promising.

II. RELATED WORKS

It was only recently that various sensor types, particularly those relying on depth information, were developed, which led to all kinds of real-time applications, such as gesture recognition and recognition of sign language in real-time. As a result of their low cost, sensors such as Microsoft Kinect and leap motion controllers are widely used by researchers and institutions. The recognition of sentences as well as isolated words and letters can be separated into three subproblems within the context of sign language recognition. Here, the main focus of the paper is the American Sign Language (ASL) alphabets. For developing an alphabet recognition system for American Sign Language, there are three steps necessary: Hand segmentation, Feature extraction, and Classification. It is a well-known fact that there have been numerous publications in the field of fingerspelling in American Sign Language, but very few have delved into user-independent situations due to the large change of each signer's style. In March 2006, Pugueault and Bowden proposed using a Microsoft Kinect sensor to recognize hand gestures used in American Sign Language.

A total of 24 images of the English alphabet were submitted by five different signers for them to compile the RGB and depth values for. In order to predict the label of each fingerspelling letter, a Gabor filter is used as a texture feature extractor from a multiclass random forest classifier, which uses texture features. It was found that the Gabor filter is not capable of discriminating between different types of signals in laboratory experiments. In this study, this dataset was considered to be a benchmark dataset since it was used in a number of research papers.

Keskin et al. have recently used a random forest (RDF) method to estimate hand poses and classify hand shapes using the generative model. They assigned each depth pixel to an appropriate hand shape class by using a multilayer RDF. An RGB-D image of a hand gesture was used along with sparse auto-encoder (SAE) and principal component analysis by Li et al. The color and depth channels are learned respectively from two sparse auto-encoders using convolutional neural networks.

Several PCA layers were used to combine the features from both channels. The authors did not discuss the feasibility of their method for solving the signer-independent problem, only reporting experimental results for American sign language (ASL) datasets. Using depth contrast feature and per-pixel classification, Dong et al. segment the hand region into parts. The method used hierarchical mode-seeking to locate hand joint positions under kinematic constraints. Based on joint angles obtained, an ASL classifier was built using Random Forests (RF). In recognizing all static ASL alphabet signs under experimental tests, their method achieved above 70% and 90% accuracy using public datasets. According to Zhang and Tian, depth maps are encoded using H3DF (Histogram of 3D Facets). A 3D Facet associated with a 3D cloud point characterizes the 3D local support surface.

Signs are represented by their 3D shape, which is described by the H3DF descriptor. We achieved 73.3% recognition using SVM and 77.2% recognition using sparse representation (SR) classifiers for ASL alphabet. An earth mover distance metric using Kinect depth camera was proposed by Wang et al. to recognize gestures using hand gestures. The extracted depth, skeleton, and texture information is displayed as superpixels. In order to measure the dissimilarity between hand gestures, we applied the robust Superpixel Earth Mover's Distance metric (SP-EMD). They found 75.8% accuracy on the dataset they tested using their distance metric and features. An American sign language alphabet was represented using PCA with the Gabor filter and an orientation base hash code by Akhter and Arif-Ul-Islam. The classification was then performed using Artificial Neural Networks (ANN). In order to evaluate their method, they used a database with 576 ASL alphabet sign images. Compared to results using only RGB images alone, results using depth images worked much better in terms of timing and accuracy.

However, authors did not test their method against signer-independent scenarios, only using their own analyses, which were not publicly available. An image recognition system based on depth images was created by Kang et al. using convolutional neural networks (CNNs). The researchers trained more than 30 different alphabets and numbers into different CNNs using five different subjects. Based on the benchmark dataset and different learning hyperparameters used, they achieved an accuracy of 83.58% for leave-one-out testing. Using the CNN model, Ameen and Vadera used both color and depth images to recognize ASL fingerspelling. In order to extract features from each input, a CNN model is developed containing two convolutional layers. In order to classify, the features from both layers are concatenated into one layer. Based on the same benchmark dataset, accuracy was reported to be 83.34 percent. In their study, Tao et al. synthesize inferences from multiple views using CNN. In order to improve the performance of the CNN model, more perspective images are generated from the original depth image.

Based on the results of the different generated views, a final decision was made by combining their scores. While the researchers' method achieved state-of-the-art accuracy, they had to incur a high computational cost to ensure different viewpoints were possible. An alternative approach to American sign language recognition based on Recurrent Neural Networks (RNNs) and Leap Motion Controllers (LMCs) was proposed by Avola et al. An LMC device was used to track and detect the position and motion of the hand. In order to capture angles between bones of the fingers, LMC was used. Additionally to the

acquisition of data using LMC, the long term context of these dynamic gestures was modeled using RNN. A combination of static and dynamic gestures were used to evaluate their system. A majority of computer vision and gesture recognition problems are solved by deep learning algorithms. To solve hand gesture recognition, researchers have recently used different deep learning algorithms.

PCANet is one of the more powerful unsupervised deep learning algorithms that were successfully used to solve many complex object recognition problems among other deep learning algorithms. To learn features from American fingerspelling depth images, we utilize a PCANet model as opposed to the commonly used supervised CNN architecture. For the PCANet features that were extracted, a linear support vector machine classifier was used to classify them.

III. THE DATA SET

Data sets for American Sign Language (ASL) from MNIST were used for this project. The data sets are available at Kaggle. A total of 27455 training images and 7172 test images with a 28 x 28 pixel shape are contained in this dataset. As a group, these images can be classified into 25 classes of English alphabet (Z is not classified because of gestures). The dataset on Kaggle is available in CSV format (Comma Separated Values (CSV) is a delimited text file that uses a comma to separate values), with 27455 rows and (784*784) 785 columns. This dataset has one column for the class label and 784 columns for the pixels. Test data follows the same paradigm.

Fig. 3: Sample Training Data

IV. SYSTEM ARCHITECTURE AND MODEL DESIGN

Step-by-step instructions on how to create a neural network or deep learning model with Keras, utilizing the six key steps outlined below:

1. The data will be loaded.
2. Keras can be used to create neural networks.
3. The efficient numerical backend is used to compile Keras models.
4. Data training is the key to modeling success.
5. Analyze data to determine the effectiveness of a model.
6. Model-based predictions.

Models built in Keras are most easily constructed sequentially. Models can be built layer by layer with it. Our model is built using the 'add()' function. We have two Conv2D layers at the top. We use these convolution layers to deal with two-dimensional matrices representing our input images. The number of nodes in the first layer is 32 and in the second layer is 64 while in the third layer there are 128 and in the fourth layer there are 512. A larger or smaller dataset will affect this number. We'll use this for now.

Our filter matrix is termed the kernel size. If the kernel is three, then the filter matrix will be three by three.

An activation function is used to activate a layer. It is proposed that we use RECLAIM, which stands for Rectified Linear Activation, for the ReLU. Neuronal

networks have proven to work well with this activation function. As well as an input shape, we also have a first layer. The input image has a rectangular shape, 28,28, which is grayscale. Spatial 2D data can be pooled to the maximum.

This downsample takes the maximum value for each channel for each input window (of size defined by pool_size) over the input space (height and width). Steps are taken along each dimension to shift the window. During a training phase, dropouts are set randomly into effect to create hidden layers from hidden units (neurons).

A layer called 'Flatten' sits between the Conv2D and dense layers. Convolution and dense layers are connected by flattening. We will use the layer type 'Dense' for the output layer. In many neural networks, dense layers are used. In this case, softmax is activated. So that the output can be interpreted as a probability, Softmax makes the output sum to 1. Based on the option with the highest probability, the model will make its prediction. Check the summary of our model after we define it.

V. MODEL SUMMARY

You can see the output of the summary () function in the following diagram. Rows represent layers whose names are unique, allowing us to refer to them without any ambiguity. We added a layer to each model, and you will see the layers included in the diagram. Layers have outputs, whose shapes are listed in the "Output Shape" column. The output of each layer becomes the input for the next layer. For each layer, you can see the parameter number in the "Param #" column. There are a total of 12 parameters in the resulting output, which is equal to the number of trainable and non-trainable parameters. This model allows all layers to be trained.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_6 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 1, 1, 128)         0
_________________________________________________________________
dropout_7 (Dropout)          (None, 1, 1, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               66048
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 25)                12825
=================================================================
Total params: 171,545
Trainable params: 171,545
Non-trainable params: 0
```

Fig. 4: Model Summary

VI. CALCULATION OF PARAMETERS

A. MaxPooling2D Layers

There is no parameter for any MaxPooling2D layer. That's because all parameters are set to 0. There's no learning going on here since it's a static layer. By finding the maximum value for both 2 x 2 pools, it reduces the model complexity and extracts local features. This layer's output is used by the MaxPooling2D layer. Thus, max_pooling2d's input comes from the conv2d layer, which goes like this: (26, 26, 32). (26, 26) is the shape of each filter (n=32) in the max pooling. In the model, the pool size for max_pooling2d layer is 2 x 2, causing the shape of the data to become (13, 13), i.e. (26 / 2, 26 / 2).

Also, the input shape for MaxPooling2D layer 1 (MaxPooling2D_1) is (11, 11, 64). The resulting output shape in the "Output Shape" column is (5, 5, 64) following 2 x 2 pooling. Additionally, the third MaxPooling2D layer (max_pooling2d_1) has the input shape of (3, 3, 128). As can be seen in the "Output Shape" column, the resulting output shape is (1, 1, 128), when a 2 x 2 pooling algorithm is used. Additionally, the third MaxPooling2D layer (max_pooling2d_1) has the input shape of (3, 3, 128).

The result of applying a pooling scheme of 2 x 2 is (1, 1, 128).

B. Conv2D Layers

As noted in the formula below, our model has three Conv2D layers, and we calculate the parameters for each layer as described in the formula. A bias of 1 indicates that each of the filters we are learning has a bias. The model summary shows that 32 * (1 * 3 * 3 + 1) = 320 parameters are involved. Originally, it was 28 x 28 x 1, so 1 was the input channel number. This number corresponds to the number shown in the summary for the second Conv2D layer (conv2d_1), which was 64 * (32 * 3 * 3 + 1). As described in the model-building process, the output channel number is 64 and the input channel number from the previous MaxPooling2D layer (max_pooling2d) is 32. Calculating how many parameters the third Conv2D layer has is similar to calculating the second.

Fig. 5: Illustration of Processing Effects after MaxPooling and Conv2D Layers

C. Flatten Layer

Due to the lack of learning, there are no parameters for the Flattern layer. It would be interesting to learn how the output of these calculations is determined. In the case of the flatten layer, you can see that the input has the form (1, 1, 128). The flatten layer merely flattens the input data and outputs a shape that is simply a concatenation of all existing parameters using 1 * 1 * 128, which is not unlike the output of the flatten layer, which is 128.

D. Dense Layers

A Dense layer is present in our model. The parameter numbers are calculated using the following formula. Input_channel_number + (output_channel_number + 1) This formula can be used to calculate how many parameters are present in a Dense layer. 512 * (128 + 1) = 66048. This is the number of parameters for the first Dense layer (Dense): 128 input channels and 512 output channels. The input and output channels numbers of the second Dense layer (i.e., dense_1) are 512 and 25, respectively. 25*512+1=12825, so 25 times (512 + 1) is the number of parameters.

VII. TRAINING THE MODEL

The next step is to train the model. We will train our model using the 'fit()' function using the following parameters: training data (train_X), target data (train_Y), validation data, and the number of epochs. In order to validate our model, we will use our test set, which we have divided into X_test and y_test. Each epoch represents a cycle through the data by the model. Until a certain point, a model will improve as more epochs are run. When the model reaches that point, it will stop improving each time. We will set 50 epochs for our model.

Epoch is an arbitrary cutoff, used to separate training into distinct phases to facilitate logging and periodic evaluation. It can be expressed as "one pass over the entire dataset". To explain this further, an epoch represents the number of times you go through a training set.

As a result, batch_size signifies the size of the subset of data (such as 512) from which the network will be trained during its learning process. During each batch of training, the weight of the previous batch is updated, and the network is trained in a sequential manner. For tuning hyperparameters, models are fitted on these datasets to provide unbiased evaluations. This dataset is used for assessing the model fit on the training dataset in an unbiased

manner. On the training dataset and the validation dataset, both the loss and the accuracy are displayed on the verbose output on each interval.

VIII. COMPILING THE MODEL

After constructing our model, we need to compile it. To build the model, three parameters must be input: the optimizer, the loss, and the metrics. Controlling the learning rate is the function of the optimizer. As our optimizer, we will be using Adam. In general, Adam is a good optimizer for a wide range of situations. The learner rate is adjusted by the Adam optimizer during training. Calculation of the optimal weights for the models is based on the learning rate. A lower learning rate may result in more accurate weights (up to a point), but computing weights takes more time. The loss function for our data set will be 'categorical_crossentropy. Most classifications are done in this way. The model performs better when its score is lower. To make things even easier to understand when training the model, we will use the 'accuracy' metric.

The verbose setting allows you to specify how much detail you want to see for each epoch. The verbose=0 option shows nothing (silent). If verbose=2 is set, the epoch will be simply mentioned: Epoch 1/50.

After successful training, visualize the training performance of the CNN model.

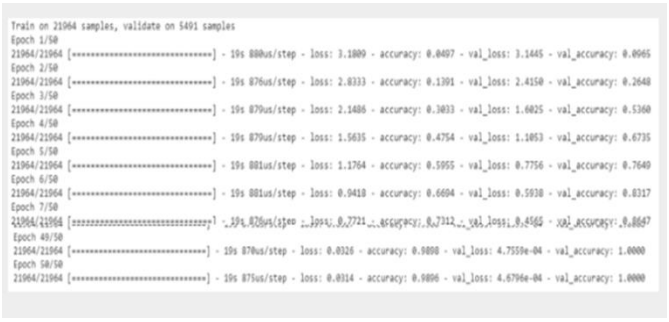

Fig. 6: Training Process

IX. EXPERIMENTAL RESULTS

Tests on publicly available hand posture datasets have been conducted using the proposed methodology. There are two datasets, the first of which is a hand posture dataset from the National University of Singapore (NUS) with cluttered backgrounds with 200 images in each class. 40 individuals from various ethnicities participate in the postures in complex natural settings. A three-layered convolutional operation is used in the training phase to extract the features. The filters used are 19x19, 17x17, and 15x15, respectively. Eight, sixteen, and 32 filters are applied in each layer and stride size is applied to maintain the input size equal to the output size in each layer. Using the max pooling layer with filter size (2,2) and stride size of three, each of the convolutional layers produces fewer dimensional feature maps. An optimization function for stochastic gradient descent with momentum (SGDM), having momentum rate of 0.90 and learning rate of 0.01 is used in this study to train the CNN.

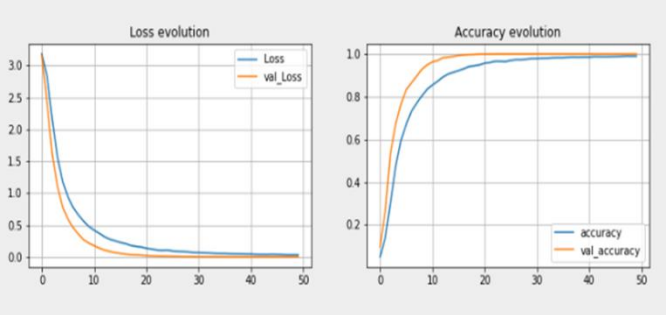

Fig. 7: Graph showing the accuracy of the proposed CNN model to recognize the hand postures.

In 20 epochs of training, the CNN model was optimally trained to recognize hand postures. Five fold cross validation is used to evaluate the performance of the classification. Datasets consisting of 40 sample images of each gesture class are divided into five subsets. Training is done on four subsets and testing is done on the remaining subset. We repeat the experiment five times in the same way until each of the subsets has been developed and tested. The performance metrics have been calculated using a macro-averaging approach.

X. CONCLUSION

This work uses deep learning to recognize gestures from raw images (RGB) using hand postures as inputs. The proposed CNN architecture eliminates the need to detect and segment hands from captured images, thereby reducing the computational burden faced by classical approaches when recognizing hand postures. Moreover, the model has the capability of automatically determining characteristics that distinguish hands based on very small intra class differences. Two publicly accessible datasets have been used to evaluate the performance of the proposed method. As measured by accuracy, precision, recall, and F1-score, the proposed CNN model is more capable of recognizing objects.

XI. REFERENCES

[1]. Rastgoo, R.; Kiani, K.; Escalera, S. Sign Language Recognition: A Deep Survey. Expert Syst. Appl. 2021, 164.

[2]. Zhao, T.; Liu, J.; Wang, Y.; Liu, H.; Chen, Y. Towards Low-Cost Sign Language Gesture Recognition Leveraging Wearables. IEEE Trans. Mob. Comput. 2021, 20, 1685–1701.

[3]. Sharma, S.; Kumar, K. ASL-3DCNN: American sign language recognition technique using 3-D convolutional neural networks. Multimed. Tools Appl. 2021.

[4]. Wadhawan, A.; Kumar, P. Sign Language Recognition Systems: A Decade Systematic Literature Review. Arch. Comput. Methods Eng. 2019, 28, 785–813.

[5]. Cooper, H.; Holt, B.; Bowden, R. Sign Language Recognition. In Visual Analysis of Humans; Moeslund, T., Hilton, A., Krüger, V., Sigal, L., Eds.; Springer: London, UK, 2011.

[6]. Dong, J.; Tang, Z.; Zhao, Q. Gesture recognition in augmented reality assisted assembly training. J. Phys. Conf. Ser. 2019, 1176, 032030.

[7]. Ascari Schultz, R.E.O.; Silva, L.; Pereira, R. Personalized interactive gesture recognition assistive technology. In Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems, Vitória, Brazil, 22–25 October 2019.

[8]. Kakkoth, S.S.; Gharge, S. Real Time Hand Gesture Recognition and its Applications in Assistive Technologies for Disabled. In Proceedings of the Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 16–18 August 2018.

[9]. Simão, M.A.; Gibaru, O.; Neto, P. Online Recognition of Incomplete Gesture Data to Interface Collaborative Robots. IEEE Trans. Ind. Electron. 2019, 66, 9372–9382.

[10]. Ding, I.; Chang, C.; He, C. A kinect-based gesture command control method for human action imitations of humanoid robots. In Proceedings of the 2014 International Conference on Fuzzy Theory and Its Applications (iFUZZY2014), Kaohsiung, Taiwan, 26–28 November 2014; pp. 208–211.

[11]. Barbhuiya, A.A.; Karsh, R.K.; Jain, R. CNN based feature extraction and classification for sign language. Multimed. Tools Appl. 2021, 80, 3051–3069.

[12]. Warchoł, D.; Kapu´sci ´nski, T.; Wysocki, M. Recognition of Fingerspelling Sequences in Polish Sign Language Using Point Clouds Obtained from Depth Images. Sensors 2019, 19, 1078. Lee, C.K.M.; Ng, K.K.H.; Chen, C.-H.; Lau, H.C.W.; Chung, S.Y.; Tsoi, T. American sign language recognition and training method with recurrent neural network. Expert Syst. Appl. 2021, 167, 114403. Appl. Sci. 2021, 11, 5594 20 of 20

[13]. Rastgoo, R.; Kiani, K.; Escalera, S. Multi-Modal Deep Hand Sign Language Recognition in Still Images Using Restricted Boltzmann Machine. Entropy 2018, 20, 809.

[14]. Yang, S.; Lee, S.; Byun, Y. Gesture Recognition for Home Automation Using Transfer Learning. In Proceedings of the 2018 International Conference on Intelligent Informatics and

Biomedical Sciences (ICIIBMS), Bangkok, Thailand, 21–24 October 2018; pp. 136–138.

[15]. Ye, Q.; Yang, L.; Xue, G. Hand-free Gesture Recognition for Vehicle Infotainment System Control. Proceedings of the 2018 IEEE Vehicular Networking Conference (VNC), Taipei, Taiwan, 5–7 December 2018; pp. 1–2.

[16]. Akhtar, Z.U.A.; Wang, H. WiFi-Based Gesture Recognition for Vehicular Infotainment System—An Integrated Approach. Appl. Sci. 2019, 9, 5268.

[17]. Cheok, M.J.; Omar, Z.; Jaward, M.H. A review of hand gesture and sign language recognition techniques. Int. J. Mach. Learn. Cyber. 2019, 10, 131–153.

[18]. Elakkiya, R. Machine learning based sign language recognition: A review and its research frontier. J. Ambient. Intell. Hum. Comput. 2020.

[19]. Luqman, H.; El-Alfy, E.S.; BinMakhashen, G.M. Joint space representation and recognition of sign language fingerspelling using Gabor filter and convolutional neural network. Multimed. Tools Appl. 2021, 80, 10213–10234.

[20]. Shi, B.; Del Rio, A.M.; Keane, J.; Michaux, J.; Brentari, D.; Shakhnarovich, G.; Livescu, K. American sign language fingerspelling recognition in the wild. In Proceedings of the 2018 IEEE Spoken Language Technology Workshop (SLT), Athens, Greece, 18–21 December 2018; pp. 145–152.

[21]. Jiang, X.; Zhang, Y.D. Chinese sign language fingerspelling via six-layer convolutional neural network with leaky rectified linear units for therapy and rehabilitation. J. Med. Imaging Health Inform. 2019, 9, 2031–2090.

[22]. Tao, W.; Leu, M.C.; Yin, Z. American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion. Eng. Appl. Artif. Intell. 2018, 76, 202–213.

[23]. Bird, J.J.; Ekárt, A.; Faria, D.R. British Sign Language Recognition via Late Fusion of Computer Vision and Leap Motion with Transfer Learning to American Sign Language. Sensors 2020, 20, 5151.

[24]. Chong, T.-W.; Lee, B.-G. American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach. Sensors 2018, 18, 3554.

[25]. Pezzuoli, F.; Corona, D.; Corradini, M.L.; Cristofaro, A. Development of a Wearable Device for Sign Language Translation. In Human Friendly Robotics; Ficuciello, F., Ruggiero, F., Finzi, A., Eds.; Springer: Cham, Switzerland, 2019.

[26]. Yuan, G.; Liu, X.; Yan, Q.; Qiao, S.; Wang, Z.; Yuan, L. Hand Gesture Recognition Using Deep Feature Fusion Network Based on Wearable Sensors. IEEE Sensors J. 2020, 21, 539–547.

[27]. Ahmed, M.A.; Zaidan, B.B.; Zaidan, A.A.; Salih, M.M.; Al-qaysi, Z.T.; Alamoodi, A.H. Based on a wearable sensory device in 3D-printed humanoid: A new real-time sign language recognition system. Measurement 2021, 108431.

[28]. Khomami, S.A.; Shamekhi, S. Persian sign language recognition using IMU and surface EMG sensors. Measurement 2021, 108471.

[29]. Siddiqui, N.; Chan, R.H.M. Hand Gesture Recognition Using Multiple Acoustic Measurements at Wrist. IEEE Trans. Hum. Mach. Syst. 2021, 51, 56–62.