Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images

Aditya Das[1], Shantanu Gawde[1], Khyati Suratwala[1] and Dr. Dhananjay Kalbande*
Department of Computer Engineering
Sardar Patel Institute of Technology
Mumbai, India
adityadas96@gmail.com, shantanu.gawde@gmail.com, khyati.s.96@gmail.com,
drkalbande@spit.ac.in

***Abstract*—Sign Language detection by technology is an overlooked concept despite there being a large social group which could benefit by it. There are not many technologies which help in connecting this social group to the rest of the world. Understanding sign language is one of the primary enablers in helping users of sign language communicate with the rest of the society. Image classification and machine learning can be used to help computers recognize sign language, which could then be interpreted by other people. Convolutional neural networks have been employed in this paper to recognize sign language gestures. The image dataset used consists of static sign language gestures captured on an RGB camera. Preprocessing was performed on the images, which then served as the cleaned input. The paper presents results obtained by retraining and testing this sign language gestures dataset on a convolutional neural network model using Inception v3. The model consists of multiple convolution filter inputs that are processed on the same input. The validation accuracy obtained was above 90% This paper also reviews the various attempts that have been made at sign language detection using machine learning and depth data of images. It takes stock of the various challenges posed in tackling such a problem, and outlines future scope as well.**

Keywords—Sign Language Recognition, Deep Learning, Inception V3, Image Processing.

I. INTRODUCTION

Understanding human motions can be posed as a pattern recognition problem. If a computer can detect and distinguish these human motion patterns, the desired message can be reconstructed. Static sign gestures used to signify alphabets and numbers have been detected successfully. However, this system can be extended for words and sentence recognition too. Here, we have used American Sign Language (ASL) as the sign language whose gestures we attempt to detect. Various other sign languages have originated from this as well.

Spoken language is the medium of communication between a majority of the population. Without spoken language, it would not be possible for a large proportion of the population to communicate. However, despite the existence of spoken language, a section of the population is not able to communicate with the majority of the population. Mute people are unable to communicate using spoken languages. Sign language comes to the aid of this section of the community. Using facial expressions, static hand symbols, and hand gestures, sign language provides tools to communicate just as spoken language does. Similar to spoken languages, there are various types of sign languages as well. These have originated like spoken language, in accordance with dialect and geography. Examples include Polish Sign Language, American Sign Language, Indian Sign Language, etc. Due to these differences, it does have its own drawbacks. Primarily, like all spoken languages, it can be used only amongst people who know the language. Users of sign language who are speech-impaired, are unable to interact with the rest of the population who is only well-versed in spoken language. Therefore, it is vital that the gap between the two is bridged to facilitate better and more efficient communication. Another hindrance this section of the population faces is in the field of human-computer interaction, especially when their only means of communication is through sign. The users of ASL have been put at 250,000 to 500,000. Compared to the size of the population, this is a small number[19]. Technology in the form of multiple software packages has been developed for the purpose of teaching and understanding sign language[20][21]. But the progress in using modern technology for sign language recognition has been good but limited[22]. There arises a need for a software capable of recognizing and interpreting sign language accurately. More importantly, it should act as a bridge between the users of sign language and those who do not have any immediate motivation to learn/understand the language. Our paper contributes towards this effort, by experimenting with one such methodology to verify its efficiency in recognizing sign language more efficiently. For this, we have taken into consideration ASL (American Sign Language) in our experimentation.

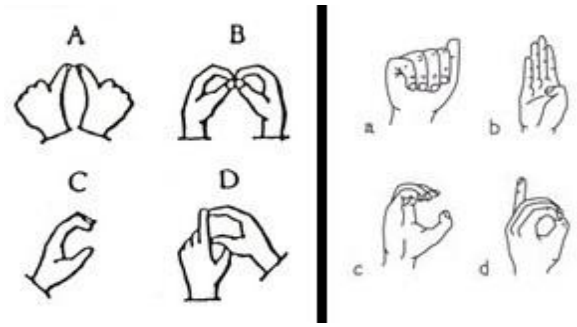

Fig. 1. Basic gestures in American Sign Language and Indian Sign Language.

II. CONVOLUTIONAL NEURAL NETWORKS

In a normal neural network, the neurons belonging to various layers will be individually connected to each other. If the input to this neural network is an image, each pixel in the image will be connected to a neuron in the input layer.

[1]All co-authors have equal contribution in the paper

*Dr. Dhananjay R. Kalbande works as the Head of Computer Department and is the Dean of Industry Relations at S. P. I. T.

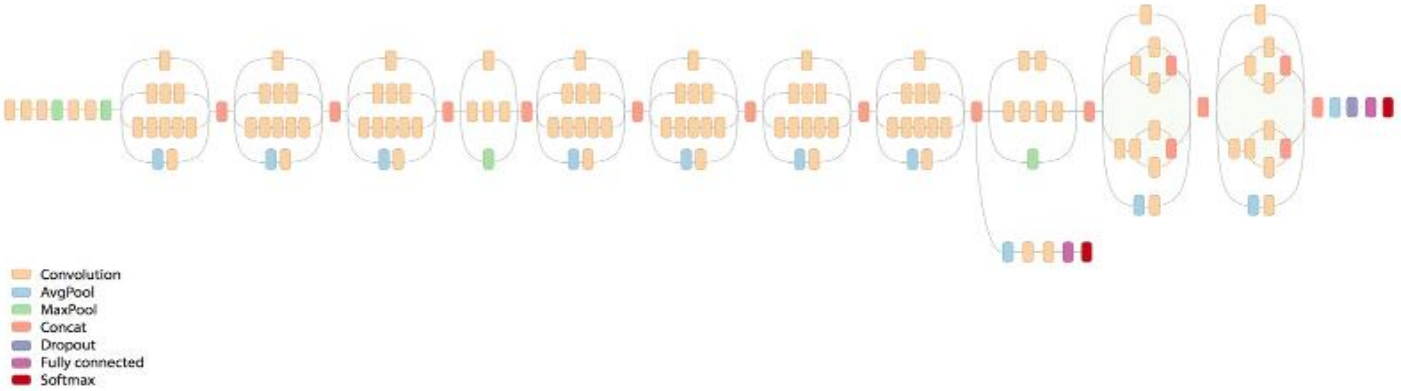

Fig. 2. Architecture of Inception v3

Considering a RGB image of 256 x 256 x 3 will result in 196608 values, each ranging in intensity from 0 to 255. As one could expect in such a scenario, this is an infeasible value with so many neurons and subsequently, so many weights to be updated in a normal neural network. Convolutional neural networks help in overcoming this drawback. In any image, not all pixels carry useful information[9]. Pixels that carry unrequired information about the background are numerous, and do not give the computer useful data about the object. Convolutional neural networks instead convolve around the input pixels i.e. the image, to reduce its dimensions when it is input to the next layer. This is done by a series of operations such as convolving, which is repeatedly reducing the dimensions of a patch of image by using filters[17]. Subsequent operations include pooling, applying final activation map to a fully connected layer, and then classifying using a softmax layer[7].

III. Inception Model

Inception also makes use of convolutional neural networks. However, instead of stacking convoluting, pooling, softmax, and related layers on top of each other, these layers are instead stacked parallel to each other. Post this, outputs are concatenated, which are fed as inputs to the next block[6]. Traditional convolutional neural networks which do not have this kind of parallelism, usually require a larger amount of processing and computing power. Inception blocks are the repeated loops of operations which are depicted in the figure. Inception type architecture enables processing of large amounts of data with the use of limited computer resources. Therefore, this also makes it a better fit for the problem that we are attempting to solve as deploying a system or a camera for capturing of sign language images for training will likely be done on a device with limited computing capabilities.[6][8][11]

IV. Literature Review

Zafar Ahmed Ansari and Gaurav Harit [2] have carried out extensive research in the field of correctly categorising Indian Sign Language gestures. They have classified 140 classes which include finger-spelling numbers, alphabets, and other common phrases as well. For the dataset, a Kinect sensor is used. RGB images of resolution 640 x 480 are captured along with their depth data. Depth values of each pixel are captured in the depth data of the image. Each pixel of the image corresponds to an value in the depth data file. As the subjects in the dataset were all standing with their hands outstretched, the hand in the image will have the least depth. We were able to confirm this by masking pixels whose corresponding depth values were more than a certain threshold value. The part of the image which was left after doing this process had the palm with the gesture. However, as this had certain inconsistencies, we were unable to use this dataset for training with convolutional neural networks. Unsupervised learning was carried out using the traditional K-means algorithm. Points of local maxima were used as K clustering centers to initialize K seeds in order to correctly detect various parts of the body such as torso, hand, etc. To extract features and train the dataset, SIFT feature mapping, and Gaussian masks were used. Resultant accuracy was over 90%.

The recognition of sign symbols by Divya Deora and Nikesh Bajaj[2] is done with PCA (Principal Component analysis). The paper also proposes recognition with neural networks. The data they acquired was using a 3 mega pixels camera and hence the quality was poor. For each sign they took 15 images and stored it in their database. The small dataset was one of the reasons why their results were not satisfactory. They performed segmentation and separated the RGB into components and performed a simple boundary pixel analysis. They have mentioned that even though combining finger tip algorithm with PCA provided a good result, a better output can be obtained using neural networks.

Sign language recognition carried out by Lionel et al[3] uses Convolutional Neural Network (CNN). To automate the process of sign language recognition, two steps were carried out - feature extraction and classifications of the actions. The first step is performed using CNN and the next step using an Artificial Neural Network. The dataset includes a total of 20 distinct Italian gestures that have been signed by 27 people in different surroundings. Videos are recorded using Microsoft Kinect. A total of 6600 images were used for development purposes out of which 4600 were used for training and the rest for validation. The images were preprocessed to crop the higher part of the hand and torso. For all gestures, only one side is required to be learnt by the model. Max pooling is used for feature extraction and classification. It consists of 2 CNNs, the output of which enters an ANN. The ANN combines the output of the two CNNs. A CPU was used for augmentation of the data and a GPU for training the model. Zooming rotation and transformation was done as a part of data distortion.

Training using this model resulted in an accuracy of 91.7%, and a mean Jaccard index of 0.789 was obtained using the model in ChaLearn 2014.

V. Dataset Used

- The dataset we have used[5] has images of static sign language gestures in American Sign Language.
- There are 24 labels of static gestures from letters A to Y, excluding J.
- J is not included as it is a dynamic gesture.
- There are on an average 100 images per class.
- From each of the classes, 20% of the images were used specifically for testing.

VI. Implementation

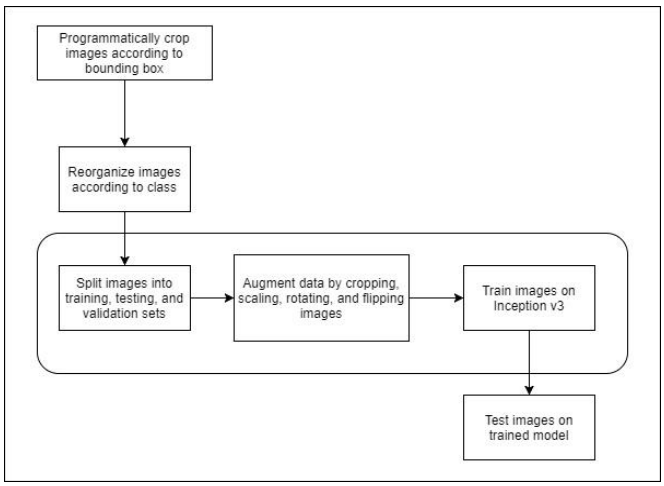

Fig. 3. Flow diagram of implementation

A. Image Cropping

In the given dataset, images show static hand gestures. However, along with the hand gesture which is present in the image, there are also portions of the background and the face in the image. The dataset also provides bounding box values for the static hand gesture in the image. While it already seems intuitive to crop images to only have the area which is to be recognized, it is also essential to understand the logic behind it. The different layers in the convolutional neural networks are used to detect different features in the image; these go from basic to more complete features. The initial layers detect curves, colours, and other basic image features. In subsequent layers, the shape of the object is detected, and then the classification is done by the softmax layer. Therefore, in order to prevent seemingly unnecessary shapes from being detected, these images are cropped using a Python script. The resultant images are on an average 160 x 160 in size. These have then been classified into their appropriate classes. 20% of the images of each class have been separated out for testing.

B. Data Augmentation

This is performed during training in order to run the images through simple distortions. This includes operations such as cropping, scaling, and flipping. By doing this, it is ensured that the neural network is not overfitted to a particular type of images. This also reflects the variations present in real-world data.

- Cropping: In the full image, a bounding box is placed randomly at random positions. This does not include the cropping preprocessing that has been already done. In a real life scenario, it is possible that while cropping, the entire gesture is not captured correctly. Cropping parameter is used to determine amount of cropping. In order to get a section of the image that is half that of the image, this parameter is initialized to 50%. In order to get the same image, the parameter is set to zero.
- Scaling: It is similar to cropping, except that the box used for bounding is always centered and within the given range the size is varied randomly. However, unlike cropping, the bounding box does not contain only the section of the image contained in it. Instead, the image is resized to fit the bounding box.
- Rotation: The images were rotated by 5 degrees and 10 degrees in both clockwise and anti-clockwise direction. A greater amount of rotation would've changed the meaning of the symbol.
- We could not flip the images to increase the dataset because that would result in the symbols changing their meaning.

C. Splitting into Sets

Once the images are recognised, the images are required to be split into training, testing and validation sets. The validation and testing percentage are specified as arguments. This is done in order to have control over how many images should be trained on the model. In order to have a consistent method of splitting images, a custom algorithm is used. A hash is created from the file name using SHA-1. This is then converted into percentage. If this percentage is less than the given validation percentage, it is added to the validation set. If it is less than the sum of the validation and testing percentages, it is added to the testing set. In all other cases, it is added to the training set. This algorithm has been chosen only to provide a consistent method of splitting, and any other method of splitting can also be used.

D. Training

The top-layer was retrained so that the new classes are identified. We were able to obtain results of significant accuracy by setting the learning rate at 0.05. On further increase or decrease of learning rate however, the accuracy reduced drastically.

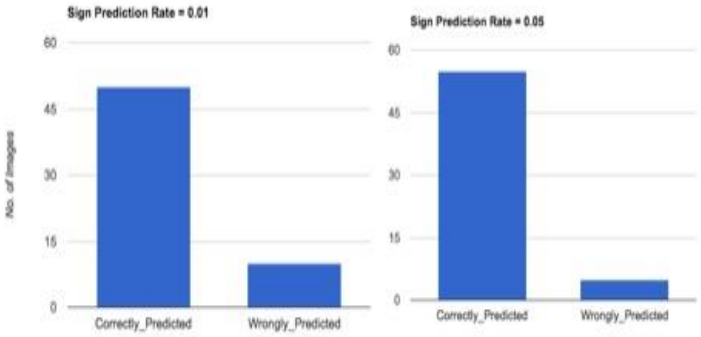

Fig. 4. Sign prediction rates

As seen above when the learning rate was 0.01, out of 60 images tested 48 were predicted correctly and 12 erroneously. However, when we tweaked the learning rate to 0.05 and trained the model again, on testing, only 4 images were predicted erroneously whereas 56 were predicted correctly. This is because when the learning rate is changed, the weights get changed, and therefore the error rate varies.[16]

E. Testing

As seen in the flow diagram described above, the last stage of implementation is the testing of images on the trained model to determine the level of accuracy of the final model. This has been implemented using the TensorFlow python library for deep learning. For testing, the images in the testing partition created as mentioned above, are accessed.

Fig. 5. Symbol of letter 'R' used for testing

First, the image data is read sequentially. Along with this, the label files are loaded and the carriage return is stripped off. Next, the file representing the new trained model is loaded as a TensorFlow graph. Once the model is loaded, the image data is loaded as input to the graph, for a preliminary prediction. As a result of this preliminary run, each image is assigned a percentage confidence. The confidence percent depicts the level up to which the model was able to predict the result. After the first prediction run, the labels are shown, sorted in order of confidence.

VII. Results

A. Prediction

We designed our system to predict the percentage probability of the input symbol. A sample is shown below in which we passed an image of the sign symbol of 'R'.

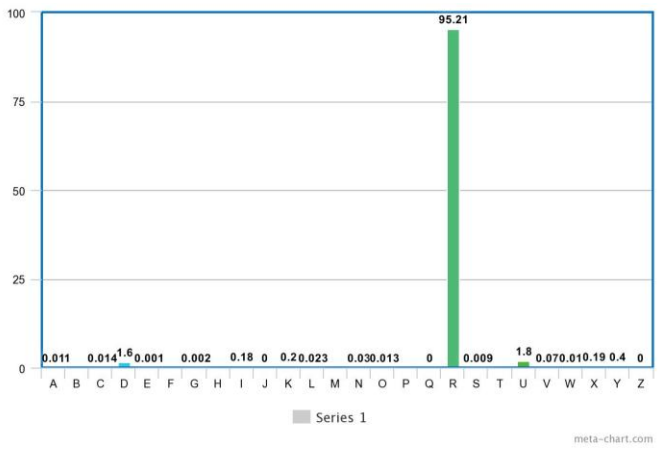

Fig. 6. Probability graph obtained

Our model correctly predicted it as being 'R' with a probability of over 95%.

B. Cross Entropy

The usual loss function for neural networks cannot be applied to neural networks with the softmax layer, which is used for classification. Usually, the error function used puts emphasis on the incorrect outputs as well. This is the mean-square method for calculating error function. When considering neural networks with a softmax layer, an alternate error function is used. This error function is called cross entropy. Following formula is used for calculation of cross entropy.

$$-\sum_{i=0}^{n} \ln(o_i) * t_i \qquad (1)$$

In our case as we increased the epochs the cross entropy, an error function reduced drastically. During the first epoch we got a cross entropy over 3.0 but soon after 250 epochs the cross entropy dropped below 0.1. Also, as the cross entropy decreased the validation accuracy increased from 9% at the first epoch to 97% at 350th epoch.

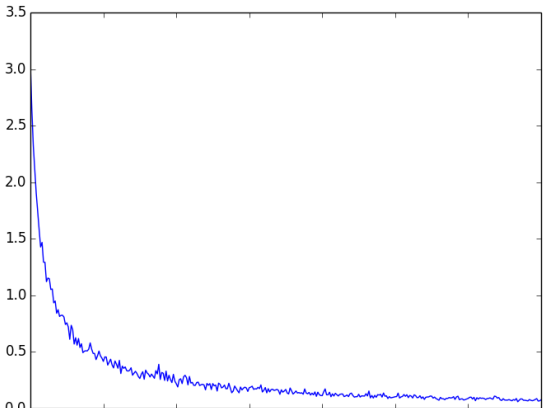

Fig. 7. Cross entropy obtained

C. *Validation Accuracy*

The below graph shows how the validation accuracy varies as the number of epochs increases. Total epochs plotted in the graph is 350.

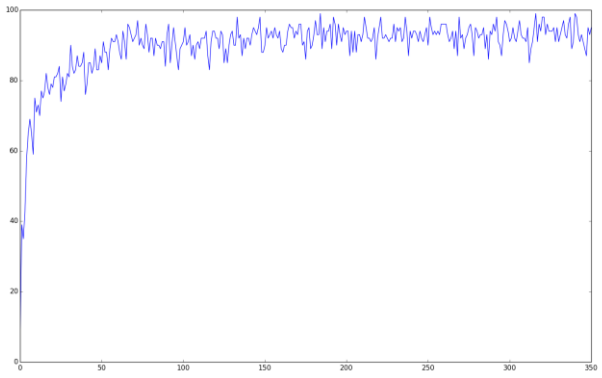

Fig. 8. Validation accuracy obtained

As the epochs increase the top layer of the pretrained model is trained more for our dataset. Thus, we see an increase in validation accuracy and a decrease in cross entropy. We were able to obtain an average validation accuracy of 90% with the greatest validation accuracy being 98%.

VIII. Future Scope

Future scope includes but is not limited to:

- The implementation of our model for other sign languages such as Indian Sign Language, as it is currently limited to American Sign Language
- Enhancement of the model to recognize common words and expressions
- Further training the neural network to efficiently recognize symbols involving two hands
- Improving the results by using linear classifiers
- Including dynamic hand gestures in addition to the current static finger spelling
- Using convolutional neural networks to also account for depth data in order to detect gestures captured by devices such as Kinect

IX. Challenges

- The biggest challenge for us was to find the right dataset. Some datasets had very few images while others had only a few symbols.
- Choosing the correct dataset-model combination was tricky as there were models which tended to overfit to a particular dataset.
- The next challenge faced was modifying the Inception v3 model for our use, since it is relatively new, not many references was available online.
- Also we first started with the Indian Sign Language, but due to the lack of dataset we had to switch to American Sign Language.

X. Conclusion

Using Inception v3, we were successfully able to use convolutional neural networks for correctly recognizing images of static sign language gestures. The results obtained showed a consistently high accuracy rate of over 90%. This goes on to show that given a proper dataset, and correctly cropped images, Inception v3 is an apt model for static sign language gesture recognition.

Acknowledgment

We would like to thank the open-source community on the Internet, in particular the tutors at TensorFlow, whose help was indispensable to our learning process.

References

[1] Divya Deora and Nikesh Bajaj, "Indian Sign Language Recognition", in Emerging Technology Trends in Electronics, Communication and Networking (ET2ECN), 2012 1st International Conf. , 2012. doi: 10.1109/ET2ECN.2012.6470093

[2] Zafar Ahmed Ansari and Gaurav Harit, “Nearest Neighbour Classification of Indian Sign Language Gestures using Kinect Camera”, in Sadhana, Vol. 41, No. 2, February 2016, pp. 161-182

[3] Lionel Pigou et al, "Sign Language Recognition using Convolutional Neural Networks", presented at the Ghent University, ELIS, Belgium

[4] Chenyang Zhang et al, "Multi-modality American Sign Language Recognition", in Image Processing (ICIP), 2016 IEEE International Conf. ,2016. doi: 10.1109/ICIP.2016.7532886

[5] Sreehari (2016, Dec 26). Weekend project: sign language and static-gesture recognition using scikit-learn [Online]. Available: https://medium.freecodecamp.org/weekend-projects-sign-language-and-static-gesture-recognition-using-scikit-learn-60813d600e79

[6] Christian Szegedy et al (2015, Dec 2). Rethinking the Inception Architecture for Computer Vision(2nd ed.) [Online]. Available: https://arxiv.org/abs/1512.00567

[7] Evgeny A. Smirnov et al (2014, Dec). "Comparison of Regularization Methods for ImageNet Classification with Deep Convolutional Neural Networks"

[8] Alex Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", in NIPS, 2012, pp. 1106â€“1114

[9] Michael Nielsen (2017, Aug). Using neural nets to recognize handwritten digits [Online]. Available: http://neuralnetworksanddeeplearning.com/chap1.html

[10] Sanil Jain and Kadi Vinay Sameer Raja, Indian Sign Language Gesture Recognition [Online]. Available: https://www.cse.iitk.ac.in/users/cs365/2015/_submissions/vinsam/slides.pdf

[11] Adrian Rosebrock (2017, March 20). ImageNet: VGGNet, ResNet, Inception, and Xception with Keras [Online] Available: https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

[12] Anup Kumar et al, "Sign Language Recognition", in Recent Advances in Information Technology (RAIT), 2016 3rd International Conference, 2016. doi: 10.1109/RAIT.2016.7507939

[13] N. Tanibata et al, "Extraction of Hand Features for Recognition of Sign Language Words", In Proc. Intl Conf. Vision Interface, pages 391-398, 2002.

[14] S. Tamura and S. Kawasaki, "Recognition of Sign Language Motion Images", In Pattern Recognition, volume 21, pages 343-353, 1988.

[15] Brandon Garcia and Sigberto Viesca, "Real-time American Sign Language Recognition with Convolutional Neural Networks", in Convolutional Neural Networks for Visual Recognition at Stanford University, 2016

[16] How to Retrain Inception's Final Layer for New Categories [Online]. Available: https://www.tensorflow.org/tutorials/image_retraining

[17] The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3) [Online]. Available: https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

[18] TensorFlow [Online]. Available: https://github.com/tensorflow/tensorflow

[19] American Sign Language [Online].

[20] Start ASL - The fun way to learn sign language for free [Online]. Article: https://www.startasl.com/