Sign Language Recognition using Deep Neural Networks

Joann Rachel Jacob
Northeastern University
Boston, MA
jacob.joan@northeastern.edu

Manasa Krishnan
Northeastern University
Boston, MA
krishnan.man@northeastern.edu

1 Introduction

Communication is a very important part of our lives. Though most of our communications are through speech, some people are unable to do so due to not being able to speak or hear. Deafness may be caused by genetics, birth defects, ear infections, drug use, excessive noise exposure and aging. The deaf and mute community benefit greatly from the use of sign language. While sign language is vital to their ability to communicate with others and with themselves, it receives little attention from the general public and ordinary people tend to overlook the relevance of sign language unless they are our close ones. Using the services of a sign language interpreter is one way but it can be quite expensive and thus requires an efficient and low cost solution.

There has been a lot of research to help this community in terms of translating the signs in the most efficient way through image recognition. The traditional pipeline of sign language recognition includes collection of image data or creation of the dataset, applying pre-processing techniques to images, feature extraction, training of the chosen model and classification of images by using Convolutional Neural Networks(CNNs).

Image and video datasets often have problems such as occlusion, improper background detection, imbalanced intensity of light etc. However, CNNs prove to be effective in creating a robust model with a high level of resistance to noise. The image recognition method is used to extract features from input images that are invariant to scale, translation, rotation, and movements among other factors. CNNs can also identify a hand gesture in a photograph. Deep Learning is a relatively new approach to machine learning that uses multiple hidden layers in contrast to traditional feed-forward networks and have more biologically inspired architecture and learning algorithms.

The aim of this project is to device a sign language recognition system made possible using deep learning techniques and to find the best method amongst them. Our proposed system for sign language recognition translates alphabets, digits and simple words from a custom dataset of hand gestures created utilizing OpenCV VideoCapture. Our system makes use of VGG16 and ResNet50 architectures which have been proved to be effective for image related tasks. VGG16 and ResNet50 models are two of the best vision model architectures till date and works best with image classification and object detection.

We initially trained our model using two American Sign Language (ASL) datasets that were already available, inorder to understand the process and create baseline models. Utilizing one of these datasets, we experimented with multiple models namely, VGG16 and Resnet50 after which we created our own custom dataset including more classes and chose ResNet50 for our final model based on the experiments we conducted.

In our project, we not only use CNNs but also augment the dataset to include more images and create a dataset which can be expanded and modified as required. Our pipeline consists of dataset generation, pre-processing of images, modeling and prediction of signs provided. It is to achieve a sign language recognition system of alphabets, digits and simple words.

2 Related Work

There are different methodologies available for gesture recognition right now. The primary aim of these methods is to ease the communication by deducing the correct meaning of gestures and signs shown by the user. The process includes data acquisition and pre-processing, feature extraction and language recognition.

Abiyev, Rahib H[1] used ASL Alphabet dataset based on which they modeled using SSD, Inception V3 and SVM(hybrid model). The Inception model represented a field of 299 x 299 x 3 inputs (299 pixels and 3 channels representing the RGB channel). The modules of the hybrid model were defined sequentially. The system used squared_hinge loss and l2 penalty and made use of about 10 layers in their CNN architecture. Using this model, they were able to achieve an accuracy of about 97 per cent and a loss value of 0.0054. They also performed simulation with HOG plus neural networks where HOG was used for data pre-processing and feature extraction. The average value of the accuracy rate obtained was 99 per cent and the RMSE was 0.0126.

Sakshi Sharma and Sukhwinder Singh[2] in vision-based hand gesture recognition[2] have used ISL dataset to create the classification model. They used a new type of CNN called G-CNN(Gesture CNN) which consisted of 12 layers. The main advantage of using G-CNN was its compact representation which produced greater computational efficiency. They also examined the performance of VGG-11 and VGG-16 models. By using these models, they were able to produce a very high accuracy of over 94 per cent using G-CNN. This performance has been implemented on an augmented dataset which was created by generating new samples by rotation and scaling operations. K-fold cross validation is used to improve the performance of the model. 10-fold cross-validation has been used for G-CNN and it surpassed the accuracy of all the other models.

Garcia and Viesca[3] proposed Transfer Learning on the ASL dataset. They created a web application which makes use of cache of classified images. The system chose the top-5 most likely letters based on cache. Data augmentation was utilized here through which the images were padded with black pixels such that the aspect ratio was preserved upon resizing. One unique metric used in this paper was the top-5 accuracy which provided us with the percentage of classification in which the correct label got displayed in the said 5 classes with the highest scores. They also performed real-time classification and the model correctly classified in more than 70 per cent of the cases.

Chandhini Grandhi, Sean Liu and Divyank Rahoria[4] performed hand gesture recognition using ASL dataset with 29 classes. They trained a model on a VGG16 network from scratch using pre-trained weights. They also trained using InceptionNet and ResNet50. They preprocessed the data by resizing to 200 x 200 pixels, using random shuffle and normalization of the data. For InceptionNet model, they scaled down the resolution to 50 x 50 for faster computation and lower memory consumption. They used categorical cross-entropy loss for VGG16 and InceptionNet, CNN and ResNet50. They used Adam optimizer in baseline for InceptionNet and ResNet50 model and SGD for both the VGG16 and VGG16-pretrained models because of a study that found that SGD worked much better in reducing training and testing error, and Adam actually performed worse than any optimizer tried specifically for VGG16. All models were built using ReLU activation functions and a softmax layer at the output to predict the classes. ResNet50 gave the best results while using pretrained weights for VGG16 proved the effectiveness of transfer learning.

S. Dinesh, S. Sivaprakash, M. Keshav and K. Ramya [5] did hand gesture recognition from live video and utilized a pre-trained GoogLeNet architecture trained on the ILSVRC2012 dataset, as well as the Surrey University and Massey University ASL datasets in order to apply Transfer-Learning to this task. This paper proposes a model that consistently classifies letters a-e correctly with first-time users and another that correctly classifies letters a-k in a majority of cases. They have also made use of Xavier initialization for GoogLeNet. They have extracted spatial features for individual frames using CNN and temporal features using Faster RNN. They received an accuracy of 93.33% using this approach.

Sarfaraz Masood1, Harish Chandra Thuwal and Adhyan Srivastava [6] performed ASL hand gesture recognition for 36 classes including alphabets and digits using VGG16 obtaining an average accuracy of 95.54 percent. For pre-processing, the mean value of RGB over all pixels was subtracted from each pixel value thus centering the data. Each feature is made to have a similar range, in order to prevent the gradients from getting out of control. The training of the model was done using stochastic gradient descent and a batch size of 128.

3 Method/Study

3.1 Overview

We use deep convolutional neural networks for our model in this project. The tasks involved in our proposed system is shown in Figure 3.1.1.

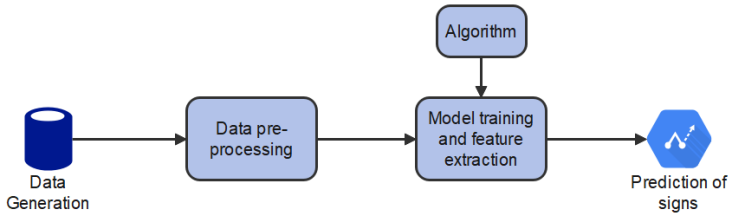

Figure 3.1.1 Structure of the proposed model

Our final system makes use of the state-of-the-art model ResNet50 to predict 38 different hand gestures including 26 alphabets, 9 digits and 3 symbols utilizing our custom generated dataset and gave good prediction results in comparison to multiple models implemented for this problem domain.

3.2 Dataset Creation

Apart from ASL alphabet dataset, we created our own dataset using OpenCV VideoCapture feature. This helps create a video capture object which is useful in capturing videos through webcam. We captured 1200 images per class where 1000 is for train and 200 is for test. Hence, the script will run until 1200 images are taken and makes use of the read() method to read the frames using the above created object. cv2.imshow() method is used to show the frames in the video. We have added a bounding box in the frame shown and our hand is placed in this particular box so that only the portion within the bounding box is cropped and saved as the image. The folder structure is in such a manner that there is a main folder dataset having two folders train and test in it. Train includes 1000 images for each class and test includes 200 images for each class where each image is of size 422 x 422. Images in the custom dataset are shown in Figure 3.2.1.

Figure 3.2.1 Images created using OpenCV Video Capture

3.3 Data Processing

Pre-processing plays an important role in this problem as the image we generate using our webcam must be properly detected given occlusion and background problems. Some of the robust techniques of image pre-processing include applying gaussian filter, image cropping, bootstrapping and image segmentation techniques include gray scaling, canny edge detection, skin segmentation etc. As part of image acquisition, we tried creating images for few alphabets and tried converting the RGB image obtained into grayscale. We tried understanding image segmentation to partition the image into multiple segments. This was done in order to better analyse the image. The hand image captured through the web camera will have a background which can be made uniform to white or black. Image segmentation can be done to extract the hand alone and separate out the background.

Since all the images present in the ASL alphabet dataset had a fairly white background and not much of a variance, we decided to progress along with just resizing the image size to 128 x 128. Using Dataloader of FastAI, we divided the datasets into batches according to their labels. This process was carried out for both VGG16 and ResNet50 models.

For the final custom dataset we created, all images were on a white background and performed well without much pre-processing. We tried augmentation techniques as part of Keras ImageDataGenerator including horizontal flip, random rotation range of 90 degrees, zoom range of [0.5,1.0], brightness range of [0.2, 1.0] and height shift range of 0.5.

3.4 Model

Our model makes use of Keras pretrained ResNet50 model which is a 50-layer deep neural network as shown in Figure. 3.4.1 and can be utilized as a state-of-the-art image classification model. This model that has been pre-trained on the ImageNet dataset - a dataset that has 100,000+ images across 200 different classes. Unlike other models, ResNet50 focuses more on having a large number of convolution layers than hyper-parameters. In ResNet architecture, a skip connection allows the gradient to be directly backpropagated to earlier layers. The model consists of 5 stages each with a convolution and identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers.

We used input image size of 128 x 128 for our final model, normalized our data and utilized imagenet weights defined by Keras pretrained models particular for ResNet50. Our resnet50 model is added with a dropout of 0.2 followed by a final dense layer for 38 classes. Adam optimizer and categorical cross entropy loss are utilized for training with a batch size of 32.

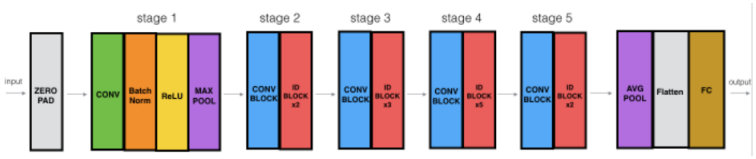

Figure 3.4.1 ResNet50 Model Architecture[10]

The total number of parameters utilized for this configuration is $24,832,934$ out of which $1,245,222$ are trainable and $23,587,712$ are non-trainable. Our model was trained using $30,400$ images and validated using 7600 images belonging to 38 classes and had 7600 images in testing. For the final model, we did not perform any data augmentation on the dataset and still were able to achieve a considerable accuracy. We anticipate to achieve a higher accuracy by fine-tuning the same model with combinations of image pre-processing techniques and making use of freezing or unfreezing of layers.

4 Experiments

4.1 Benchmark Datasets and Evaluation Metrics

We used two open-source datasets for our baseline model. The datasets are:

1. American Sign Language dataset [7]

2. Sign Language MNIST dataset [8]
3. Our custom American Sign Language dataset [9]

ASL Alphabet - The data set is made up of images of alphabets from American Sign Language, organized into 29 folders that reflect different classes. There are 87,000 $200 * 200$ pixel images in the training data set. There are 29 classes, including 26 for letters A through Z and three for SPACE, DELETE, and NOTHING. These three classes are extremely useful in real-time and classification applications. There are only 29 photos in the test data set.

Sign Language MNIST - Each training and test case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z. Due to gesture motions, there are no cases for 9=J or 25=Z. There are 27,455 examples in the training data and 7172 cases in the test data.

American Sign Language custom dataset - It consists of 38 classes including 26 alphabets, 9 digits and 3 words, each having 1000 images in train and 200 images in test folders with image size 422 x 422.

The evaluation metrics used for our baseline models are accuracy score in order to obtain the validation and test accuracy. Confusion matrix is used for visualising the comparison of different scores such as Precision, Recall, F-1 Score, Prevalence and Accuracy. Confusion matrix is a matrix which describes the complete performance of the model used to interpret the type of errors made by the model based on which corrections are to be made. F1 Score helps to measure a test's accuracy. Precision is the number of correct positive results divided by the number of positive results predicted by the classifier. Recall is the number of correct positive results divided by the number of all relevant samples. The values can be obtained by the following equations:

$$Accuracy = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions}$$

$$TruePositiveRate = \frac{TruePositive}{TotalPositive}$$

$$TrueNegativeRate = \frac{TrueNegative}{TotalNegative}$$

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

We also measured the difference of predictions using Categorical Cross-entropy which is best for probabilistic interpretation. In multi-class classification tasks, categorical cross entropy is a loss function. It is given to be:

$$CrossEntropyLoss = -\sum_{i \forall classes} p_i log(s_i)$$

where p_i is the true value of the observation i, s_i is the score obtained after training for each class in the set of classes.
These are problems in which an example can only fit into one of many possible categories, and the model must choose one. Its formal purpose is to calculate the difference between two probability distributions. We might have imbalanced datasets for our domain of problem in the future. In order to handle that, we use the aforementioned metrics because they are robust to such problems. Adam Optimizer achieves a greater efficiency and faster convergence and used especially in deep conventional neural networks. As our datasets are very large, using Adam would be a good metric for faster convergence. These are the benchmark evaluation metrics for our proposal.

4.2 Baseline Results

For our baseline model, we considered two datasets for better interpretation of domain knowledge using CNN. CNNs take an input image and captures the spatial and temporal dependencies of the image. Each CNN layer learns filters of increasing complexity.

Using ASL Alphabet dataset[7], we designed a model with 6 convolutional layers, 3 dropout layers, 2 fully-connected layers and a batch size of 64. We used stride values of 1 and 2 alternatively to capture the low-level features of the images. We used ReLU activation functions for all layers along with a filter size of 4. ReLU is one of the best non-linear activation functions which makes the model more interpretable because the values are always greater than or equal to 0. We used softmax loss at the end of the dense layer so the predicted values are interpretable. We used Adam optimizer and Categorical Crossentropy loss and then converted all the images to a size of 64 x 64 finally apply the convolutional layers to the dataset. We obtained an accuracy of about 95.54 and a loss value of 0.1242 after 5 epochs with 1224 steps in each epoch.

Using ASL MNIST dataset[8], we trained a basic CNN on images of size 28 x 28, having 3 convolutional layers in the beginning to learn the features of the images, each followed by Max Pooling layer to reduce the number of parameters to learn and also to help in translation invariance and had 2 fully connected layers at the end. We applied dropout of 0.25 after second and third convolutional layers and also between the two fully connected layer. ReLU activation function was used for all layers in order to bring non-linearity in the model and softmax loss was applied to final fully connected layer. We used categorical cross entropy loss and Adam Optimizer for this model. This architecture gave an accuracy of 94.7 percent in 10 epochs with a batch size of 32.

4.3 Preliminary Results

The deep learning library fastai gives practitioners with high-level components that can provide state-of-the-art results in conventional deep learning domains quickly, as well as researchers with low-level components that can be mixed and matched to create novel techniques[11]. In-order to provide a foundation for modeling a custom dataset and mirroring the kinds of data that should be present in our custom dataset, we utilized fastai for both VGG16 and ResNet50 models.

Fastai's datablock is used to separate the dataset into batches according to their labels. The train-validation split is 80-20 of the training data. The image size used here is 128 x 128. The labels will be the parent folders' names. A batch size of 64 is used here for processing. From vision.models, we use vgg16_bn and resnet50 to train our model. With cross entropy loss and 5 epochs, we get an accuracy of about 96% for VGG16 and about 99% for ResNet50.

Figure 4.3.1 Predictions using ResNet50 on ASL alphabet dataset

We also tried creating multiple versions of the datasets using the OpenCV script with different number of train and test images and finally created the entire dataset used for the final implementation. We initially created images for three classes namely A, B and C with 149 train images in each class and one image for test. We used this dataset to run a simple CNN model that we had as our baseline model and received good accuracy of 99 percent without any pre-processing. The model used for this was trained for a large dataset thus producing such high accuracy.

4.4 Final Results

Our final model utilizing the custom dataset utilized keras ImageDataGenerator with class mode as categorical to divide the entire data of 1200 images for 38 classes with 60-20-20 train-validation-test split. Each image in this dataset has an original size of 422*422 and was resized to try with 224 x 224 and 128 x 128 out of which the latter gave the best results. We trained the model with a batch size of 32 for 10 epochs and received a test accuracy of 94.55 percent. We made use of two Keras callbacks, EarlyStopping and ReduceLROnPlateau. Early stopping was made to monitor validation loss with a patience of 5 and learning rate reduction by a factor of 0.05 was provided for monitoring the validation loss with a patience of 5 and with minimum learning rate value of 0.000002. Table 4.4.1 shows the comparison of the various models we tried for the problem at hand.

Table 4.4.1 Comparison of models

Model	Image size	Batch size	Epochs	Accuracy(%)
Baseline CNN using MNIST	28*28	32	10	94.74
Baseline CNN using Kaggle ASL dataset	64*64	64	5	95.54
VGG16 using Kaggle ASL dataset	128*128	32	5	95.87
ResNet50 using Kaggle ASL dataset	128*128	32	5	98.92
ResNet50 using custom dataset	128*128	32	10	94.55

Figure 4.4.2 and figure 4.4.3 shows the confusion matrix and the accuracy and loss curves obtained using ResNet50 model on custom dataset respectively

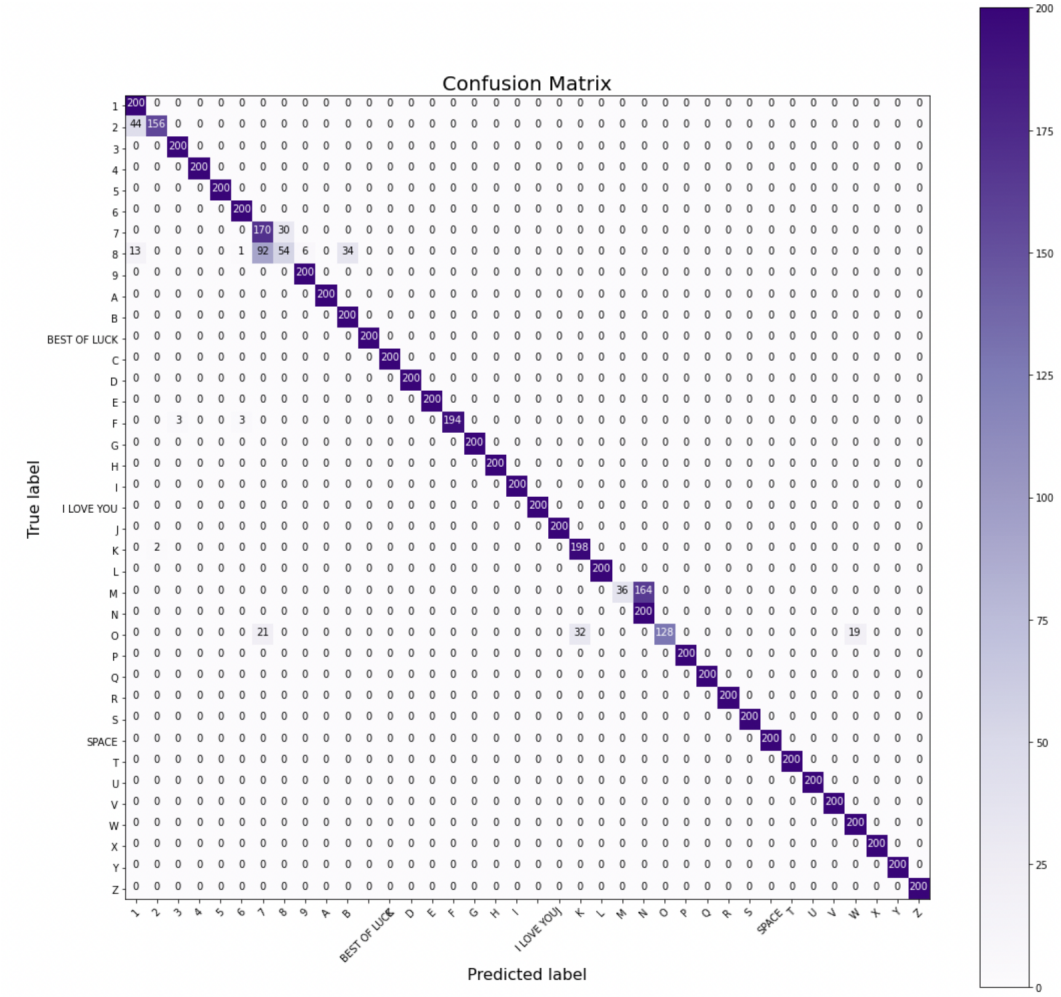

Figure 4.4.2 Confusion Matrix for ResNet50 model on custom dataset

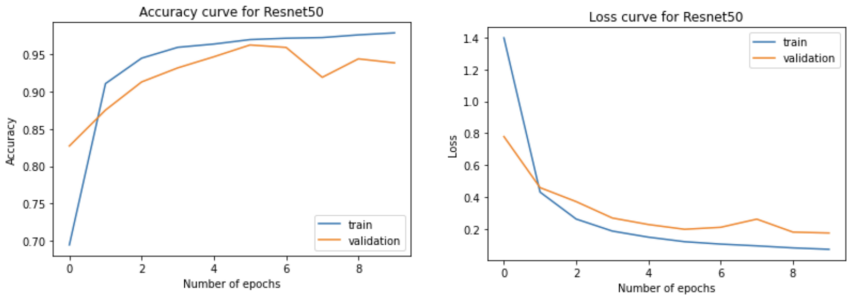

Figure 4.4.3 Accuracy vs Epochs and Loss vs Epochs curves for ResNet50 model on custom dataset

5 Conclusion

5.1 Discussion

Out of the five models we tried on multiple ASL image datasets, we found ResNet to be very efficient. In our final ResNet50 model, we had used a dropout of 0.6 initially which caused the training accuracy to be lesser than validation and test accuracies. We were able to turn this around by reducing the dropout value to 0.2. Similarly, batch size of 32 seemed to give good accuracies than 64.

In our final model, we have images which are very similar to each other with a white background. Since there is not much of a variance, our model would not always generalize well. If more variance is brought into the images with different backgrounds and settings, it would be helpful in training a better model. With such data, better pre-processing techniques of images which include segmentation, data augmentation and adjusting brightness and color intensities can be performed and this would be eventually help in better recognition of images from a live video with continuous hand motions.

5.2 Future Works

As an expansion of this project, we would like to achieve a real time detection system of sign language gestures with the help of OpenCV. This can be further developed to include phrases which require motion and not just still images. We could also enhance this by trying multiple deep learning models like MobileNet, InceptionNet etc that are available currently and go on to implement translation of entire sentences employing the usage of Natural Language Processing methods as well.

6 Contribution of work

Joann worked on dataset generation, baseline CNN modeling using Sign language MNIST dataset, ResNet50 modeling using custom dataset and demo script. Manasa worked on baseline CNN modeling using ASL dataset, VGG16 modeling using ASL dataset and ResNet50 modeling using ASL dataset.

Github Repository: https://github.com/joannjacob/Sign-Language-Detection

7 References

1. Abiyev, Rahib H., et al. "Sign Language Translation Using Deep Convolutional Neural Networks." KSII Transactions on Internet and Information Systems, vol. 14, no. 2, Feb. 2020, pp. 631+. Gale Academic OneFile, link.gale.com/apps/doc/A618573632/AONE?u=mlin_owebsid=googleScholarxid=7ab3cc37. Accessed 24 Mar. 2022
2. Sakshi Sharma, Sukhwinder Singh, Vision-based hand gesture recognition usingb deep learning for the interpretation of sign language, Expert Systems with Applications, Volume 182, 2021, 115657, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2021.115657

3. S. Shahriar et al., "Real-Time American Sign Language Recognition Using Skin Segmentation and Image Category Classification with Convolutional Neural Network and Deep Learning," TENCON 2018 - 2018 IEEE Region 10 Conference, 2018, pp. 1168-1171, doi: 10.1109/TENCON.2018.8650524
4. Chandhini Grandhi, Sean Liu and Divyank Rahoria. American Sign Language Recognition using Deep Learning
5. S. Dinesh, S. Sivaprakash, M. Keshav and K. Ramya. Real-Time American Sign Language Recognition with Faster Regional Convolutional Neural Networks. International Journal of Innovative Research in Science, Engineering and Technology Vol. 7, Special Issue 2, March 2018
6. Sarfaraz Masood, Harish Chandra Thuwal and Adhyan Srivastava. American Sign Language Character Recognition using Convolution Neural Network
7. Akash. (2018 April). ASL Alphabet. Version 1. Retrieved February 20, 2022 from https://www.kaggle.com/grassknoted/asl-alphabet
8. tecperson. (2018 April). Sign Language MNIST. Version 1. Retrieved February 20, 2022 from https://www.kaggle.com/datasets/datamunge/sign-language-mnist
9. joann jacob. (2022 April). American Sign Language Dataset. Version 6. Retrieved May 2, 2022 from https://www.kaggle.com/datasets/joannracheljacob/american-sign-language-dataset
10. Priya Dwivedi. (2019 January). Understanding and Coding a ResNet in Keras. Retrived May 5, 2022 from https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33
11. fastai. (2022 May). Welcome to fastai. Retrieved May 5, 2022 from https://docs.fast.ai/