

Project **Report**

1. INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY

- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming

4. REQUIREMENT ANALYSIS

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5. PROJECT DESIGN

- 5.1 Data Flow Diagrams & User Stories
- 5.2 Solution Architecture

6. PROJECT PLANNING & SCHEDULING

- 6.1 Technical Architecture
- 6.2 Sprint Planning & Estimation
- 6.3 Sprint Delivery Schedule

7. CODING & SOLUTIONING

8. PERFORMANCE TESTING

- 8.1 Performance Metrics

9. RESULTS

- 9.1 Output Screenshots

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

- Source Code

GitHub & Project Demo Link

1. INTRODUCTION:

1.1 PROJECT OVERVIEW

The ASL Alphabet Image Recognition project uses technology to bridge the communication gap between the deaf and hearing cultures. American Sign Language (ASL) is the predominant language of deaf people in North America, and it is based on visual cues such as hand gestures, facial expressions, and body movements. There has been an increasing interest in developing technical solutions to ease communication across various populations in recent years. The project focuses on ASL Alphabet Image Recognition, which is an image classification challenge. The primary goal is to develop a machine learning model capable of reliably categorizing photos exhibiting hand signs matching to the English alphabet's 26 letters. In addition, three additional classes are included for the signs "space," "delete," and "nothing."

1.2 PURPOSE

The ASL Alphabet Image Recognition project aims to empower persons in the deaf community by leveraging machine learning and computer vision technology. The project's goal is to contribute to the creation of applications that can be utilized in real-time communication scenarios by developing a model capable of accurately detecting ASL alphabet signs from photographs.

The major goal is to reduce communication obstacles and improve interactions between deaf and hearing people. The ASL Alphabet Image Recognition model can be linked into applications that allow for real-time translation of hand signs, allowing the deaf community to communicate more effectively. This technology not only encourages diversity, but it also has the potential to be a useful tool in educational settings.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

The existing issue is the continuing communication barrier between the deaf and hearing communities. Despite the fact that American Sign Language (ASL) is the primary language of deaf people, there is a dearth of effective tools to bridge the divide. Manual interpretation and traditional procedures frequently result in misconceptions, impeding efficient communication. Current technology has yet to adequately solve real-time ASL interpretation, restricting accessibility and inclusivity for the deaf. The lack of powerful image recognition technologies for ASL alphabets impedes the development of applications critical to developing seamless

communication and understanding across these communities.

2.2 REFERENCES

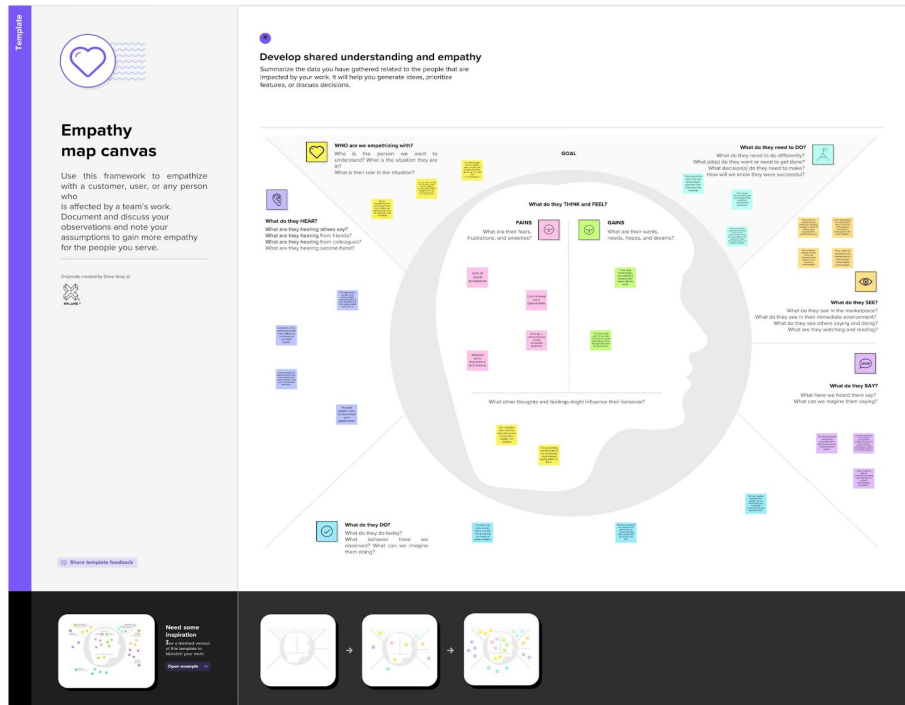
1. https://www.researchgate.net/publication/331854265_Alphabet_Sign_Language_Image_Classification_Using_Deep_Learning
2. https://www.irjmets.com/uploadedfiles/paper//issue_10_october_2023/45608/final/fin_irjmets1698812812.pdf
3. Rivera-Acosta M, Ortega-Cisneros S, Rivera J, Sandoval-Ibarra F. American Sign Language Alphabet Recognition Using a Neuromorphic Sensor and an Artificial Neural Network. *Sensors* (Basel). 2017 Sep 22;17(10):2176. doi: 10.3390/s17102176. PMID: 28937644; PMCID: PMC5677181.

2.3 PROBLEM STATEMENT DEFINITION

The ASL Alphabet Image Recognition project uses machine learning to bridge the communication gap between the deaf and hearing cultures. The difficulty is in teaching the model to correctly classify hand signs that match the English alphabet and special symbols. The problem statement focuses on developing a dependable picture recognition system capable of real-time interpretation, allowing the deaf to communicate more effectively. The goal is to create applications that allow for seamless ASL letter identification from uploaded image files and live video broadcasts, giving a practical solution for increasing inclusion and understanding among people with different communication needs.

3. IDEATION AND PROPOSED SOLUTION

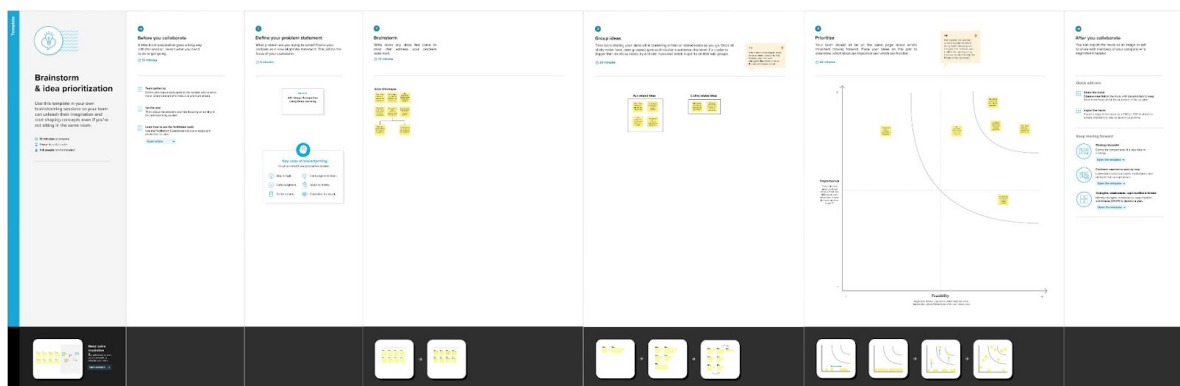
3.1 EMPATHY MAP CANVAS



Google drive link:

<https://drive.google.com/file/d/16qSLK7BQUqYTKN1vadI69DKsnS0IqCz/view?usp=sharing>

3.2 IDEATION AND BRAINSTORMING



Google drive link:

https://drive.google.com/file/d/1TB_F7yk6BSPKU1eug0u8_HCeXBCSXWJJ/vi ew?usp=sharing

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

The functional requirements for ASL- Alphabet Image Recognition project include:

- User-Friendly Application Interface
- Easy to navigate interface
- Quick responsiveness
- Scalability
- Cross Platform Compatibility
- Model Accuracy
- Input Handling

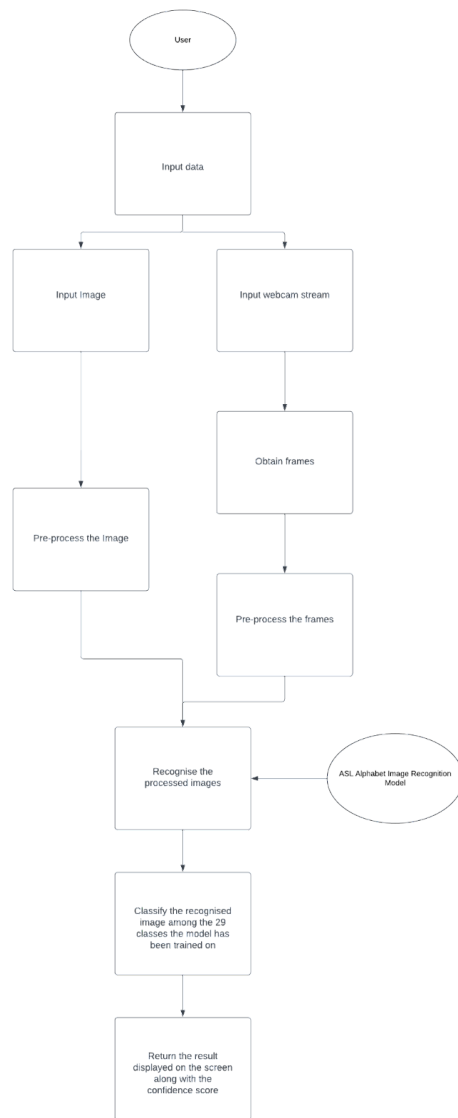
4.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements for ASL- Alphabet Image Recognition project include:

- Performance- Response Time and throughput
- Accuracy
- Reliability
- Security
- Interoperability
- Usability
- Maintainability
- Availability

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS AND USER STORIES



Flow:

1. The user can choose the input option as uploading an image file or by using the webcam stream.
2. In both cases the images or frames are pre-processed(converted into arrays)
3. This pre-processed data is fed to the model
4. The model that is trained on the dataset, works to recognise the image
5. The model then classifies the image among the 29 classes of the dataset
6. The model produces an output which is displayed in the user-interface

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Uploading an image for recognition	USN-1	As a user, I can upload an image from my computer into the application successfully	I can upload an image	High	Sprint-1
		USN-2	As a user, I can confirm which image is uploaded because when I upload an image by displaying the name of the chosen file	I can confirm image selection	Low	Sprint-1
		USN-3	As a user, I can upload images using any type of browsers	I can upload images across any browser	Low	Sprint-2
	Using Webcam Stream for Live Recognition	USN-4	As a user, I can use my webcam to share my video stream successfully.	I can use my webcam to share my live video stream	High	Sprint-1

Customer	Result after uploading an image	USN-5	As a user, I get a quick and accurate result after clicking on the "Predict!" button	I get a recognition result after uploading an image	High	Sprint-1
	Result after sharing live webcam video stream	USN-6	As a user, I can share a live video stream via a webcam and get a quick low latency prediction result along with the confidence scores.	I get a recognition result after sharing my webcam live video stream.	High	Sprint-1

5.2 SOLUTION ARCHITECTURE

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- The solution is designed to help provide a simple, easy to use and reliable way to bridge the communication gap between ASL users and non-users. The solution provides an interactive user interface through a flask application using which the users can either provide an image of an ASL alphabet and recognize it or turn on their webcam in order to live recognize the webcam feed.
- The model is developed using Deep Learning methods such as CNN and VGG-16 and the best performing model is utilized. The model is then trained using pre-processed images from the given dataset. This model is then saved and used in the flask application. The flask app is developed using HTML, Javascript, CSS and python. The flask app allows the user to provide an input and the input image or frame is processed and is sent to the server where it hits the model and returns a prediction which is displayed in the application as a result.

- The model gives accurate results and the features of our application include a user-friendly interface, easy to navigate design, and provides the users with options to either upload an image or use live recognition. The application can be used to identify 26 ASL alphabets along with “space”, “delete” and “nothing” symbols. It displays a result along with the confidence score. The model is scalable and secure to use.

The development of the above solution follows a phase-wise approach. Firstly, we understand the problem statement, empathize with the users and ideate our solution that can effectively return the output. Then we plan the development of the idea and then we break it down into sub-tasks including model development, model training, fine-tuning the model, and flask application development. The final solution is then given by integrating all the above modules.

The solution requires to develop an accurate ASL model, a low-latency response for real-time video image recognition. The solution can be run on any browser and requires a functional webcam to run.

Solution Architecture Diagram:

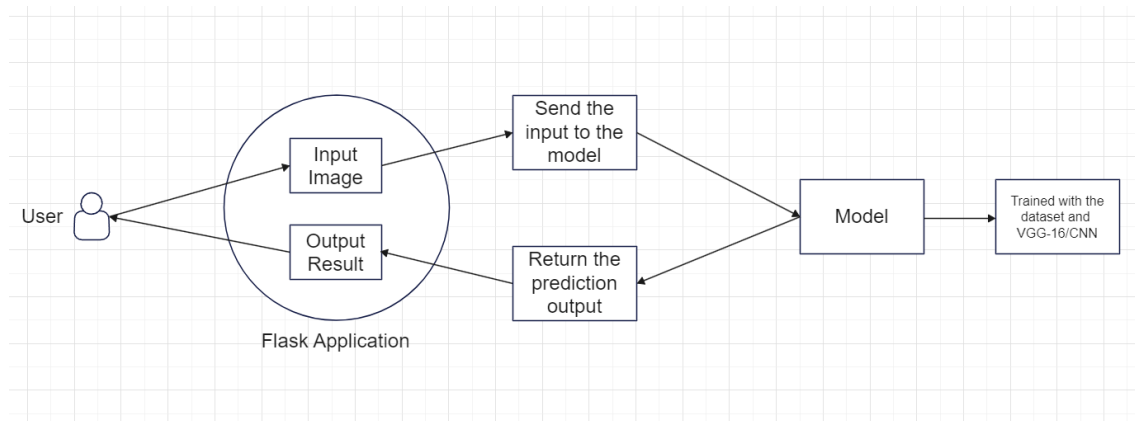


Figure : Architecture and data flow of the ASL- Alphabet Image Recognition application

6. PROJECT PLANNING AND SCHEDULING

6.1 TECHNICAL ARCHITECTURE

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table2

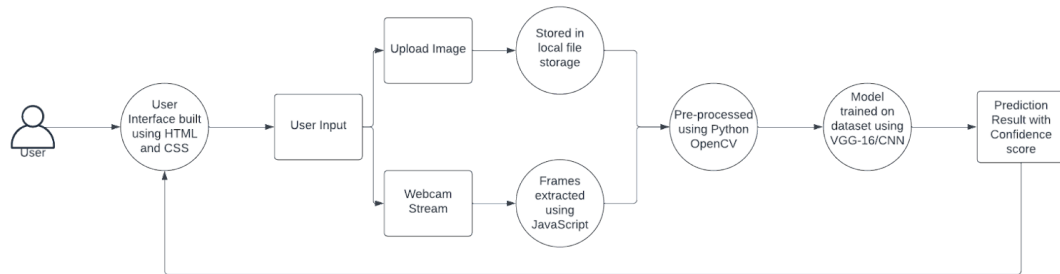


Figure: Technical Architecture Diagram for ASL- Alphabet Image Recognition Flask Web Application

Table-1 : Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	An interface that allows that allows the user to interact with the Flask Web Application	HTML, CSS, JavaScript
2.	Application Logic-1	Logic for Uploading an Image	Python
3.	Application Logic-2	Logic for using live webcam streaming	Python
4.	Application Logic-3	Logic for frame extraction in webcam streaming	Python
5.	Application Logic-4	Logic for sending images and frames to server	JavaScript
6.	Application Logic-5	Logic for creating the Model	Python
7.	File Storage	File storage requirements	Local Filesystem
8.	External API-1	Used to download dataset and train the model	Kaggle API
9.	Machine Learning Model	Model trained on an ASL dataset containing 29 classes used to recognise an input ASL Sign	Image Recognition and classification Model

10.	Infrastructure (Server / Cloud)	Application Deployment on Local System	Local System
-----	---------------------------------	----------------------------------------	--------------

Table-2: Application Characteristics:

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	TensorFlow, Keras, OpenCV, Flask, NumPy, Socket.IO, Bootstrap, jQuery, Git	<p>TensorFlow, Keras: Deep Learning APIs used to train model</p> <p>OpenCV: Computer Vision software library used to process images and frames</p> <p>Flask: A micro web framework used to develop a web app using Python</p> <p>Socket.IO: A library that enables real time communication between server and browser used for webcam live stream recognition</p> <p>Bootstrap: Front-end framework that makes an app responsive used to get interactive responses</p> <p>jQuery: A JavaScript library used to receive data asynchronously, handle events and maintain compatibility across multiple browsers.</p>

6.2 SPRINT PLANNING AND ESTIMATION

Product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Member
Sprint-1	Uploading an image for recognition	USN-1	As a user, I can upload an image from my computer into the application successfully	2	High	Sriya Chinmayee
Sprint-1		USN-2	As a user, I can confirm which image is uploaded because when I upload an image by displaying the name of the chosen file	1	Low	Sriya chinmayee
Sprint-2		USN-3	As a user, I can upload images using any type of browsers	2	Medium	Sriya Chinmayee
Sprint-2	Using Webcam Stream for Live Recognition	USN-4	As a user, I can use my webcam to share my video stream successfully.	2	High	Sriya Chinmayee

Sprint-1	Result after uploading an image	USN-5	As a user, I get a quick and accurate result after clicking on the "Predict!" button	1	High	Sriya Chinmayee
Sprint-3	Result after sharing live webcam video stream	USN-6	As a user, I can share a live video stream via a webcam and get a quick low latency prediction result along with the confidence scores.	2	High	Sriya Chinmayee

6.3 SPRINT DELIVERY SCHEDULE

Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	11th Nov 2023	17th Nov 2023	19	18th Nov 2023
Sprint-2	20	6 Days	15th Nov 2023	21st Nov 2023	18	23rd Nov 2023
Sprint-3	20	4 Days	21st Nov 2023	25th Nov 2023	19	26th Nov 2023

Sprint Average Velocity:

$$\text{Sprint-1} = 19/6 = 3.167$$

$$\text{Sprint-2} = 18/6 = 3$$

$$\text{Sprint-3} = 19/4 = 4.75$$

$$\text{Average Velocity} = (\text{Sprint-1} + \text{Sprint-2} + \text{Sprint-3})/3$$

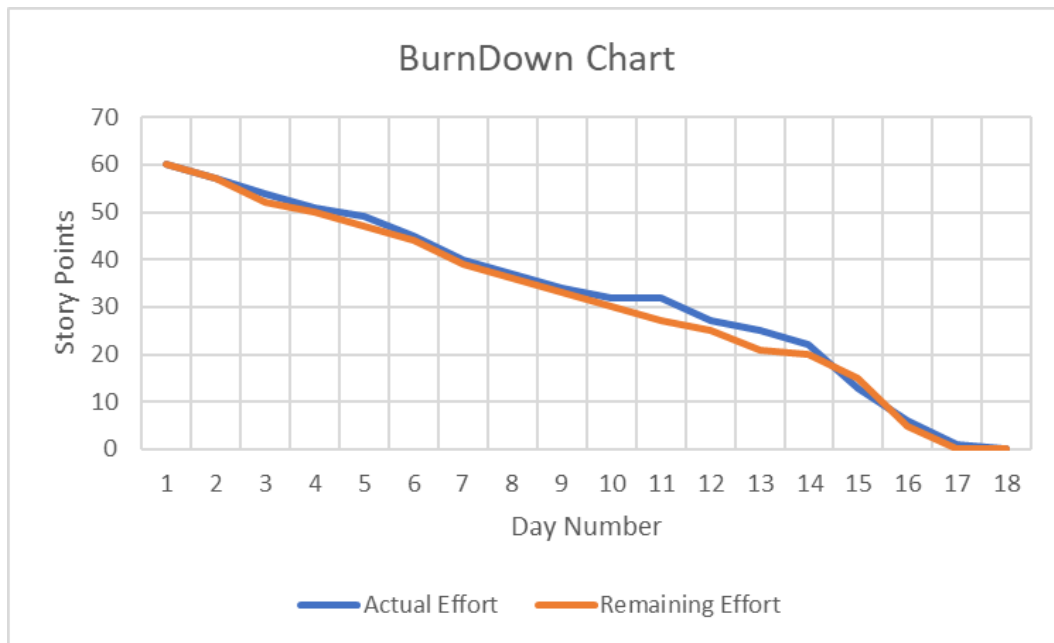
$$= (3.167 + 3 + 4.75)/3$$

$$= 10.917/3$$

$$= 3.639$$

Therefore, Average Velocity = 3.639.

Burndown Chart:



7. CODING AND SOLUTIONING

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To develop an interactive Flask Application that works to recognise ASL Images using a model trained with images from the dataset. This application helps to bridge the communication gap with regards to the American Sign Language.
2.	Idea / Solution description	The solution involves development of a model that is trained with the provided dataset and is able to classify the images across 29 classes belonging to the ASL. This model shall be further implemented into a Flask App, that will provide the users with an option to either input an image or use live webcam to recognise the ASL Signs.
3.	Novelty / Uniqueness	The model is being trained using deep learning methods like VGG-16 and the most accurate one is chosen, helping to improve the accuracy. The application provides ease of use and convenience to the users by virtue of the two input methods.
4.	Social Impact / Customer Satisfaction	The solution provides a reliable means for interpreting the ASL helping the deaf and mute communities to bridge the communication gap between themselves and the people who are not familiar with ASL.

5.	Business Model (Revenue Model)	This solution can be licensed and integrated with other organizations that provide communication services. Can be partnered with other businesses which are in need of our solution.
6.	Scalability of the Solution	This solution can be easily integrated into multiple platforms making it scalable. The model can be further trained using other datasets and models as per requirement. Further cloud services can also be used to facilitate scalability as per demand.

CODING:

1)MODEL TRAINING:

1. LOADING THE DATASET

```
!pip install opendatasets
```

```
Collecting opendatasets
```

```
  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
```

```
Requirement already satisfied: tqdm in
```

```
/usr/local/lib/python3.10/dist-packages (from opendatasets)
```

```
(4.66.1)
```

```
Requirement already satisfied: kaggle in
```

```
/usr/local/lib/python3.10/dist-packages (from opendatasets)
```

```
(1.5.16)
```

```
Requirement already satisfied: click in
```

```
/usr/local/lib/python3.10/dist-packages (from opendatasets)
```

```
(8.1.7)
```

```
Requirement already satisfied: six>=1.10 in
```

```
/usr/local/lib/python3.10/dist-packages (from kaggle-
```

```
>opendatasets) (1.16.0)
```



```
Requirement already satisfied: certifi in
/usr/local/lib/python3.10/dist-packages (from kaggle-
>opendatasets) (2023.11.17)
Requirement already satisfied: python-dateutil in
/usr/local/lib/python3.10/dist-packages (from kaggle-
>opendatasets) (2.8.2)
Requirement already satisfied: requests in
/usr/local/lib/python3.10/dist-packages (from kaggle-
>opendatasets) (2.31.0)
Requirement already satisfied: python-slugify in
/usr/local/lib/python3.10/dist-packages (from kaggle-
>opendatasets) (8.0.1)
Requirement already satisfied: urllib3 in
/usr/local/lib/python3.10/dist-packages (from kaggle-
>opendatasets) (2.0.7)
Requirement already satisfied: bleach in
/usr/local/lib/python3.10/dist-packages (from kaggle-
>opendatasets) (6.1.0)
Requirement already satisfied: webencodings in
/usr/local/lib/python3.10/dist-packages (from bleach->kaggle-
>opendatasets) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.10/dist-packages (from python-slugify-
>kaggle->opendatasets) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle-
>opendatasets) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle-
>opendatasets) (3.6)
Installing collected packages: opendatasets

Successfully installed opendatasets-0.1.22
```

```
import opendatasets as od
data = od.download("https://www.kaggle.com/datasets/grassknoted/asl-
alphabet")
```

Please provide your Kaggle credentials to download this dataset. Learn more: <http://bit.ly/kaggle-creds>

Your Kaggle username: selfstudyzoneindia

Your Kaggle Key:

Downloading asl-alphabet.zip to ./asl-alphabet

100%  1.03G/1.03G [00:11<00:00, 99.4MB/s]

2. DATA PRE-PROCESSING

A. Importing necessary libraries

```
!pip install imutils
import warnings
warnings.filterwarnings("ignore")
import os
import glob
import cv2
import numpy as np
import pandas as pd
import gc
import string
import time
import random
import imutils
from PIL import Image
from tqdm import tqdm
tqdm.pandas()
```

```

import matplotlib
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img,
img_to_array, array_to_img
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten, Dropout
from keras.models import load_model, Model
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint

```

B. Null Value Handling

1. Downloading any sample file from "asl_alphabet_train/del" folder of the training data.
2. Renaming that file as "del_test.jpg"
3. Uploading the file, "del_test.jpg" in the "asl_alphabet_test" folder of the testing data.

C. Labelling the dataset

```

labels = ['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',
'del', 'nothing', 'space']
print(labels)

```

```

['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', 'del',
'nothing', 'space']

```

```
classes = 0
for i in labels:
    classes+=1
print("The number of classes are: ", classes)
```

```
The number of classes are: 29
```

```
train='/content/asl-alphabet/asl_alphabet_train/asl_alphabet_train'
list_path = []
list_labels = []
for label in labels:
    label_path = os.path.join(train, label, "*")
    image_files = glob.glob(label_path)

    sign_label = [label] * len(image_files)

    list_path.extend(image_files)
    list_labels.extend(sign_label)
metadata = pd.DataFrame({"image_path": list_path, "label":
list_labels})
metadata
```

D. Defining a seeding function

```
def seedingFunction(seed: int):
    random.seed(seed)
    os.environ["PYTHONHASHSEED"] = str(seed)
    np.random.seed(seed)
    tf.random.set_seed(seed)
```

3. SPLITTING INTO TRAINING AND TESTING DATA

```
#Testing data
X_train, X_test, Y_train, Y_test =
train_test_split(metadata["image_path"], metadata["label"],
                  test_size=0.15,
                  random_state=2500,
                  shuffle=True,

stratify=metadata["label"]
                  )

data_train = pd.DataFrame({"image_path": X_train,
                           "label": Y_train
                           })

#Validation data
X_train, X_val, Y_train, Y_val =
train_test_split(data_train["image_path"], data_train["label"],
                  test_size=0.15/0.70,
                  random_state=2500,
                  shuffle=True,

stratify=data_train["label"]
                  )

data_train = pd.DataFrame({"image_path": X_train,
                           "label": Y_train
                           })

data_val = pd.DataFrame({"image_path": X_val,
                          "label": Y_val
                          })
```

```
data_test = pd.DataFrame({"image_path": X_test,  
                           "label": Y_test  
                           })
```

```
#TRAINING DATA  
display(data_train)
```

```
len(data_train)
```

```
58103
```

```
#TESTING DATA  
display(data_test)
```

```
len(data_test)
```

```
13050
```

```
#VALIDATION DATASET  
display(data_val)
```

```
len(data_val)
```

```
15847
```

Data Augmentation

```
def data_augmentation():
```

```

datagenerator = ImageDataGenerator(rescale=1/255.,
                                    rotation_range=20,
                                    zoom_range=0.2,
                                    horizontal_flip=True)

#Augmenting Training Images:
train_dataGen = datagenerator.flow_from_dataframe(data_train,
                                                    directory="./",
                                                    x_col="image_path",
                                                    y_col="label",
                                                    class_mode="categorical",
                                                    batch_size=64,
                                                    target_size=(64,
64),
                                                    )

#Augmenting Testing Images:
test_dataGen = datagenerator.flow_from_dataframe(data_test,
                                                  directory="./",
                                                  x_col="image_path",
                                                  y_col="label",
                                                  class_mode="categorical",
                                                  batch_size=1,
                                                  target_size=(64,
64),
                                                  shuffle=False
                                                  )

```

```

    #Augmenting Validation Images:
    validation_dataGen = datagenerator.flow_from_dataframe(data_val,

directory="./",

x_col="image_path",

y_col="label",

class_mode="categorical",

batch_size=64,

target_size=(64, 64),

)

    return train_dataGen, test_dataGen, validation_dataGen

```

```

seedingFunction(2500)
train_dataGen, validation_dataGen, test_dataGen = data_augmentation()

```

```

Found 58103 validated image filenames belonging to 29 classes.

```

```

Found 13050 validated image filenames belonging to 29 classes.

```

```

Found 15847 validated image filenames belonging to 29 classes.

```

```


```

4. MODEL BUILDING

USING VGG-16

```

# Load the VGG16 base model

```



```

model_vgg = VGG16(weights='imagenet', include_top=False,
input_shape=(64, 64, 3))

# Freeze all layers in the base model
for layer in model_vgg.layers:
    layer.trainable = False

# Create your custom model
x = model_vgg.output
x = Flatten()(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x) # Add dropout
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x) # Add dropout
predictions = Dense(29, activation='softmax')(x)

model = Model(inputs=model_vgg.input, outputs=predictions)

# Display the model summary
display(model.summary())

```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5

58889256/58889256 [=====] - 0s 0us/step

Model: "model"

Layer (type)	Output Shape	Param #
--------------	--------------	---------

=====		
=====		

input_1 (InputLayer)	[(None, 64, 64, 3)]	0
block1_conv1 (Conv2D)	(None, 64, 64, 64)	1792
block1_conv2 (Conv2D)	(None, 64, 64, 64)	36928
block1_pool (MaxPooling2D)	(None, 32, 32, 64)	0
block2_conv1 (Conv2D)	(None, 32, 32, 128)	73856
block2_conv2 (Conv2D)	(None, 32, 32, 128)	147584
block2_pool (MaxPooling2D)	(None, 16, 16, 128)	0
block3_conv1 (Conv2D)	(None, 16, 16, 256)	295168
block3_conv2 (Conv2D)	(None, 16, 16, 256)	590080
block3_conv3 (Conv2D)	(None, 16, 16, 256)	590080
block3_pool (MaxPooling2D)	(None, 8, 8, 256)	0
block4_conv1 (Conv2D)	(None, 8, 8, 512)	1180160
block4_conv2 (Conv2D)	(None, 8, 8, 512)	2359808
block4_conv3 (Conv2D)	(None, 8, 8, 512)	2359808
block4_pool (MaxPooling2D)	(None, 4, 4, 512)	0

block5_conv1 (Conv2D)	(None, 4, 4, 512)	2359808
block5_conv2 (Conv2D)	(None, 4, 4, 512)	2359808
block5_conv3 (Conv2D)	(None, 4, 4, 512)	2359808
block5_pool (MaxPooling2D)	(None, 2, 2, 512)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 512)	1049088
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262656
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 29)	14877
=====		
=====		
Total params: 16041309 (61.19 MB)		
Trainable params: 1326621 (5.06 MB)		
Non-trainable params: 14714688 (56.13 MB)		
None		

MODEL COMPILING

```
model.compile(optimizer=Adam(learning_rate=0.001),  
loss='categorical_crossentropy', metrics=['accuracy'])  
  
checkpoint = ModelCheckpoint('asl_vgg16_best_weights.h5',  
save_best_only=True, monitor='val_accuracy', mode='max')
```

MODEL TRAINING

```
history = model.fit(train_dataGen,  
                    steps_per_epoch=train_dataGen.samples // 64,  
                    epochs=15,  
                    validation_data=validation_dataGen,  
                    validation_steps=validation_dataGen.samples // 64,  
                    callbacks=[checkpoint]  
                    )
```

Epoch 1/15

907/907 [=====] - 134s 141ms/step - loss: 1.9712 - accuracy: 0.3929 -
val_loss: 1.1563 - val_accuracy: 0.6749

Epoch 2/15

907/907 [=====] - 124s 136ms/step - loss: 1.2918 - accuracy: 0.5810 -
val_loss: 0.8059 - val_accuracy: 0.7241

Epoch 3/15

907/907 [=====] - 123s 136ms/step - loss: 1.1231 - accuracy: 0.6332 -
val_loss: 0.6289 - val_accuracy: 0.8473

Epoch 4/15


```
907/907 [=====] - 122s 135ms/step - loss: 0.7834 - accuracy: 0.7419 -  
val_loss: 0.4475 - val_accuracy: 0.8867
```

Epoch 13/15

```
907/907 [=====] - 121s 134ms/step - loss: 0.7717 - accuracy: 0.7470 -  
val_loss: 0.4090 - val_accuracy: 0.8571
```

Epoch 14/15

```
907/907 [=====] - 122s 134ms/step - loss: 0.7533 - accuracy: 0.7535 -  
val_loss: 0.4214 - val_accuracy: 0.8818
```

Epoch 15/15

```
907/907 [=====] - 121s 134ms/step - loss: 0.7486 - accuracy: 0.7558 -  
val_loss: 0.3797 - val_accuracy: 0.9163
```

CodeText

5. MODEL EVALUATION

```
scores = model.evaluate(test_dataGen)  
print("%s: %.2f%%" % ("Evaluate Test Accuracy", scores[1]*100))
```

```
248/248 [=====] - 34s 137ms/step - loss:  
0.4429 - accuracy: 0.8686
```

```
Evaluate Test Accuracy: 86.86%
```

■

```
# Plot training accuracy and validation accuracy  
plt.figure(figsize=(12, 4))  
plt.subplot(1, 2, 1)  
plt.plot(history.history['accuracy'], label='Train Accuracy')  
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
```

```

plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Plot training loss and validation loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()

```

6. SAVING THE MODEL

```

fine_tuned_model = load_model("/content/asl_vgg16_best_weights.h5")
fine_tuned_model.save("asl_model.h5")

```

2-A) IMAGE UPLOAD APPLICATION:

i)app.py:

```
import numpy as np
```

```
import os
```

```
import cv2
```

```
import base64
```

```
from tensorflow.keras.models import load_model
```

```
from tensorflow.keras.preprocessing import image
```

```
from flask import Flask, request, render_template, Response, send_from_directory
```

```

app = Flask(__name__)

# Load the model

model = load_model("C:/Users/SRIYA CHINMAYEE/Desktop/ASL/IMAGE UPLOAD
APPLICATION/asl_vgg16_best_weights.h5", compile=False)

@app.route('/')
def index():

    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def upload():

    f = request.files['image']

    print("current path")

    basepath = os.path.dirname(__file__)

    print("current path", basepath)

    filepath = os.path.join(basepath, 'uploads', f.filename)

    print("upload folder is ", filepath)

    f.save(filepath)

    img = image.load_img(filepath, target_size=(64, 64))

    x = image.img_to_array(img)

    print(x)

    x = np.expand_dims(x, axis=0)

    print(x)

    y = model.predict(x)

    preds = np.argmax(y, axis=1)

    index = ['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', 'del', 'nothing', 'space']

    text = "The ASL Sign is recognised as: " + str(index[preds[0]])

```



```

    return text

@app.route('/static/<path:path>')
def send_static(path):
    return send_from_directory('static', path)

@app.route('/asl_vgg16_best_weights.h5')
def send_model():
    return send_from_directory('.', 'asl_vgg16_best_weights.h5')

if __name__ == '__main__':
    app.run(debug=False)

```

ii)index.html:

```

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>ASL Image Recognition</title>

    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">

    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>

    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>

    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>

    <script src="{{ url_for('static', filename='js/main.js') }}"></script>

    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">

</head>

```

```

<body>

  <nav class="navbar navbar-dark bg-info">

    <div class="container text-center">

      <a class="navbar-brand" href="#">ASL Image Recognition</a>

    </div>

  </nav>

  <div class="container text-center">

    <div id="content" style="margin-top:2em">

      <div class="container text-center">

        <div class="row">

          <div class="col-sm-6 bd">

            <h3>ASL Image Recognition: </h3>

            <p>The American Sign Language (ASL) is the primary language used by deaf
            individuals in North America. It is a visual language that uses a combination of hand gestures,
            facial expressions, and body movements to convey meaning. In recent years, there has been an
            increasing interest in developing technologies to help bridge the communication gap between the
            deaf and hearing communities.</p>

            </div>

          <div class="col-sm-6">

            <div class="btn-section">

              <h4>Upload an Image:</h4>

              <div>

                <button type="button" class="btn btn-info btn-lg" id="btn-upload">Upload
                Image</button>

```

```
        </div>
    </div>
    <form action="{{ url_for('upload') }}" id="upload-file" method="post"
enctype="multipart/form-data" style="display:none;">
        <br>
        <label for="imageUpload" class="upload-label">
            Choose...
        </label>
        <input type="file" name="image" id="imageUpload" accept=".png, .jpg,
.jpeg">
    </form>
    <div class="image-section component-container" style="display:none;">
        <center>
            <div class="img-preview">
                <div id="imagePreview"></div>
            </div>
        </center>
        <div>
            <button type="button" class="btn btn-info btn-lg" id="btn-
predict">Predict</button>
        </div>
    </div>
    <div class="result-section" style="display:none;">
        <h3 id="result">Result:</h3>
    </div>
</div>
</div>
</div>
</div>
</div>
```

</body>

</html>

iii)main.js:

```
$(document).ready(function () {  
    console.log("Document ready!");  
  
    // Image preview and upload  
    function readURL(input) {  
        if (input.files && input.files[0]) {  
            var reader = new FileReader();  
            reader.onload = function (e) {  
                $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');  
                $('#imagePreview').hide();  
                $('#imagePreview').fadeIn(650);  
            };  
            reader.readAsDataURL(input.files[0]);  
        }  
    }  
  
    // Trigger file input on button click  
    $("#btn-upload").click(function () {  
        console.log("Upload button clicked!");  
        $("#imageUpload").click();  
    });  
  
    // Handle file input change  
    $("#imageUpload").change(function () {  
        console.log("File input changed!");  
    });  
});
```

```

        $('.image-section').show();
        $('#btn-predict').show();
        $('.result-section').hide();
        readURL(this);
    });

    // Predict button click
    $("#btn-predict").click(function () {
        console.log("Predict button clicked!");
        var form_data = new FormData($('#upload-file')[0]);
        $.ajax({
            type: 'POST',
            url: '/predict',
            data: form_data,
            contentType: false,
            cache: false,
            processData: false,
            async: true,
            success: function (data) {
                console.log('Success!');
                console.log(data);
                $('.result-section').show();
                $('#result').html(data);
            },
        });
    });
});

```

iv)main.css:

```
.bg-dark {
  background-color: #87CEFA!important;
}

#result {
  color: #0a1c4ed1;
}

body {
  background-image: url("https://wallpaperaccess.com/full/1799643.jpg");
  background-size: cover;
}

.btn-section {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.btn-section button {
  margin-top: 1em;
}

.component-container {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.result-section {
  margin-top: 2em;
}
```

```

}

.img-preview {
    width: 256px;
    height: 256px;
    position: relative;
    border: 5px solid #FFF0F5;
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
    margin-top: 1em;
    margin-bottom: 1em;
}

.img-preview>div {
    width: 100%;
    height: 100%;
    background-size: 256px 256px;
    background-repeat: no-repeat;
    background-position: center;
}

#live-result {
    color: #0a1c4ed1;
}

```

2-B)WEBCAM LIVE STREAM APPLICATION:

```

i)app.py:
# app.py
from flask import Flask, render_template, request
import cv2

```

```

import numpy as np

from keras.preprocessing.image import img_to_array

from keras.models import load_model

from flask_socketio import SocketIO

import base64

import os

app = Flask(__name__)

socketio = SocketIO(app)

UPLOAD_FOLDER = 'uploads'

ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

alphabet = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',
'X', 'Y', 'Z', 'del', 'nothing', 'space']

model = load_model("C:/Users/SRIYA CHINMAYEE/Desktop/ASL/WEBCAM LIVE STREAM
APPLICATION/asl_vgg16_best_weights.h5")

def allowed_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

def classify(image):

    if image.shape[2] == 3:

        # The image is already in BGR format

        img = image

    else:

        # Convert the grayscale image to BGR

        img = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)

    img = cv2.resize(img, (64, 64))

```



```

img = img.astype("float") / 255.0
img = img_to_array(img)
img = np.expand_dims(img, axis=0)
proba = model.predict(img)
idx = np.argmax(proba)
confidence_score = proba[0][idx] * 100
if 0 <= idx < len(alphabet):
    return alphabet[idx], confidence_score
else:
    return "Invalid index", confidence_score # Handle the case where the index is out of range

@app.route('/')
def index():
    return render_template('index.html')

@socketio.on('video_frame')
def handle_video_frame(frame):
    nparr = np.fromstring(base64.b64decode(frame.split(',')[1]), np.uint8)
    img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
    alpha, confidence = classify(img)
    socketio.emit('classification_result', {'alpha': alpha, 'confidence': confidence})

if __name__ == '__main__':
    if not os.path.exists(UPLOAD_FOLDER):
        os.makedirs(UPLOAD_FOLDER)

    socketio.run(app, debug=True, use_reloader=False, allow_unsafe_werkzeug=True) # Disable
the Flask reloader

```

ii)index.html:

```
<!-- templates/index.html -->

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>ASL Recognition</title>

  <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">

  <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>

  <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>

  <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>

  <script src="https://cdn.jsdelivr.net/npm/socket.io/4.1.3/socket.io.js"></script>

  <script src="{{ url_for('static', filename='main.js') }}"></script>

  <link rel="stylesheet" href="{{ url_for('static', filename='main.css') }}">

</head>

<body style="background-image: url('https://wallpaperaccess.com/full/1799643.jpg');
background-size: cover;">

  <nav class="navbar navbar-dark bg-info">

    <div class="container text-center">

      <a class="navbar-brand" href="#">ASL Image Recognition</a>

    </div>

  </nav>

  <div class="container text-center mt-5">
```

```
<div class="row">
```

```
<div class="col-sm-6 bd">
```

```
<h3>ASL Image Recognition: </h3>
```

```
<p>The American Sign Language (ASL) is the primary language used by deaf  
individuals in
```

```
North America. It is a visual language that uses a combination of hand gestures,  
facial
```

```
expressions, and body movements to convey meaning. In recent years, there has been  
an
```

```
increasing interest in developing technologies to help bridge the communication gap  
between the deaf and hearing communities.</p>
```

```

```

```
</div>
```

```
<div class="col-sm-6">
```

```
<h2>Live ASL Recognition:</h2>
```

```
<video id="video" autoplay style="width: 100%; height: 75%"></video>
```

```
<h3><p id="result">Result: <span id="classification"></span></p></h3>
```

```
<script>
```

```
var socket = io.connect('http://' + document.domain + ':' + location.port);
```

```
navigator.mediaDevices.getUserMedia({ video: true })
```

```
.then(function (stream) {
```

```
var video = document.getElementById('video');
```

```
video.srcObject = stream;
```

```
video.addEventListener('loadeddata', function () {
```

```
video.style.display = 'block';
```

```

        setInterval(function () {
            sendFrame();
        }, 1000 / 10);
    });
})

.catch(function (error) {
    console.log("Error accessing webcam: ", error);
});

function sendFrame() {
    var canvas = document.createElement('canvas');
    var context = canvas.getContext('2d');
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    context.drawImage(video, 0, 0, canvas.width, canvas.height);
    var frameData = canvas.toDataURL('image/jpeg');
    socket.emit('video_frame', frameData);
}

socket.on('classification_result', function (data) {
    document.getElementById('classification').innerText = data.alpha + ' (Confidence: ' + data.confidence.toFixed(2) + '%)';
});
</script>
</div>
</div>
</div>
</body>

</html>

```

iii)main.js:

```
// main.js

document.addEventListener('DOMContentLoaded', function () {

    // Add event listener to the file input element
    var fileInput = document.querySelector('input[type="file"]');
    fileInput.addEventListener('change', handleFileSelect);

    // Set up SocketIO connection
    var socket = io.connect('http://' + document.domain + ':' + location.port);

    // Receive video frames from the server
    socket.on('video_frame', function (frame) {
        var img = document.getElementById('video');
        img.src = 'data:image/jpeg;base64,' + frame;
    });

    // Request the video feed when the page is loaded
    socket.emit('video_feed');
});

function handleFileSelect(event) {
    // Get the selected file
    var file = event.target.files[0];

    // Display the selected image preview (optional)
    var preview = document.getElementById('image-preview');
    preview.src = URL.createObjectURL(file);
    preview.style.display = 'block';
}
```

```
var socket = io.connect('http://' + document.domain + ':' + location.port);
```

```
document.getElementById('start-recognition').addEventListener('click', function () {  
    // Turn on the webcam when the button is clicked  
    startWebcam();  
});
```

```
function startWebcam() {  
    navigator.mediaDevices.getUserMedia({ video: true })  
        .then(function (stream) {  
            var video = document.getElementById('video');  
            video.srcObject = stream;  
  
            var canvas = document.getElementById('canvas');  
            var context = canvas.getContext('2d');  
  
            video.addEventListener('loadeddata', function () {  
                video.style.display = 'block';  
                canvas.style.display = 'block';  
  
                setInterval(function () {  
                    context.drawImage(video, 0, 0, 640, 480);  
                    sendFrame();  
                }, 1000 / 10);  
            });  
        })  
        .catch(function (error) {  
            console.log("Error accessing webcam: ", error);  
            resultText.innerText = "Error accessing webcam.";
```

```

    });
}

function sendFrame() {
    var resultText = document.getElementById('result');
    var frameData = canvas.toDataURL('image/jpeg');
    socket.emit('video_frame', frameData);
}

socket.on('classification_result', function (data) {
    document.getElementById('classification').innerText = data.alpha + ' (Confidence: ' +
    data.confidence.toFixed(2) + '%)';
});

```

iv)main.css:

```

/* main.css */

.bg-dark {
    background-color: #87CEFA!important;
}

#result {
    color: #0a1c4ed1;
}

body {
    background-image: url("https://wallpaperaccess.com/full/1799643.jpg");
    background-size: cover;
}

.btn-section {
    display: flex;

```

```
    flex-direction: column;
    align-items: center;
}
```

```
.btn-section button {
    margin-top: 1em;
}
```

```
.component-container {
    display: flex;
    flex-direction: column;
    align-items: center;
}
```

```
.result-section {
    margin-top: 2em;
}
```

```
h1, h2, h3 {
    color: #fff;
}
```

```
p {
    color: #fff;
}
```

```
.navbar {
    background-color: #17a2b8;
}
```

```
.btn-info {
```

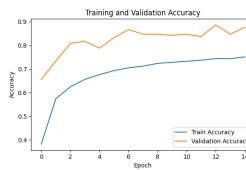
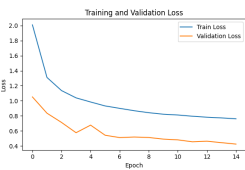


```
        margin-top: 1em;
    }
    div {
        margin-top: 20px;
    }
    /* Update the video styling in main.css */
    video {
        display: block; /* Ensure the video is treated as a block element */
        margin: 0 auto; /* Center the video within its container */
    }
```

8. PERFORMANCE TESTING

8.1 PERFORMANCE METRICS

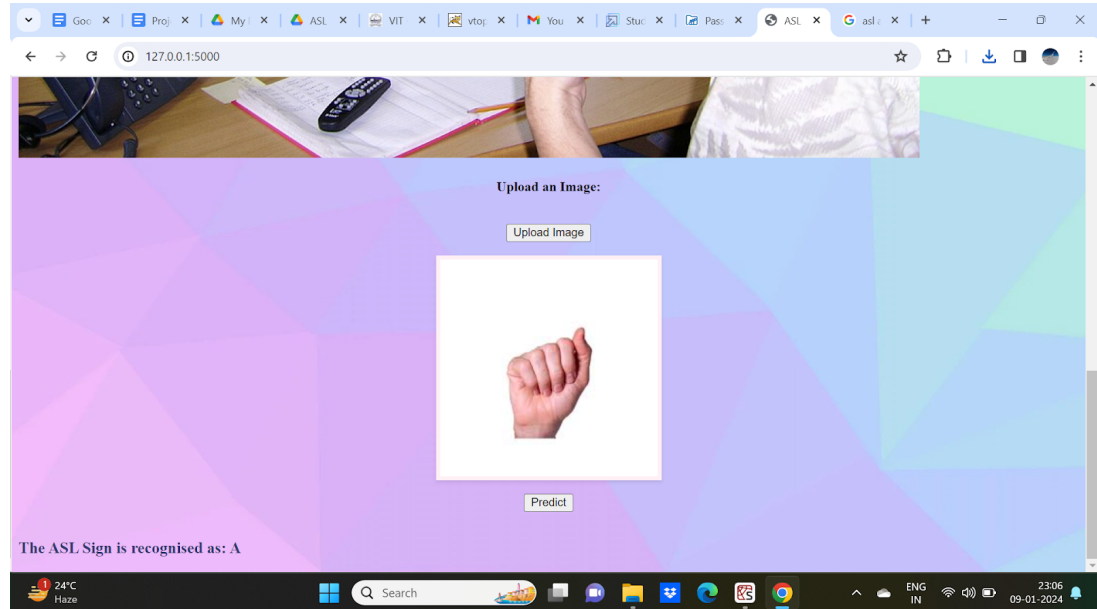
S.No.	Parameter	Values	Screenshot
-------	-----------	--------	------------

1.	Model Summary	<p>There is 1 Input Layer</p> <p>There are 13 Convolutional Layers</p> <p>There are 5 MaxPooling Layers</p> <p>ReLU and SoftMax activation functions have been used</p>	<pre> ===== input_1 (InputLayer) [(None, 64, 64, 3)] 0 block1_conv1 (Conv2D) (None, 64, 64, 64) 1792 block1_conv2 (Conv2D) (None, 64, 64, 64) 36928 block1_pool (MaxPooling2D) (None, 32, 32, 64) 0 block2_conv1 (Conv2D) (None, 32, 32, 128) 73856 block2_conv2 (Conv2D) (None, 32, 32, 128) 147584 block2_pool (MaxPooling2D) (None, 16, 16, 128) 0 block3_conv1 (Conv2D) (None, 16, 16, 256) 295168 block3_conv2 (Conv2D) (None, 16, 16, 256) 590080 block3_conv3 (Conv2D) (None, 16, 16, 256) 590080 block3_pool (MaxPooling2D) (None, 8, 8, 256) 0 block4_conv1 (Conv2D) (None, 8, 8, 512) 1180160 block4_conv2 (Conv2D) (None, 8, 8, 512) 2359008 block4_conv3 (Conv2D) (None, 8, 8, 512) 2359008 block4_pool (MaxPooling2D) (None, 4, 4, 512) 0 block5_conv1 (Conv2D) (None, 4, 4, 512) 2359008 block5_conv2 (Conv2D) (None, 4, 4, 512) 2359008 block5_conv3 (Conv2D) (None, 4, 4, 512) 2359008 block5_pool (MaxPooling2D) (None, 2, 2, 512) 0 flatten (Flatten) (None, 2048) 0 dense (Dense) (None, 512) 1049088 dropout (Dropout) (None, 512) 0 dense_1 (Dense) (None, 512) 262656 dropout_1 (Dropout) (None, 512) 0 dense_2 (Dense) (None, 29) 14877 ===== Total params: 16041309 (61.19 MB) Trainable params: 1326621 (5.06 MB) Non-trainable params: 14714688 (56.13 MB) None </pre>
2.	Accuracy	<p>Training Accuracy - 86.86%</p> <p>Validation Accuracy - 87.68%</p>	<pre> Epoch 1/15 [=====] - 136s 140ms/step - loss: 2.0090 - accuracy: 0.3021 - val_loss: 1.0514 - val_accuracy: 0.6552 Epoch 2/15 [=====] - 125s 138ms/step - loss: 1.1106 - accuracy: 0.5341 - val_loss: 0.8354 - val_accuracy: 0.7340 Epoch 3/15 [=====] - 126s 138ms/step - loss: 1.1161 - accuracy: 0.5240 - val_loss: 0.7128 - val_accuracy: 0.8079 Epoch 4/15 [=====] - 123s 138ms/step - loss: 1.0193 - accuracy: 0.6563 - val_loss: 0.5761 - val_accuracy: 0.8177 Epoch 5/15 [=====] - 123s 138ms/step - loss: 0.9084 - accuracy: 0.6762 - val_loss: 0.6269 - val_accuracy: 0.7882 Epoch 6/15 [=====] - 125s 138ms/step - loss: 0.9319 - accuracy: 0.6931 - val_loss: 0.5425 - val_accuracy: 0.8325 Epoch 7/15 [=====] - 125s 138ms/step - loss: 0.8999 - accuracy: 0.7050 - val_loss: 0.5111 - val_accuracy: 0.8670 Epoch 8/15 [=====] - 124s 137ms/step - loss: 0.8688 - accuracy: 0.7123 - val_loss: 0.5180 - val_accuracy: 0.8673 Epoch 9/15 [=====] - 125s 137ms/step - loss: 0.8421 - accuracy: 0.7217 - val_loss: 0.5126 - val_accuracy: 0.8673 Epoch 10/15 [=====] - 125s 137ms/step - loss: 0.8220 - accuracy: 0.7284 - val_loss: 0.4912 - val_accuracy: 0.8624 Epoch 11/15 [=====] - 125s 138ms/step - loss: 0.8115 - accuracy: 0.7328 - val_loss: 0.4812 - val_accuracy: 0.8673 Epoch 12/15 [=====] - 123s 138ms/step - loss: 0.7952 - accuracy: 0.7372 - val_loss: 0.4553 - val_accuracy: 0.8374 Epoch 13/15 [=====] - 124s 138ms/step - loss: 0.7821 - accuracy: 0.7444 - val_loss: 0.4633 - val_accuracy: 0.8807 Epoch 14/15 [=====] - 124s 137ms/step - loss: 0.7728 - accuracy: 0.7430 - val_loss: 0.4630 - val_accuracy: 0.8673 Epoch 15/15 [=====] - 124s 138ms/step - loss: 0.7638 - accuracy: 0.7513 - val_loss: 0.4258 - val_accuracy: 0.8768 248/248 [=====] - 40s 161ms/step - loss: 0.4500 - accuracy: 0.8670 Evaluate Test Accuracy: 86.70% </pre> <div>   </div>

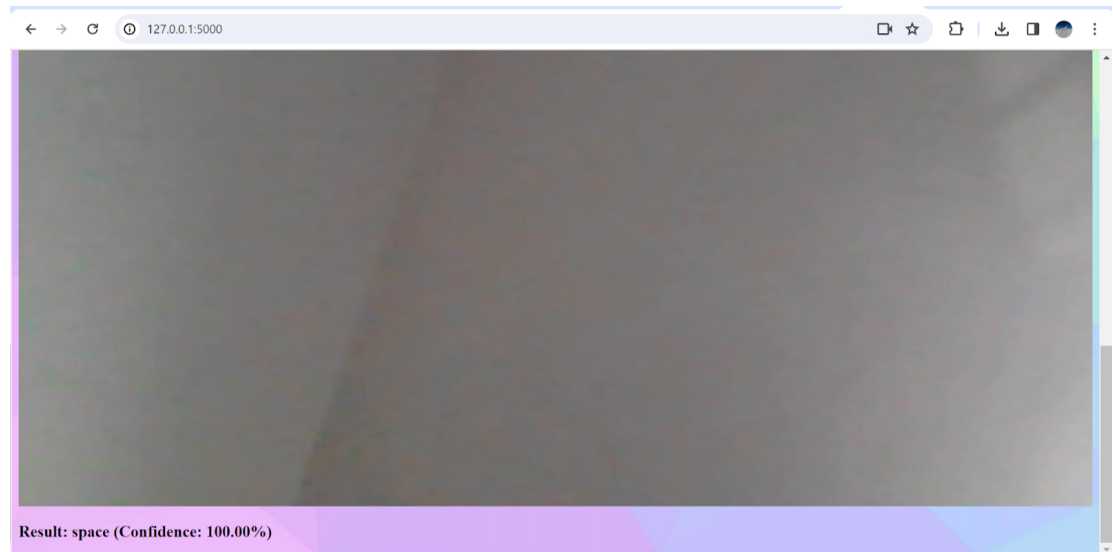
9. RESULTS

9.1 OUTPUT SCREENSHOTS

1. IMAGE UPLOAD APPLICATION:



2. LIVE WEBSTREAM APPLICATION:



10. ADVANTAGES AND DISADVANTAGES

The advantages of ASL-Alphabet Image Recognition project are:

- Bridges the communication gap between the deaf who use ASL and the people who don't

- It fosters inclusivity for ASL users, improving their job opportunities and social relations
- Can be used across various types of browsers
- Can be used for educational purposes
- Promotes the use of technology to tackle problems

The disadvantages of ASL- Alphabet Image Recognition project are:

- Training the model on a limited dataset
- Low Accuracy
- Data bias

11. CONCLUSION

Finally, the ASL Alphabet Image Recognition project is on the cutting edge of using technology to address communication gaps between the deaf and hearing communities. The project is a pioneering step towards encouraging inclusivity and understanding by focusing on the recognition of ASL alphabet signs. Its importance stems from the fact that it provides a tool that not only reads hand signs but also has the ability to revolutionize how sign language is integrated into everyday communication.

Looking ahead, the project's future scope promises to have a significant impact.

The project shows the ability of technology to effect positive change, paving the way for sign language to be integrated into the fabric of everyday communication.

12. FUTURE SCOPE

In the future we can improve the ASL-Alphabet Image Recognition project by further improving the dataset quality by introducing new signs and gestures and making the number of images per sign equal to avoid data bias. We can also expand our project to other sign languages like the Indian Sign Language(ISL) etc. We can incorporate this web application into mobile applications improving its accessibility. We can also incorporate it with the fields of education, wearable technology, assistive devices etc.

13. APPENDIX

GitHub link: <https://github.com/smartinternz02/SI-GuidedProject-703519-1704728043/upload>

Project Demo Link: <https://drive.google.com/drive/folders/1-1O5qBjivuArM7bgQ73K6vY1VBu8PAf9?usp=sharing>