Title: American Sign Language Recognition

Name: Valiaparambil Ryan Taffy

Email: ryan19vt@gmail.com

Institute Name: Fr. Conceicao Rodrigues College of Engineering

Under the guidance of

Mr. Utkarsh Sharma

Mr. Bassar Patel

2022-2023

This work is dedicated to my family.

I am very thankful for their motivation and support.

Table of Content

List of Tables

List of Figures

ABSTRACT

According to the World Federation of the Deaf, currently there are more than 70 million people who are deaf worldwide. Collectively, they use more than 300 different sign languages. Sign languages are fully fledged natural languages, structurally distinct from the spoken languages. The gestures or symbols in sign language are organised in a linguistic way. Each individual gesture is called a sign. Sign language helps deaf people to communicate with one another. In this project, I've created a machine learning model that recognizes sign language that is followed by the American Sign Language (ASL) format, from images. The model takes in an image and then predicts which alphabet or number is represented by that gesture. In this way, people who don't understand Sign Language will be able to understand and interpret them using the machine learning model. Hence, this will gap the bridge of communication that exists in today's world.

Keywords:
Sign Language Recognition, American Sign Language, Machine Learning, Deep Learning, Transfer Learning, Inception V3, Neural networks.

INTRODUCTION

As such there is no universal sign language. Different countries adopt different sign languages. In India the Indian Sign Language (ISL) is used widely, in America there is the American Sign Language and so on. Some countries adopt features of American Sign Languages in their standard of sign language. ASL is a language completely separate and distinct from English. It contains all the fundamental features of language, with its own rules for pronunciation, word formation, and word order.

My main idea here is to create a machine learning model which will recognize and predict the sign from the images and display it, so that the user will understand which alphabet or number is shown in the hand sign gesture. Hence people who have not learned ASL will be able to understand what the other person wants to convey. This will help people to know about ASL and it will also create a better environment for the deaf.

PROBLEM STATEMENT AND OBJECTIVES

Problem Statement:

To create a machine learning model that will be able to recognize the hand signs from images in the American Sign Language format.

Objectives:

1. To find a dataset having sufficient images following the ASL format.
2. To study about Neural Networks and how it can be used.
3. To split the dataset into training and testing sets.
4. To learn about Transfer Learning.
5. To create a model which gives a good accuracy.
6. To test the model's accuracy on unseen(test) data.

DIAGRAMS

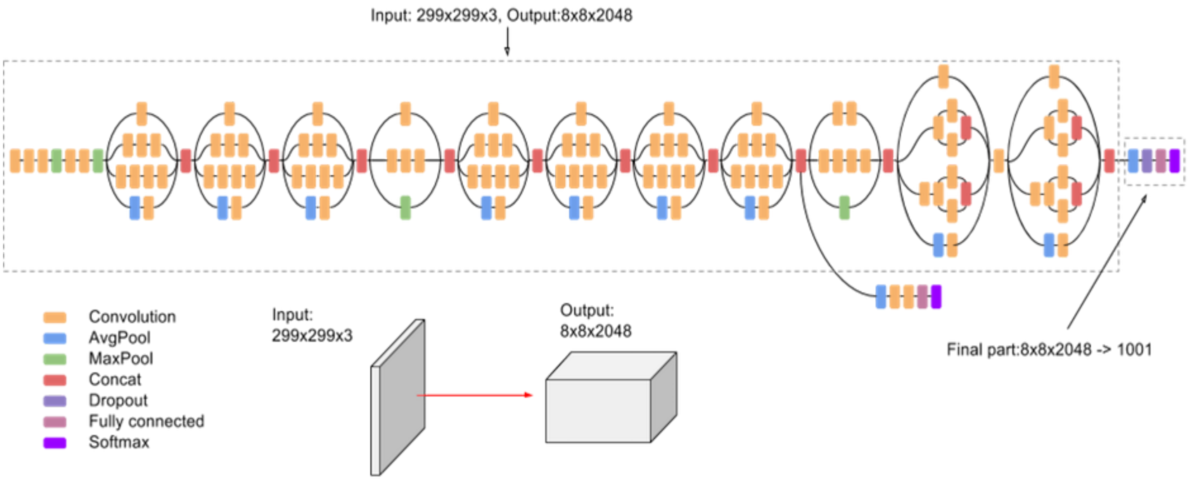

Fig 1. Inception V3 Architecture

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 max_pooling2d_4 (MaxPooling  (None, 2, 2, 2048)       0
 2D)

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 36)                294948

=================================================================
Total params: 22,097,732
Trainable params: 294,948
Non-trainable params: 21,802,784
_________________________________________________________________
```

Fig 2. Architecture of the Final Model

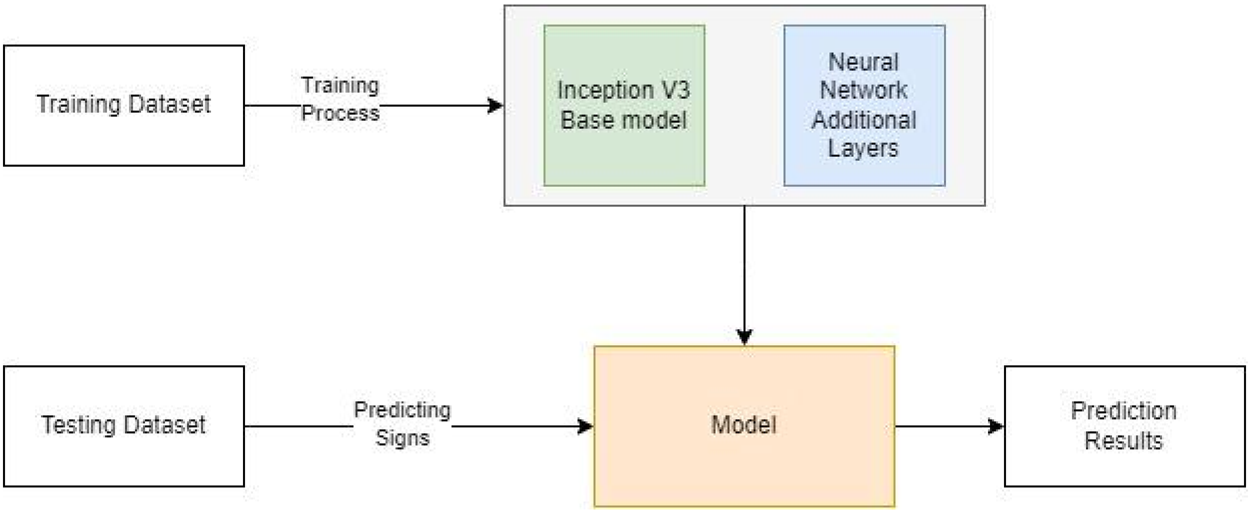

Fig 3. Block Diagram of the Model

SOFTWARE USED

Name	Version
Python	**3.10**
Tensorflow	**2.9**
Keras	**2.9**
Matplotlib	**3.1**
Seaborn	**0.12**

Table 1. Softwares and Libraries used

IMPLEMENTATION

Giving the path for the training and testing dataset. The training set contains 2012 images and the testing set contains 503 images.

Setting the path of the training and testing dataset

```
In [2]: train_path = "dataset/train"
        test_path = "dataset/test"
```

Performing data augmentation

Using ImageDataGenerator we rescale the images and and also artificially create different training and testing images through different ways of processing like shear and zoom. This introduces a sort of randomness in the dataset.

```
In [7]: train_datagen = ImageDataGenerator(rescale = 1/255,
                                           shear_range=0.2,
                                            zoom_range=0.2)

        test_datagen = ImageDataGenerator(rescale = 1/255,
                                          shear_range=0.2,
                                           zoom_range=0.2)
```

```
In [8]: train_set = train_datagen.flow_from_directory(train_path,
                                                      target_size = (224, 224),
                                                      batch_size = 32,
                                                      class_mode = 'categorical')
        test_set = test_datagen.flow_from_directory(test_path,
                                                    target_size = (224, 224),
                                                    batch_size = 32,
                                                    class_mode = 'categorical')

Found 2012 images belonging to 36 classes.
Found 503 images belonging to 36 classes.
```

Fig 4. Training and Testing data

Storing the label names in a list.

```
label_names = ['0', '1', '2', '3', '4', '5',
               '6', '7', '8', '9', 'A', 'B',
               'C', 'D', 'E', 'F', 'G', 'H',
               'I', 'J', 'K', 'L', 'M', 'N',
               'O', 'P', 'Q', 'R', 'S', 'T',
               'U', 'V', 'W', 'X', 'Y', 'Z']
```

Fig 5. Class names

Plotting sample images from the training dataset.

```python
imgs, labels = next(iter(train_set))
counter = 1
for img, label in zip(imgs, labels):
    plt.subplot(5,5,counter)
    plt.subplots_adjust(right=5, top=5, wspace=0.5, hspace=0.5)
    value=np.argmax(label)
    labelname=label_names[value]
    plt.imshow(img)
    plt.title("Image of: "+labelname, fontdict={'fontsize': 25})
    counter+=1
    plt.axis("off")
    if(counter>10):
        break

plt.show()
```

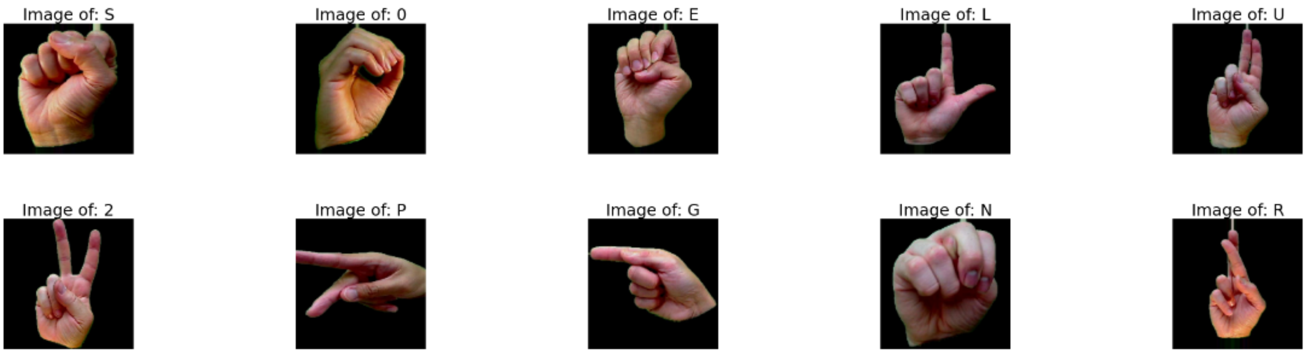

Fig 6. Images in Training set

Loading the InceptionV3 model as the base of our model and then making the layers trainable as false so that it doesn't retrain and keep the same weights as before.

```python
base_model = InceptionV3(input_shape=(224,224,3),
                         include_top=False,
                         weights = "imagenet")
```

```python
base_model.trainable = False
```

Fig 7. Loading InceptionV3 model

Adding a few more layers to the base model to create our own model.

```
model = Sequential([
    base_model,
    MaxPooling2D(),
    Flatten(),
    Dense(36, activation="softmax")])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 max_pooling2d_4 (MaxPooling  (None, 2, 2, 2048)       0
 2D)

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 36)                294948

=================================================================
Total params: 22,097,732
Trainable params: 294,948
Non-trainable params: 21,802,784
_________________________________________________________________
```

Fig 8. Adding Layers

Compiling and fitting the model

```
model.compile(optimizer=Adam(learning_rate = 0.01),
              loss = CategoricalCrossentropy(),
              metrics = [CategoricalAccuracy()])
```

```
model.fit(train_set,
         validation_data = test_set,
         steps_per_epoch = 32,
         epochs = 32)
```

Fig 9. Compiling and Fitting the model

RESULTS

Training and Testing Loss graph

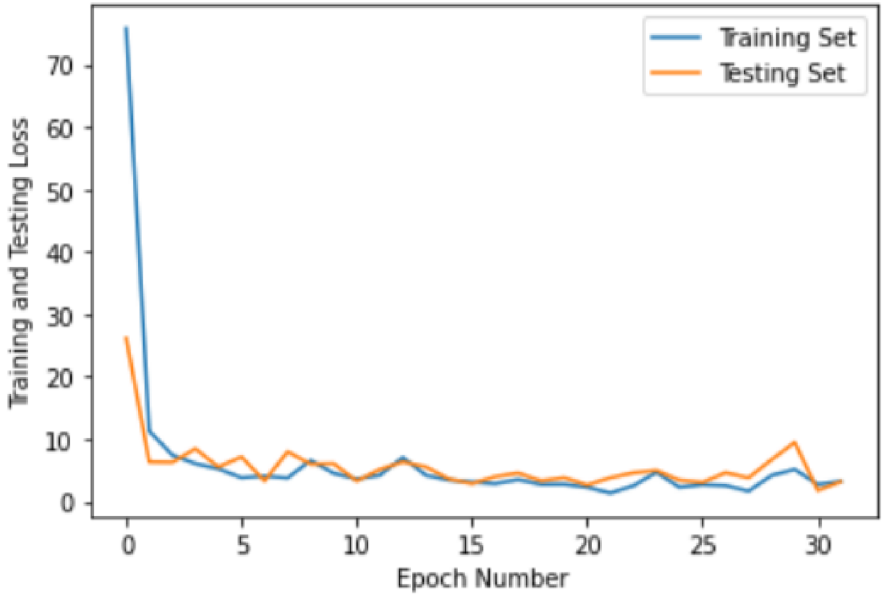

Fig 10. Loss Graph

Training and Testing Accuracy graph

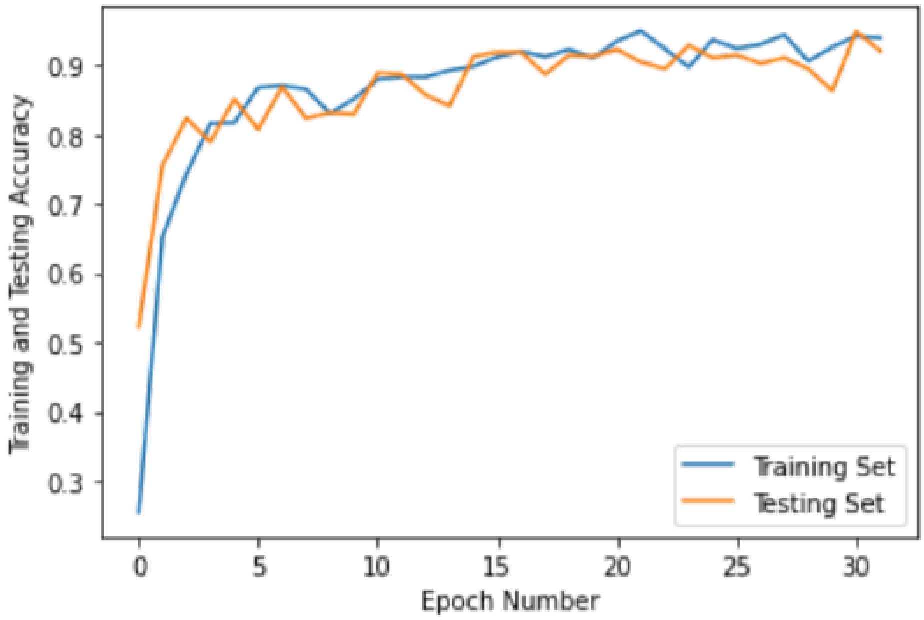

Fig 11. Accuracy Graph

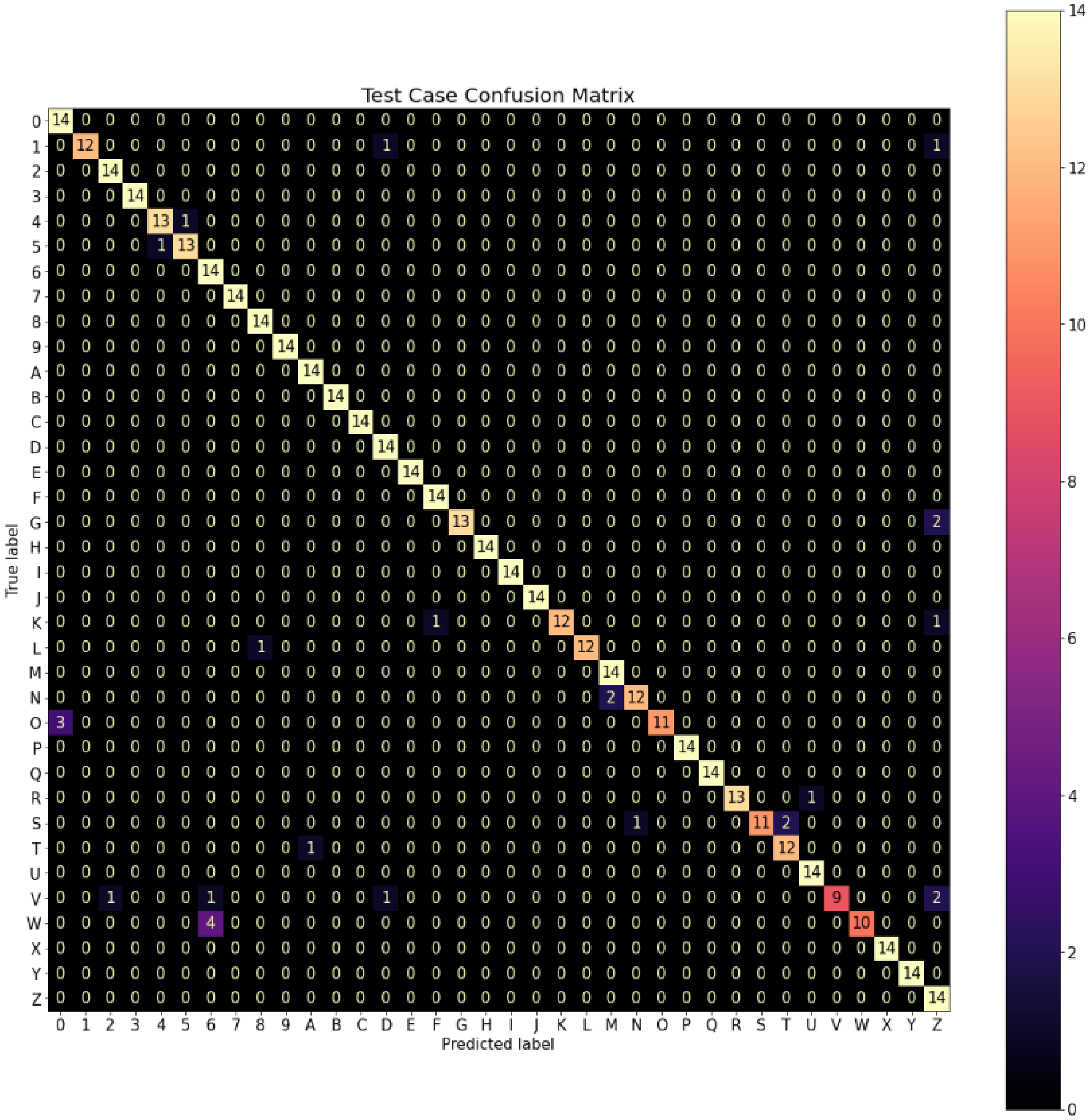

Fig 12. Confusion Matrix

Model	Training Accuracy	Testing Accuracy	Training Loss	Testing Loss
	93.92	**92.64**	**3.24**	**3.91**

Table 2. Results

CONCLUSION

To conclude, I have successfully created a model that can recognize the signs with an accuracy of 93%. This model takes in the images of the sign and then predicts the output based on the American Sign Language. The model was trained for 32 epochs and the Adam optimizer was used to compile the model. Categorical cross-entropy was used to calculate the loss and Categorical accuracy was used to calculate the accuracy. To predict hand signs in different images, the images must be added in a folder and that path should be specified. Using evaluate and predict methods, we can understand what gesture was made.

FUTURE SCOPE

The system can be further optimised and improved.

- Sequence of images can be passed to the model and the entire word could be recognized and shown as a list in the output.
- Live video recognition could be done through an external camera.
- The hand gesture can be highlighted and its class can be shown beside it on the screen.
- Multiple hand signs could be recognized at the same time.
- Addition of other sign language standards like ISL and BSL could be included.

REFERENCES

1. F. R. A. N. C. O. I. S. CHOLLET, *Deep learning with python*. S.l.: O'REILLY MEDIA, 2021.

2. D. Patel, "Deep learning with tensorflow 2.0, Keras and python," *YouTube*. [Online]. Available: https://www.youtube.com/playlist?list=PLeo1K3hjS3uu7CxAacxVndI4bE_o3BDtO. [Accessed: 17-Oct-2022].

3. Simplilearn, "AI and Machine Learning Full Course | Artificial Intelligence & Machine Learning course |simplilearn," *YouTube*, 12-May-2021. [Online]. Available: https://www.youtube.com/watch?v=wnqkfpCpK1g. [Accessed: 20-Oct-2022].

4. D. Patel, "Machine learning tutorial python | machine learning for beginners," *YouTube*, 2018. [Online]. Available: https://www.youtube.com/playlist?list=PLeo1K3hjS3uvCeTYTeyfe0-rN5r8zn9rw. [Accessed: 15-Oct-2022].

5. "(PDF) introduction to artificial neural networks - researchgate." [Online]. Available: https://www.researchgate.net/publication/5847739_Introduction_to_artificial_neural_networks. [Accessed: 17-Oct-2022].

6. "A study on CNN transfer learning for Image Classification - Researchgate." [Online]. Available: https://www.researchgate.net/publication/325803364_A_Study_on_CNN_Transfer_Learning_for_Image_Classification. [Accessed: 01-Nov-2022].