AMERICAN SIGN LANGUAGE CLASSIFICATION USING DEEP LEARNING

EE 541 – Computational Introduction to Deep Learning

Vaishnavi Channakeshava (USC ID: 96737183)

channake@usc.edu

Sanjana Vasudeva (USC ID: 7071819723)

sanjanav@usc.edu

Table of Contents

1.Introduction

American Sign Language (ASL) is a natural language that is widely used by the Deaf Communities in the United States and Canada as their sign language to communicate, making it the fifth most used language in the United States. In this project. We implement Deep Learning models that can recognize the character of ASL from the image provided. This project is a preliminary step toward developing a sign language translator that will help individuals with hearing impairments communicate better.

Using Convolution Neural Networks (CNN), we have implemented VGG-16 with additional layers, and inceptionv3 with additional layers as part of transfer learning. Along with these models, a simple CNN architecture, ResNet-50 and AlexNet has been developed from scratch. The primary aim of this project is to develop a machine learning model that can recognize the ASL alphabet from the given image.

2.Dataset

The dataset is 1 GB which is a collection of images of alphabets from the American Sign Language available in Kaggle [1]. The training data set contains a total of 87,000 images which are 200x200 pixels each. There are 29 classes each with 3000 images, of which the first 26 classes are for the alphabets A-Z and the remaining 3 classes are for space, delete, and nothing. The test data set contains 29 images. We have additionally added another 29 images to the test dataset, which were captured through a camera. The images of the data set are in .jpg format. Figure 1 represents a sample of the training dataset.

The pre-processing of the images involved resizing the images to the size of the corresponding models. For VGG-16 and ResNet-50, the images have been resized to 224X224X3. For Inceptionv3, the images have been resized to 229X229X3. For AlexNet, the images have been resized to 256X256X3 and the input image size for the CNN model is 200X200X3. Other pre-processing techniques involve horizontal flip (to include both the hands), normalization, contrast, and brightness enhancement.

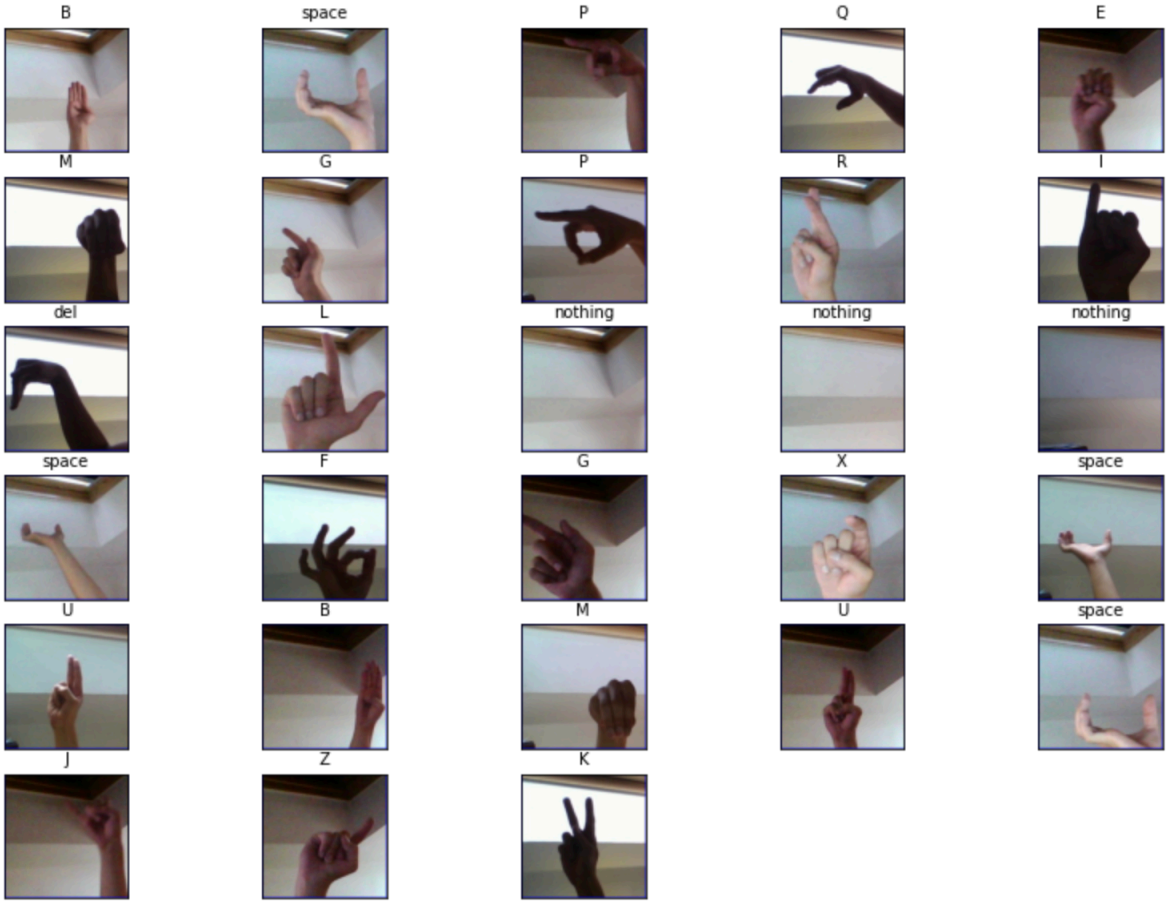


Figure 1: A sample of training data set

3. Related Work

3.1 Sign Language Recognition:

This paper [2] compares 2 models. Firstly, data preprocessing was performed using the PILLOW library. Image and Edge enhancement were performed to help the neural network identify the hand along with the boundaries. A technique using Singular Value Decomposition was used to perform image whitening.

The first model consists of blocks with 2D Convolution Layers with a 3x3 Kernel and a ReLU Activation, followed by a Max Pool and Dropout Layers. The Convolution blocks are repeated 3 times followed by Fully Connected Layers. The second model was built similarly but with only one convolution block and used varying sizes of kernels. In comparison, the latter model performed better than the first model as it did not show any significant decrease in loss. Thus, the second model was trained. The architecture proved to give a better performance when trained on the preprocessed images than on original images.

3.2 Sign Language Recognition with Unsupervised Feature Learning:

This paper [3] provides different CNN architectures to build an ASL Model. The various methods such as Baseline CNN, VGG-16 with pre-trained weights, Inception Net and ResNet50 have been implemented. All models were built using the ReLu activation function and a Softmax Layer. Based on a thorough Experimental analysis of the comparison of the various methods, ResNet50 proved to give the highest accuracy of about 99.88% on the Test Set.

3.3 Using deep convolutional networks for gesture recognition in American sign language:

This paper [4] follows deep convolutional neural networks for the American Sign Language classification problem. The input images were resized to 200x200 followed by preprocessing and data augmentation. The background from the input images was removed using background-subtraction techniques. The Data augmentation included - rotating images by 20 degrees, translating axes, and flipping the images horizontally (to include data from both hands).
The CNN architecture included 3 groups of 2 convolutional layers followed by a max-pool layer and a dropout layer, and two groups of fully connected layers followed by a dropout layer and one final output layer. A categorical cross-entropy loss function was used to train the models. Accuracy of 82.5% was achieved. The performance of the model was hindered by skin color and lighting variation.

3.4 Classification of American Sign Language by Applying a Transfer Learned Deep Convolutional Neural Network:

This paper [5] follows a transfer learning approach to solve the problem of classification of the American Sign Language. The images were resized to 224x224x3 and image augmentation techniques like flip, rotate, scale, crop, translate and zoom were applied.

They started with InceptionV3 of 311 layers, a fully connected layer having 1024 nodes supported by a dropout of 50% was added. This was followed by a fully connected layer of 512 nodes with a 50% dropout. Finally, an output layer was added for classification. While training, no layer was kept frozen. The model was trained for 25 epochs with a batch size of 24. A learning rate of 0.0001 was chosen and the model was trained using categorical cross-entropy. An overall accuracy of 98.81% was achieved.

3.5 Thai sign language recognition by using geometric invariant features and ANN classification:

In this paper [5] a neural network has been designed to classify 42 letters in the Thai Sign Language (TSL). The RGB images were converted to grayscale images. A feed-forward backpropagation training neural network was used for data set classification. It consists of input layers followed by 63-node hidden layers and 42-node output layers which are according to the number of Thai alphabets. An accuracy of 96.19% was achieved.

4 Architecture

In this project, we have utilized pre-trained models VGG-16 and InceptionV3 as the starting point and additional layers have been introduced, which is a part of transfer learning. The project also focuses on implementing ResNet-50, AlexNet, and a CNN model from scratch [9].

4.1 CNN model from scratch

This CNN model[13] that has been created from scratch has two layers followed by a fully connected layer. The first layer has 16 kernels of size 5X5 with a stride of 1 and padding of 2 with batch normalization and a Leaky ReLU activation function. The second layer consists of 32 kernels of size 5X5 with a stride of 1 and padding of 2 with batch normalization, a Leaky ReLU activation function, and max pooling. The network is flattened, followed by dropout and batch normalization. The output layers consist of SoftMax activation over 29 output nodes for classification. The architecture of the CNN model is as shown in Figure 2.

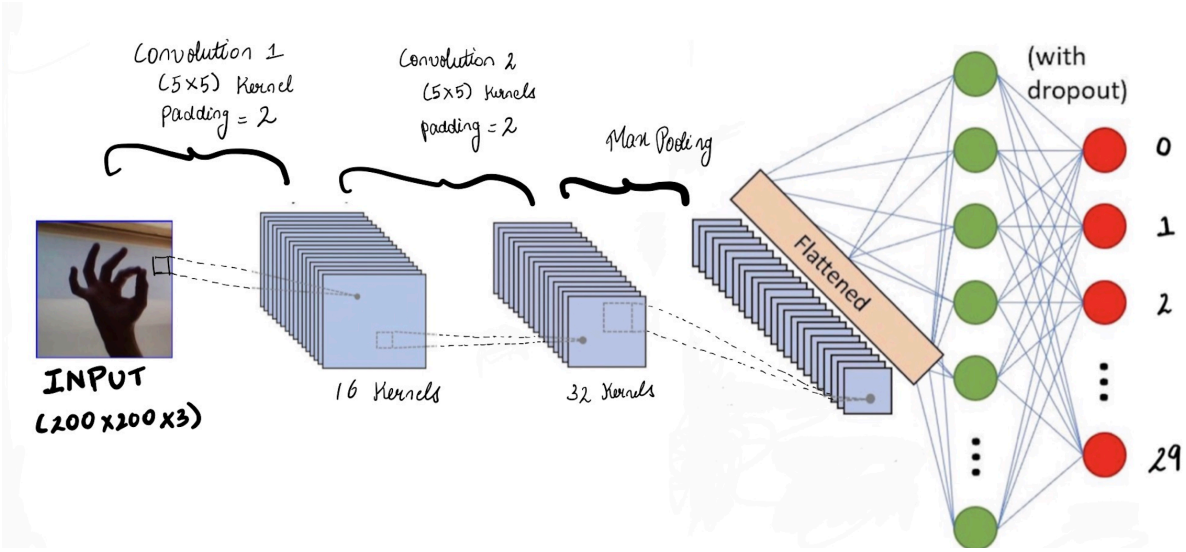


Figure 2: CNN Model Architecture from Scratch

4.2 RESNET-50

RESNET-50 [7] stands for Residual Network which has 50 neural network layers. The network is made up of ResNet blocks and skip connections. The skip connections add the outputs from previous layers to the outputs of stacked layers, making it possible to train deeper networks. They enable the model to learn the identity function and diminish the vanishing gradient problem.3-layer bottleneck blocks form the RESNET-50 architecture. The architecture of the ResNet model is as shown in Figure 3.

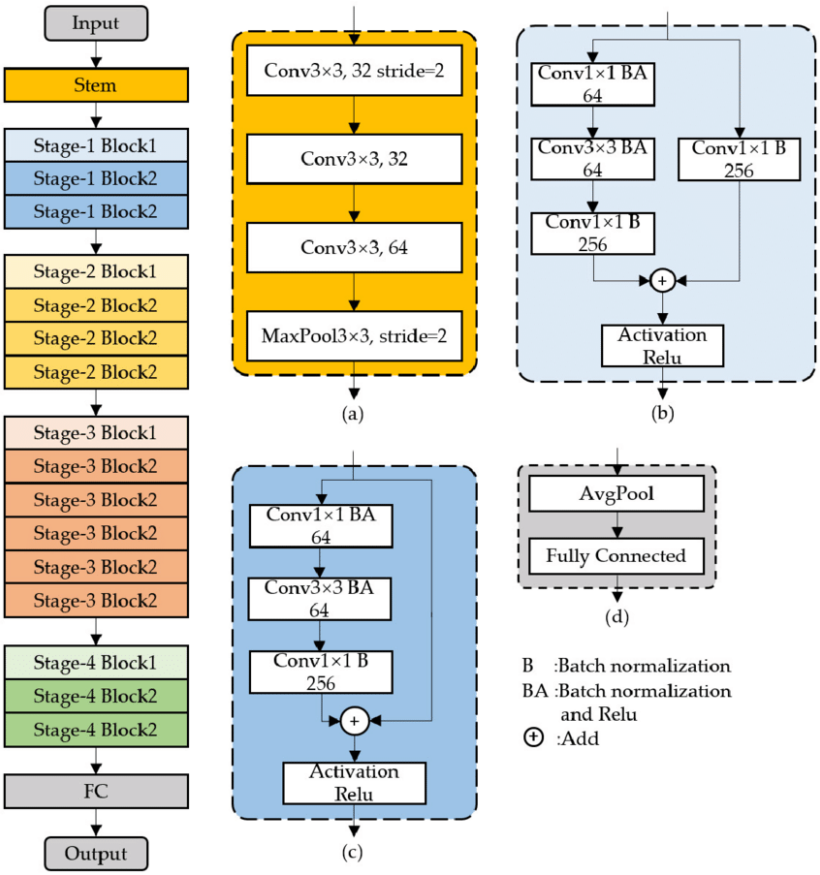


Figure 3: RESNET-50 Architecture

4.3 AlexNet

AlexNet [8] consists of eight layers, where the first five layers are convolution layers that are followed by three fully connected layers. It uses the ReLU activation function and overlapping pooling. It has around 60 million trainable parameters. The architecture of the AlexNet model is as shown in Figure 4.

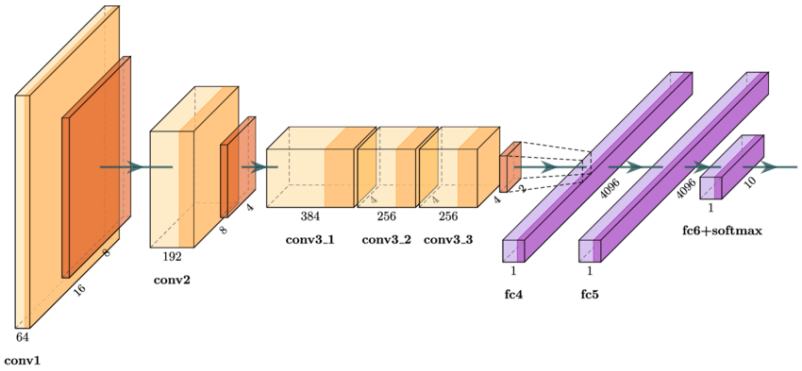


Figure 4: AlexNet Architecture

4.4 Transfer Learning Using VGG-16

VGG-16 [11] is a convolutional neural network that is 16 layers deep. The model [12] is pre-trained on ImageNet weights and consists of convolution layers of 3x3 filter with a stride 1 and always uses same padding and maxpool layers of 2x2 filter of stride 2. The network consists of approximately 138 million parameters. A linear layer with 256 nodes with ReLU activation, followed by a linear layer with 29 nodes with softmax activation has been added to the network. The architecture of VGG-16 model is as shown in Figure 5.

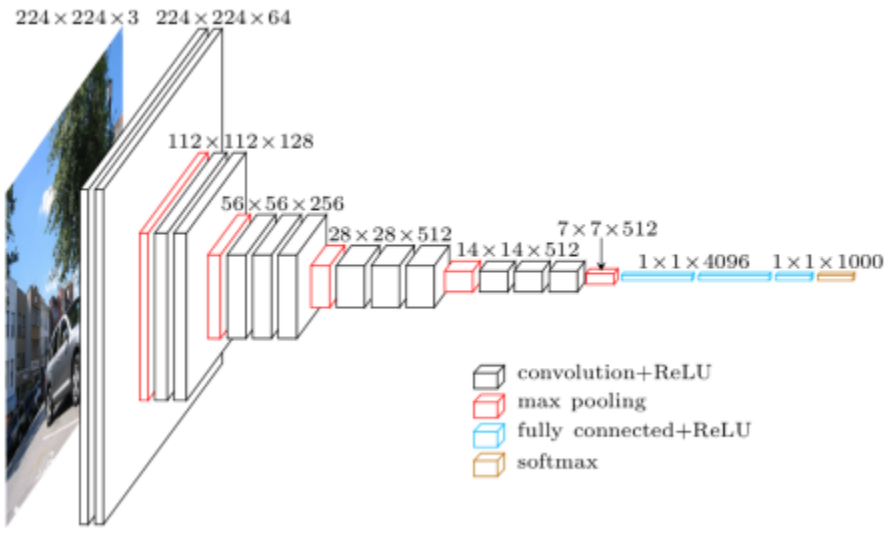


Figure 5: VGG-16 Architecture

4.4 Transfer Learning using INCEPTION v3

The inception V3 model is made of 42 layers and is made up of symmetrical and asymmetrical building blocks. It consists of average pooling, max pooling, and dropouts. Batch normalization is significantly used throughout the model. A linear layer with 1024 nodes with ReLU activation, followed by a linear layer with 29 nodes with softmax activation has been added to the network. The architecture of Inception v3 model is as shown in Figure 6.

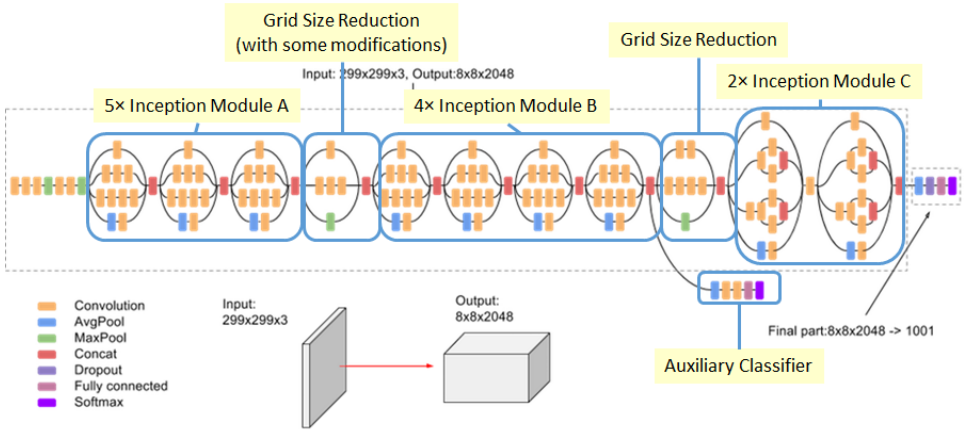


Figure 6: Inception v3 Architecture

5 Training

The models were trained using Google Colab pro with Tesla P100-PCIE-16GB GPU. Various hyperparameters - learning rate, number of epochs, batch size, dropout rate, optimizers, and weight decay. The following parameters were chosen as optimal parameters for all the models as they provided the best results.

VGG-16 and Inception V3 were trained for 15 epochs with a learning rate of 0.001 and Adam optimizer.

The CNN model, ResNet, and AlexNet were trained for 15 epochs with a learning rate of 0.0001 and Adam optimizer with a weight decay rate of 0.0001.

The models have been evaluated based on accuracy.

PARAMETERS	VALUE
Optimizer	Adam
Learning Rate	0.001 / 0.0001
Weight Decay	0.0001
Classifier Output	29

Table 1: Parameters of the model for training

PARAMETER	VALUE
No of GPUs	1
GPU	Tesla P100-PCIE-16GB
GPU Memory	32 GB
System RAM	130 GB
Time To Train	16 min/epoch

Table 2: Hardware Parameters

6 Results

All models were trained for 15 epochs for consistency except for the baseline CNN model which was trained for 30 epochs.

6.1 CNN Model

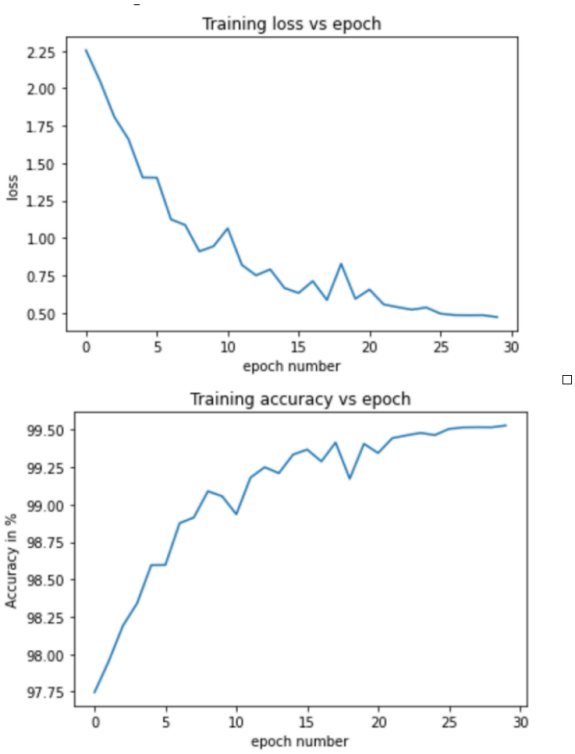


Figure 7: Accuracy and Loss vs epoch for baseline CNN model

The results obtained for training baseline CNN from scratch are presented above in Figure 7. An accuracy of 92% was obtained on the test dataset.

6.2 ResNet Model

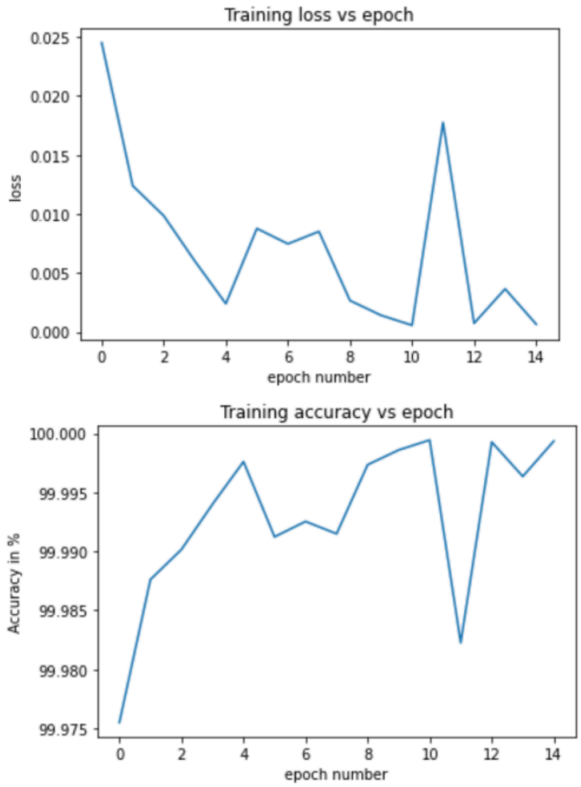


Figure 8: Accuracy and Loss vs epoch for ResNet model

The results obtained for training ResNet from scratch are presented above in Figure 8. An accuracy of 94.7% was obtained on the test dataset.

6.3 AlexNet Model

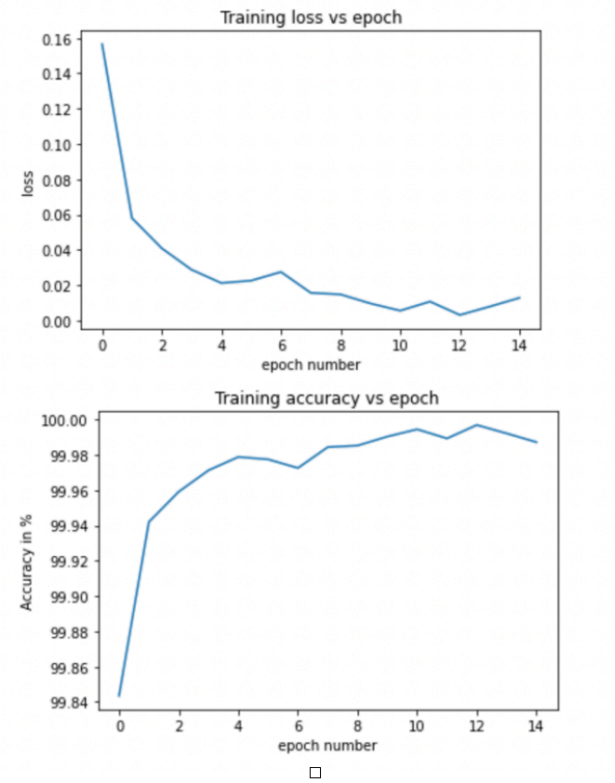


Figure 9: Accuracy and Loss vs epoch for AlexNet model

The results obtained for training AlexNet from scratch are presented above in Figure 9. An accuracy of 95% was obtained on the test dataset.

6.4 VGG16 Model

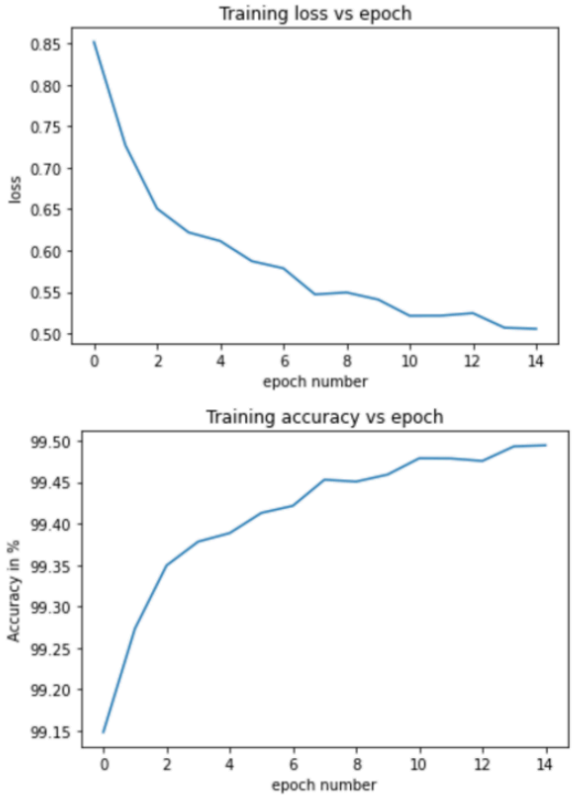


Figure 10: Accuracy and Loss vs epoch for VGG16 model

The results obtained for training VGG 16 through transfer learning are presented above in Figure 10. An accuracy of 93.2% was obtained on the test dataset.

6.5 Inception-v3 Model

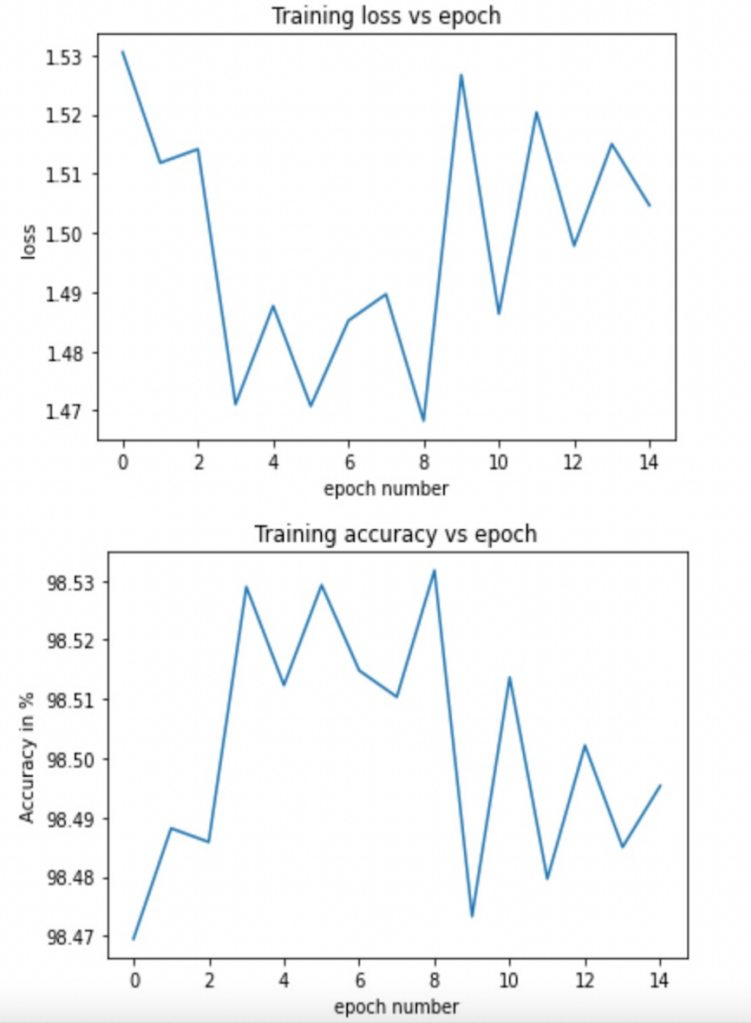
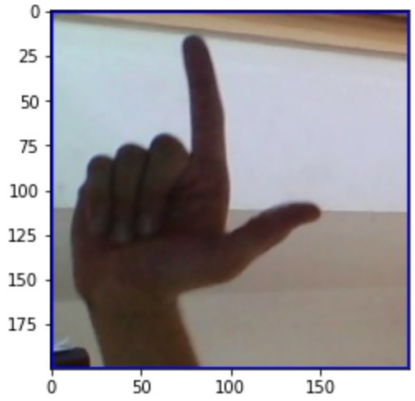


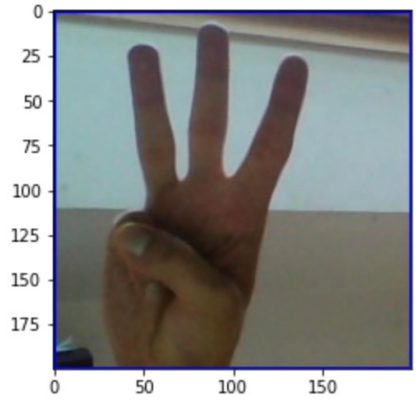
Figure 11: Accuracy and Loss vs epoch for Inception-v3 model

The results obtained for training InceptionNet through transfer learning are presented above in the Figure 11. It can be observed that the minimum test loss was achieved at eight epoch and the loss increases as the number of epochs increases. We implemented early stopping and chose the best model which was obtained after training for 8 epochs. An accuracy of 89.9% was obtained on the test dataset.

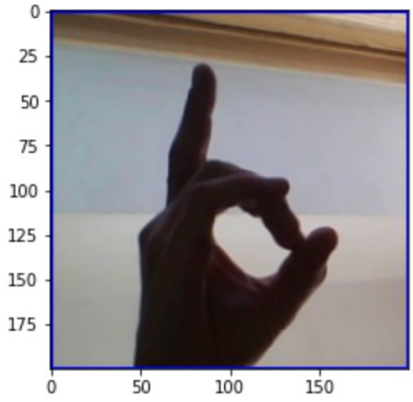
6.6 Successfully Classified ASL alphabets



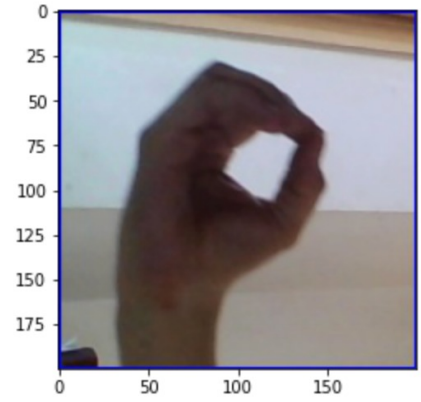
True Label = L and Prediction : L



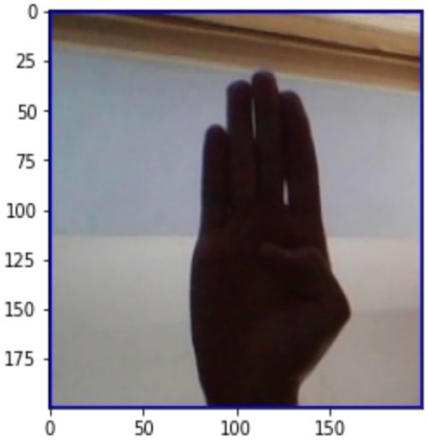
True Label = W and Prediction : W



True Label = D and Prediction : D



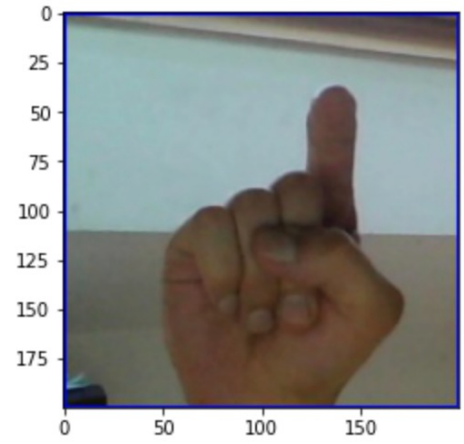
True Label = O and Prediction : O



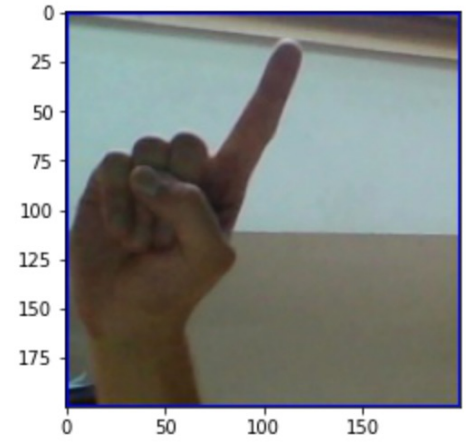
True Label = B and Prediction : B

True Label = G and Prediction : G

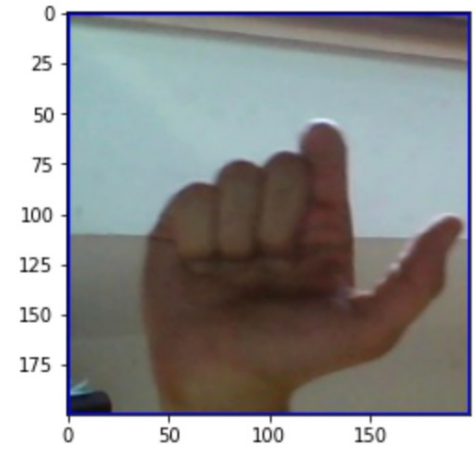
6.7 Incorrectly Classified ASL alphabets



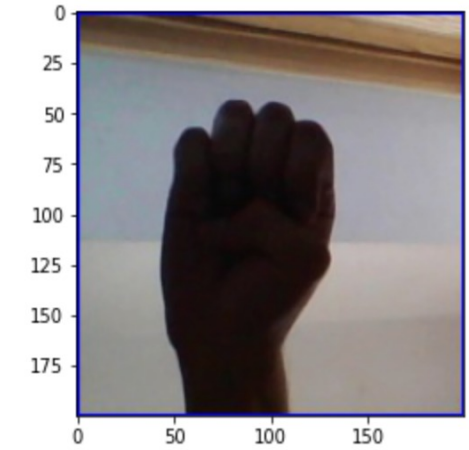
True Label = X and Prediction : U



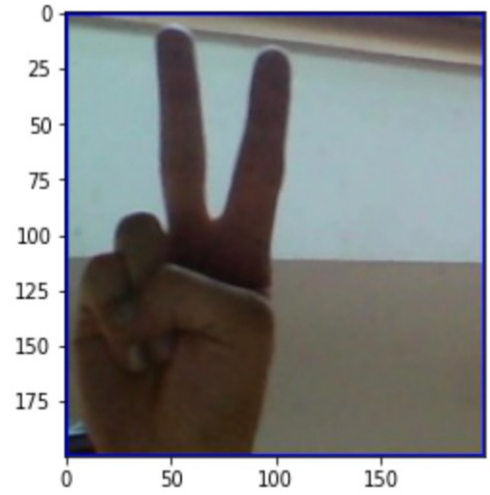
True Label = Z and Prediction : E



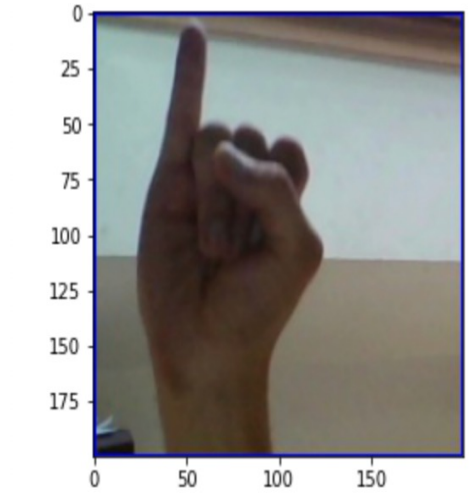
True Label = T and Prediction : J



True Label = E and Prediction : A



True Label = V and Prediction : I



True Label = I and Prediction : X

7 Challenges

- Hardware Limitations: The training of 15 epochs took around 2 hours so using a better GPU would make the process faster.
- Dataset: The test dataset has 29 images which was not representative of the entire dataset, so additional images had to be added to make the test case more robust.

8 Future Work

We want to implement the following as a part of our future work for this project:

- Improving the accuracy of the ASL detection model by fine-tuning hyperparameters and testing various models.
- Extrapolate the project to predict ASL words and sentences.
- Convert the output of the model to speech using speech to text libraries.

9 Applications

The applications of this project include implementing a model that detects the ASL alphabet in real-time. Developing a robot to convey information using ASL.

10 Conclusion

The goal of this project was to find the best model which gives the highest accuracy for the classification of American Sign Language. We have trained 5 models in total, out of which CNN, AlexNet, and Resnet were trained from scratch. We performed transfer learning for 2 models with VGG16 and Inceptionv3 as pre trained models. From the above results we can see that AlexNet model achieved the best results on the test dataset.

11 Evaluation

The results of the evaluation of all 5 models are shown in the below table and can be concluded that AlexNet performed the best for the classification of American Sign Language.

MODEL	TEST ACCURACY
1. CNN Model	92%
2. ResNet Model	94.7%
3. AlexNet Model	95%
4. VGG16 Model	93.2%
5. Inception v3 Model	89.9%

12 References

1. https://www.kaggle.com/datasets/grassknoted/asl-alphabet
2. Deza, Anna, and Danial Hasan. "MIE324 Final Report: Sign Language Recognition."
3. Grandhi, Chandhini and Sean Liu. “American Sign Language Recognition using Deep Learning.”
4. Bheda, Vivek, and Dianna Radpour. "Using deep convolutional networks for gesture recognition in American sign language." arXiv preprint arXiv:1710.06836 (2017).
5. Hasan, Md Mehedi, et al. "Classification of American Sign Language by Applying a Transfer Learned Deep Convolutional Neural Network." 2020 23rd International Conference on Computer and Information Technology (ICCIT). IEEE, 2020.
6. Adhan, Suchin, and Chuchart Pintavirooj. "Thai sign language recognition by using geometric invariant features and ANN classification." 2016 9th biomedical engineering international conference (BMEiCON). IEEE, 2016.
7. https://towardsdev.com/implement-resnet-with-pytorch-a9fb40a77448
8. https://medium.datadriveninvestor.com/cnn-architectures-from-scratch-c04d66ac20c2
9. https://medium.com/@thackerhelik/classification-of-the-american-sign-language-using-pytorch-c62232e2abe3
10. https://towardsdatascience.com/american-sign-language-recognition-using-cnn-36910b86d651
11. https://www.kaggle.com/code/anirbansaha96/asl-vgg19bn-pytorch
12. https://www.kaggle.com/code/anirbansaha96/vgg16-asl
13. https://www.kaggle.com/code/vexxingbanana/asl-alphabet-pytorch