Under the guidance of:

Dr. Preethi Jyothi

Submitted by:

Jeel Manishbhai Nathani (213050036)
Saurish Dilip Darodkar (213050046)
Vivek Raj (213050053)
Smruti Ranjan Behera (213050077)
Gautam Bhavana (213050082)

A PRESENTATION ON ASL RECOGNITION (CS-725 : FOUNDATIONS OF MACHINE LEARNING)

Table of Contents

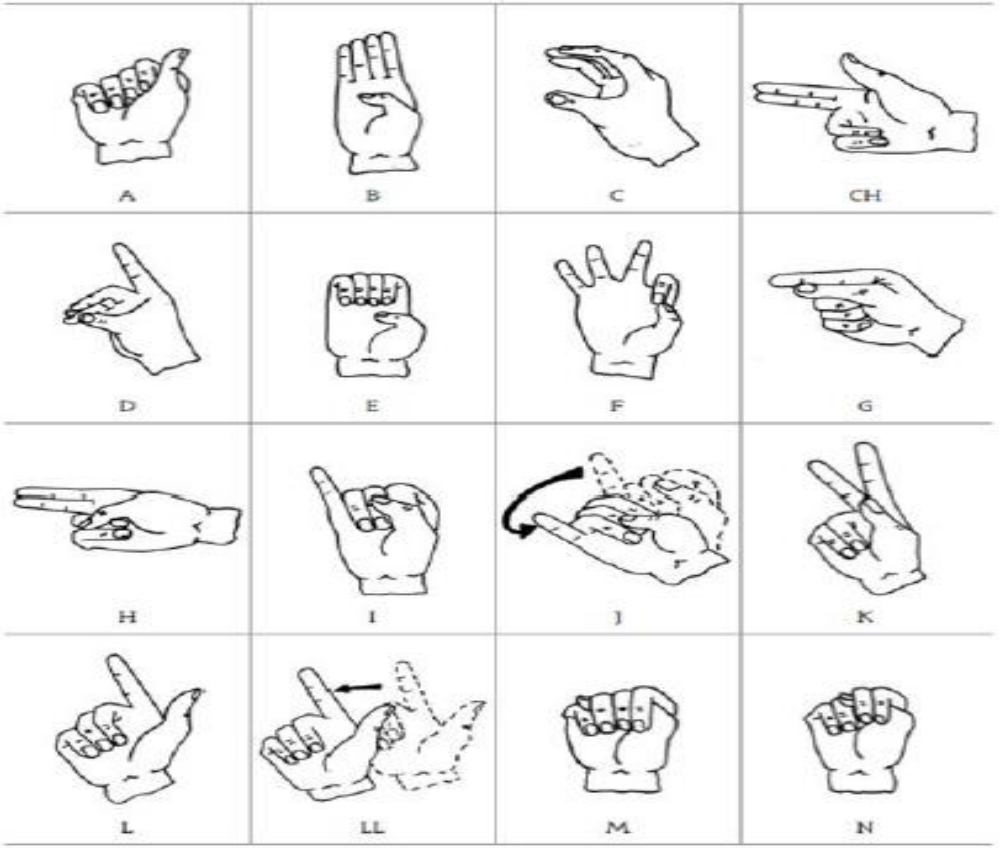
Description of Task

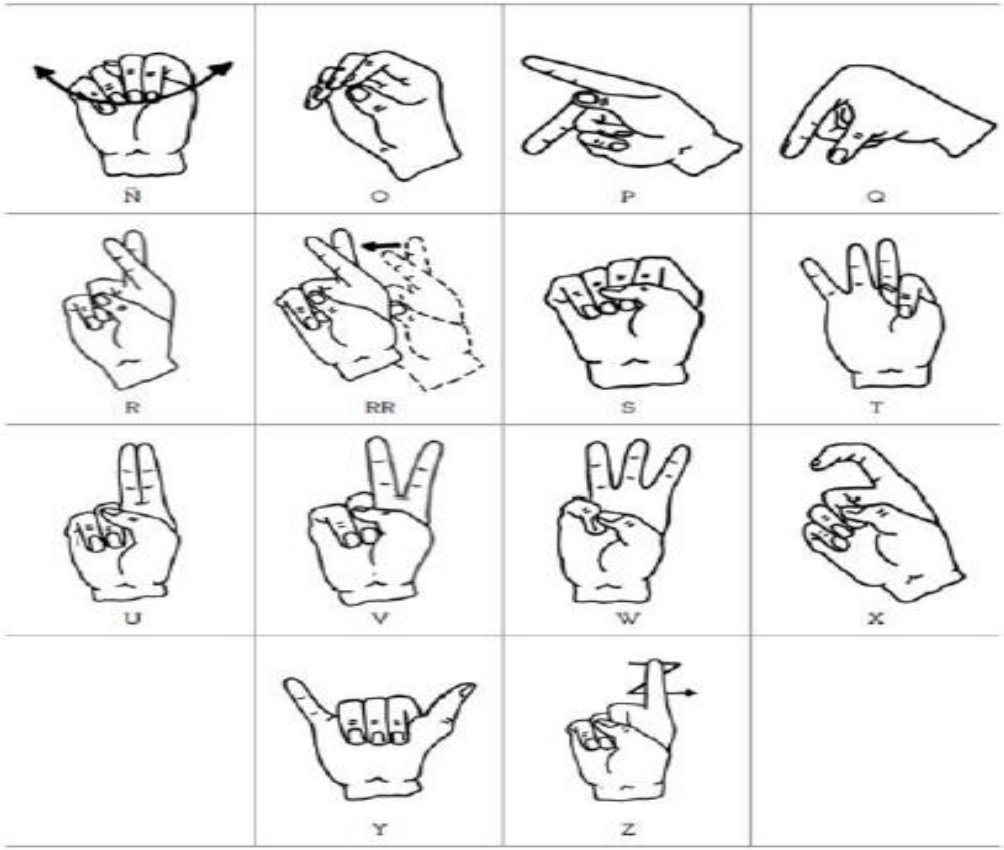
American Sign Language is the primary language used by many deaf individuals. This language is as important as spoken language of deaf people. Here we use hand gestures to classify each letter of American language.

However, as a first step, towards understanding how to build a translation system, we can reduce the size of

the problem by translating individual letters, instead of sentences. Our objective of this project is to have a feasible communication between a normal and a hearing impaired person through ASR.

•In this project, we have trained a convolutional neural network for Image classification of Hand Gestures for American Sign Language . After loading, examining, and preprocessing the data, we trained the network and tested its performance.





Dataset Used

- Dataset 1:
  - https://www.kaggle.com/datamunge/sign-language-mnist

  It has 27,455 images for training and 7172 images for testing. Each 28*28 grayscale image is represented in a tabular format with 784 columns. Number of classes is 24 as letter 'J' and 'Z' are excluded because they require rotation.

- Dataset 2:
  - https://www.kaggle.com/grassknoted/asl-alphabet

  This dataset has 78,000 colour images of A to Z alphabets each with dimension 200 x 200 x 3 . But we have converted this images to grayscale for computation efficiency. For most of the models we resized images to 50 x 50 dimension. But for transfer learning setting, we used images of dimension 75 x 75 x 3 as this is the least dimension that InceptionV3 supports.

Main Techniques Used

Convolutional Neural Network:

CNN is state of the art technique for image classification hence our approach is primarily based on CNN. We used **tensorflow** and **keras** libraries to implement the CNNs. Most of the training is done using ReLU activation function. But we also tried Leaky Relu, tanh sigmoid activation function for the hidden layers.

We always used softmax activation function in the output layer of the neural network.

Main Techniques Used

- Used Various Activation Functions:
  - RELU
  - Leaky Relu
  - Tanh
  - Sigmoid

Transfer Learning:

We used InceptionV3 model for image classification. The model has 315 layers and is trained on imagenet dataset. We can use these pretrained part of these pretrained weights for our classification task. The portion of model to reuse and portion that is to be retrained is treated as hyperparameter in this problem.

Results And Analysis

Techniques Used	Dataset Used	Epoch Used	Number of CNN Layers	Training Accuracy	Test Accuracy
ReLU	Dataset 2	10	4	99.72	99.67
ReLU	Dataset 2	10	3	98.3	97.49
ReLU	Dataset 1	10	3	99.98	85.93
ReLU	Dataset 1	10	4	98.91	99.35
Sigmoid	Dataset 1	50	3	93.4	96.6
TanH	Dataset 1	10	3	94.51	97.4
Leaky ReLU	Dataset 1	10	3	95.27	98.22
Transfer learning	Dataset 2	10	315 (Trainable 5)	68	65.7
Transfer Learning	Dataset 2	10	315 (Trainable 25)	95.6	95.36
Transfer Learning	Dataset 2	10	315(Trainable 55)	94.3	89.13

Best Model

CNN 4 Layer

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 26, 26, 64)        640

 batch_normalization_6 (Batc  (None, 26, 26, 64)       256
 hNormalization)

 conv2d_5 (Conv2D)           (None, 24, 24, 64)        36928

 batch_normalization_7 (Batc  (None, 24, 24, 64)       256
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 12, 12, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 10, 10, 128)       73856

 batch_normalization_8 (Batc  (None, 10, 10, 128)      512
 hNormalization)

 conv2d_7 (Conv2D)           (None, 8, 8, 128)         147584

 batch_normalization_9 (Batc  (None, 8, 8, 128)        512
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 flatten_1 (Flatten)         (None, 2048)              0

 batch_normalization_10 (Bat  (None, 2048)             8192
 chNormalization)

 dense_2 (Dense)             (None, 256)               524544

 batch_normalization_11 (Bat  (None, 256)              1024
 chNormalization)

 dense_3 (Dense)             (None, 26)                6682

=================================================================
Total params: 800,986
Trainable params: 795,610
Non-trainable params: 5,376
_________________________________________________________________
```

Analysis

Convolutional Neural Network:

It is found that 3 to 4 layers of CNN architecture with 3 x 3 filters and maxpooling in between Convolutional layers give the best results without overfitting the data

Analysis of Activation Functions used:

- **ReLU :** ReLU was accurate and fast to converge.
- **Leaky ReLU :** It was most accurate but fastest to ReLU
- **Tanh:** It took a lot of time to converge. Wasn't that accurate
- **Sigmoid:** It also took a lot of time to converge and was the worst in terms of accuracy.

Analysis

Transfer Learning:
When we retrained only 5 layers of the InceptionV3 model we get the worst accuracy. Here model is very rigid to be retrained on the new data

When we retrained later 25 layers of InceptionV3 model, we got 95% accuracy. Though it's the highest accuracy we are able to achieve in the Transfer learning domain, it is still low compared to other model.

When we retrained 55 layers of InceptionV3 model, we got 89% test accuracy. Until now, train and test accuracy were going hand in hand, here we saw some symptoms of overfitting as there were difference in train and test accuracy.

Overall we got less accuracy in transfer learning domain than vanilla CNN domain.

Effect of Number of Layers: More Number of layers meant better accuracy but significant computational overhead. Too much of layers can lead to overfitting.

Analysis

Optimizer Used:

Adam: Combination of the ‘gradient descent with momentum’ algorithm and the ‘RMSP’ algorithm.

Gives high performance with low training cost.

Early stopping Criteria: In order to avoid overfitting, it was used to terminate epochs if certain validation accuracy is achieved.

Dynamic Learning Rate: For better and more accurate convergence.

Contribution

Jeel Manishbhai Nathani (213050036): Model Optimization, Theoretical Interpretation

Saurish Dilip Darodkar (213050046): Model Optimization, Coding

Vivek Raj (213050053): Model Optimization

Smruti Ranjan Behera (213050077): Model Optimization, Dataset Gathering

Gautam Bhavana (213050082): Model Optimization, Documentation

References

- https://towardsdatascience.com/7-popular-activation-functions-you-should-know-in-deep-learning-and-how-to-use-them-with-keras-and-27b4d838dfe6
- https://medium.com/analytics-vidhya/transfer-learning-using-inception-v3-for-image-classification-86700411251b
- https://www.kaggle.com/datamunge/sign-language-mnist
- https://www.youtube.com/watch?v=6Bn0PY_ouBY
- https://www.kaggle.com/gargimaheshwari/asl-recognition-with-deep-learning