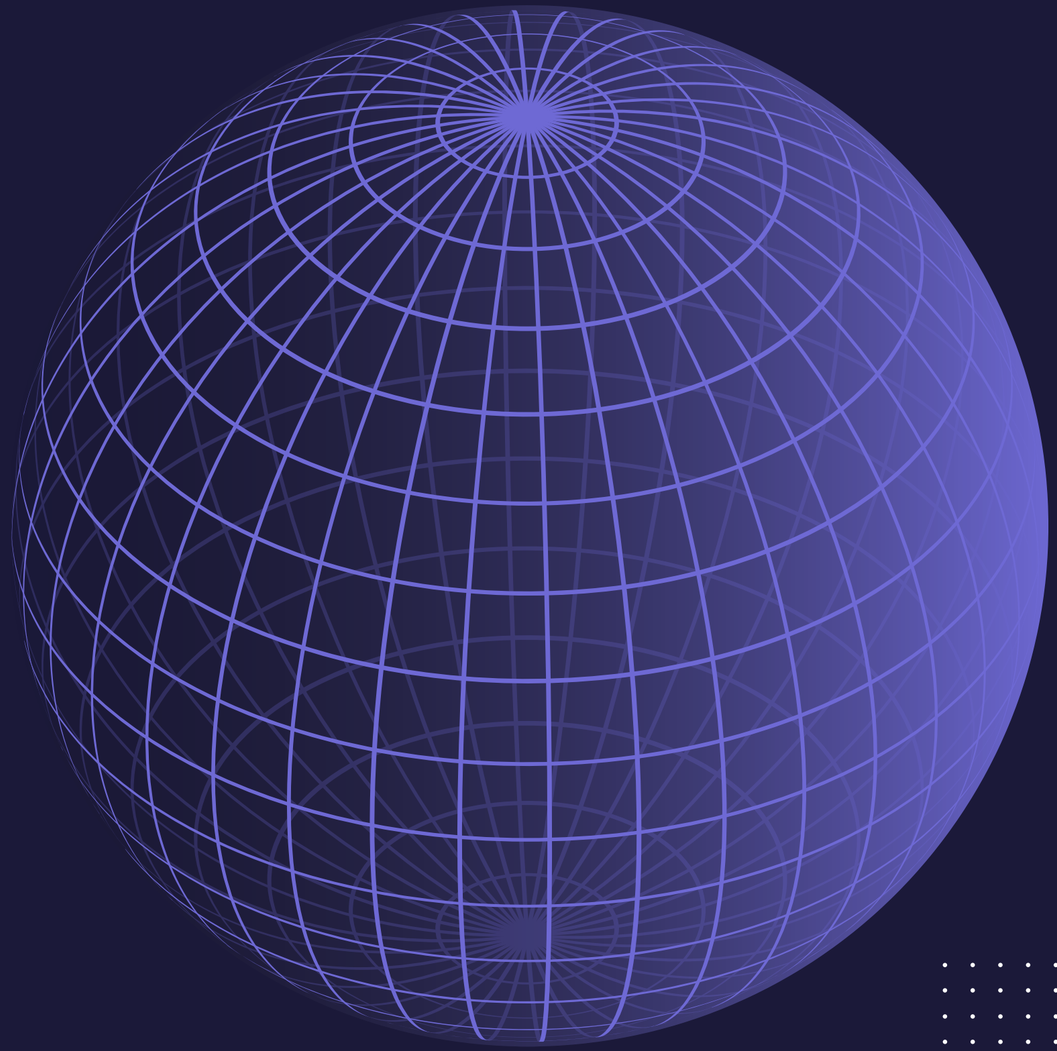
HAND GESTURE DETECTION

SIGN LANGUAGE DETECTION



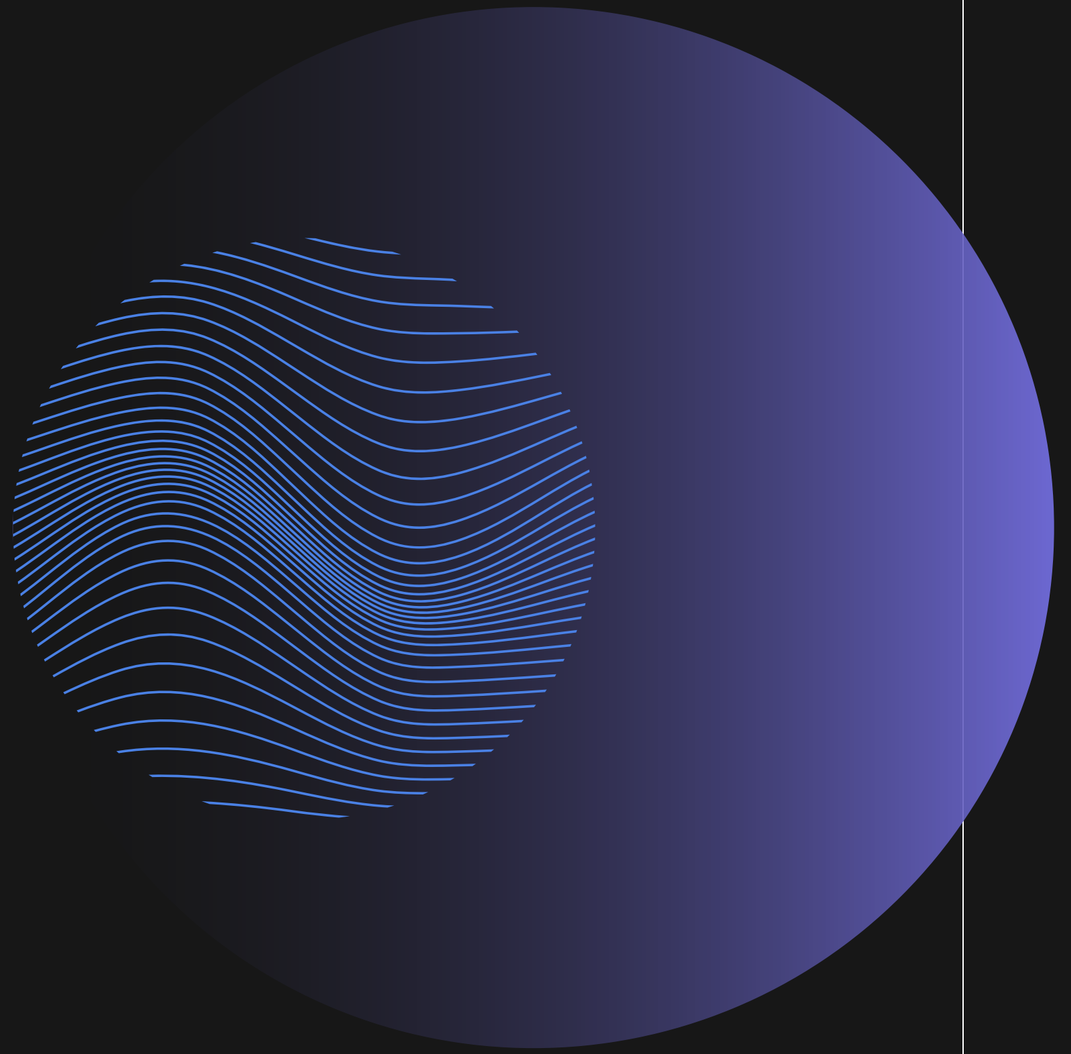
001

SIGN LANGUAGE DETECTION

GROUP - 7

GROUP MEMBERS

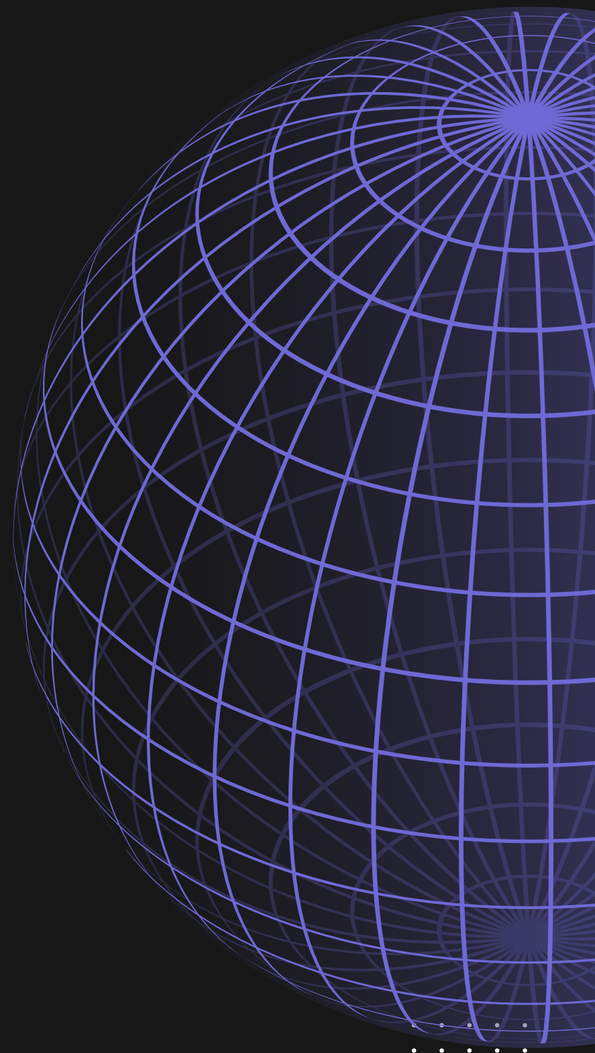
JILL SHAH - AU2040234
KHUSHEE VAKIL - AU2040242
DHRUVI SHAH - AU2040263



INTRODUCTION

- The project aims to develop a hand sign language identification system using ResNet architecture to predict hand signs from loaded images after training on a dataset of 40500 hand sign images.

- The primary objective is to accurately recognize hand gestures and translate them into text or speech.



PROBLEM STATEMENT

Brief Introduction

- The goal of this project is to develop a robust and accurate computer vision system that can accurately recognize and classify hand signs from loaded images.

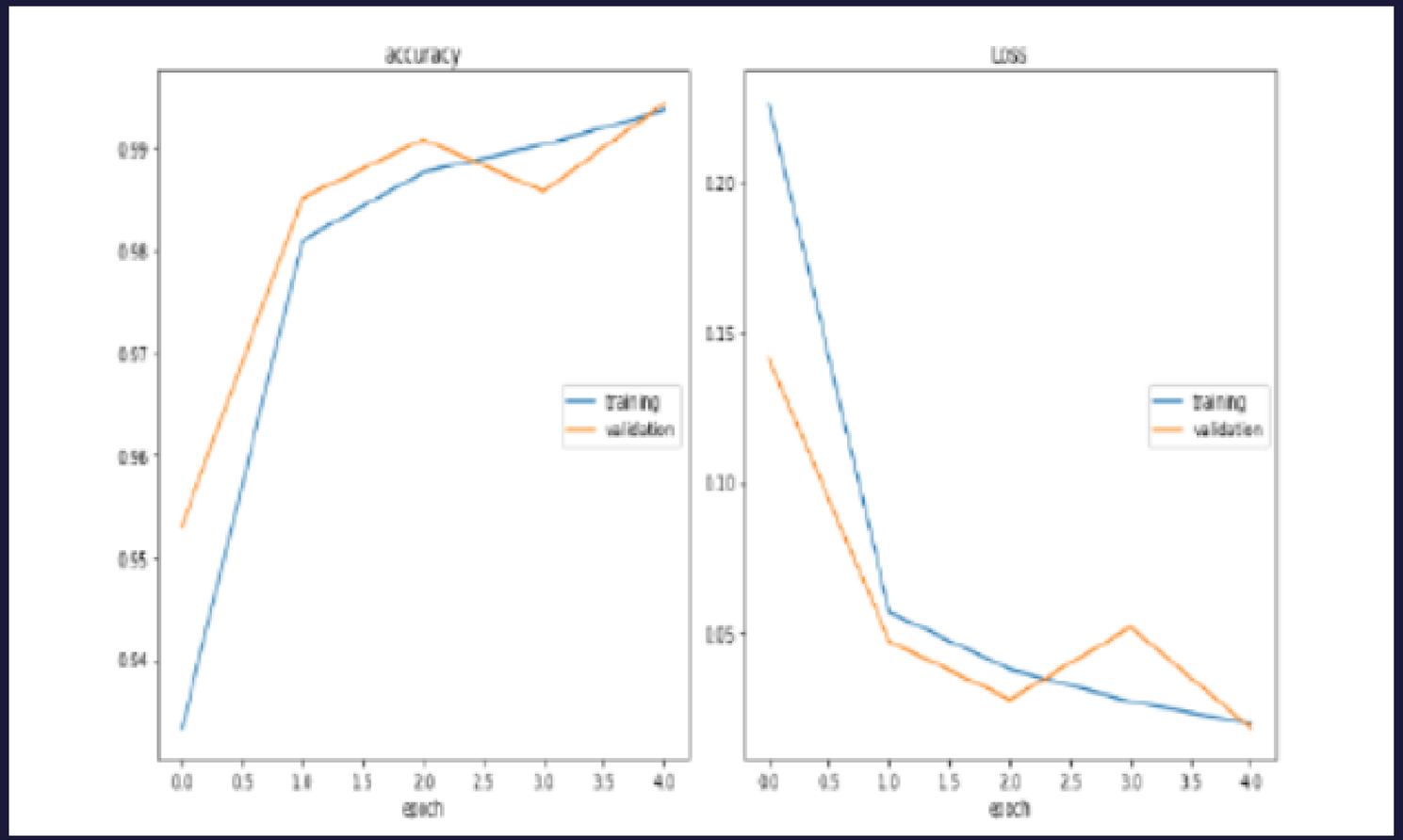
- Create a flexible system for recognising sign language motions for a range of hand gestures, postures, illumination, and backdrops.

EXISTING BODY OF WORK

The "Sign Language Recognition" research paper proposes a two-step approach using deep learning for recognizing sign language gestures:

1. Detecting and tracking hand and fingers.
2. Using a trained CNN to classify hand and finger regions into different sign language gestures.

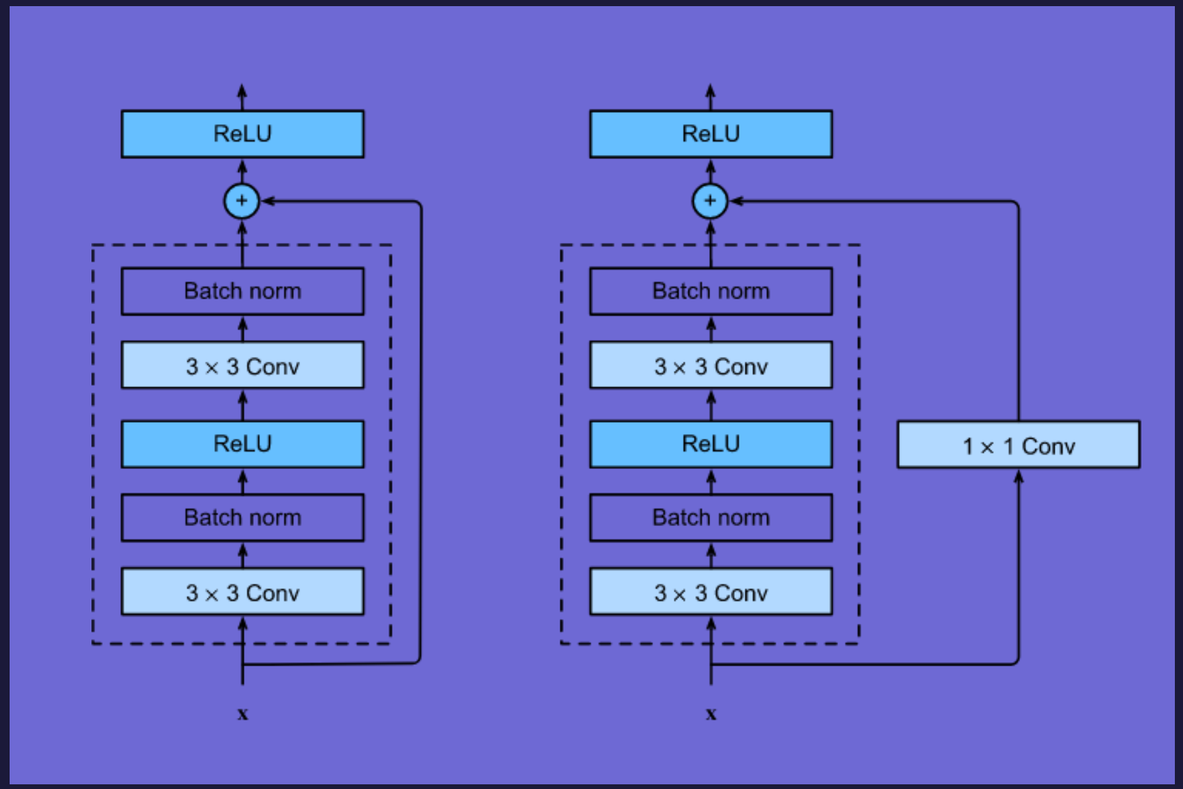
Advanced models (MobileNetV2, LeNet-5) achieved promising 99.61% accuracy on test data.



EXISTING BODY OF WORK

007

WHAT IS RESNET



- ResNet is a popular deep learning architecture used for tasks like image classification. It has superior feature extraction capabilities due to residual connections.
- These connections prevent the vanishing gradient problem that can occur in deep neural networks with many layers.
- The architecture consists of stacked convolutional layers, followed by a pooling layer and a fully connected layer.

OUR APPROACH

1. **Collect a diverse dataset of hand gesture images for sign language recognition.**
2. **Preprocess the dataset by resizing, normalizing colors, and removing backgrounds.**
3. **Train a CNN-based deep learning model on the dataset.**
4. **Deploy the trained model and test its performance in various lighting conditions and backgrounds.**
5. **Make adjustments to the model or hardware/software setup to improve performance as needed.**

- The following code snippets are about our dataset. Our dataset contains 27 classes with 1500 images in each. We have used the validation split of 0.2 which divided our dataset in a ratio of 80:20 of training to validation set.

```
[78]
#import train data
train_data = tf.keras.preprocessing.image_dataset_from_directory(
    "C:/AU/AU third year/6th sem/cv/archive/Sign Language for Alphabets", labels='inferred', label_mode='int', class_names=None,
    color_mode='rgb', batch_size=32, image_size=(50, 50), shuffle=True, seed=123,
    validation_split=0.2, subset="training"
)
```

Python

```
Found 40500 files belonging to 27 classes.
Using 32400 files for training.
```

- There are 27 classes from a to z and one more class of unknown.

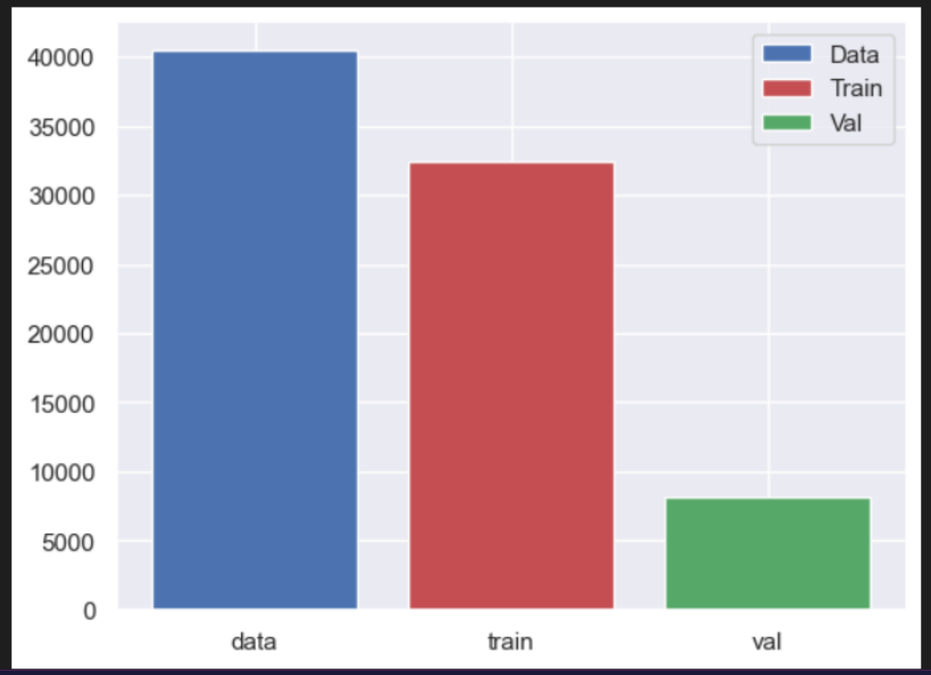
```
[80]
import os
labels = os.listdir("C:/AU/AU third year/6th sem/cv/archive/Sign Language for Alphabets")
labels.sort()
print(labels)
```

Python

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'unknown', 'v', 'w', 'x', 'y', 'z']
```

```
#visulaize data
fig, ax = plt.subplots()
ax.bar("data",40500 ,color= 'b', label='Data')
ax.bar("train",32400 ,color= 'r', label='Train')
ax.bar("val",8100 ,color='g', label='Val')
leg = ax.legend();
```

[81]



- The following plotted graph shows the size of the dataset, training statset and the validation dataset after the validation split of 0.2

```python
from tensorflow.keras.layers import Input, BatchNormalization, Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Add, ReLU
from tensorflow.keras.models import Model

# Define input shape
input_shape = (50, 50, 3)

# Define input tensor
inputs = Input(shape=input_shape)

# Initial convolution layer
x = Conv2D(64, kernel_size=(3,3), padding='same')(inputs)
x = BatchNormalization()(x)
x = ReLU()(x)

# Residual blocks
for _ in range(3):
    shortcut = x

    x = Conv2D(64, kernel_size=(3,3), padding='same')(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)

    x = Conv2D(64, kernel_size=(3,3), padding='same')(x)
    x = BatchNormalization()(x)

    # Add skip connection
    x = Add()([shortcut, x])
    x = ReLU()(x)

    x = MaxPooling2D(pool_size=(2,2))(x)

# Flatten and dense layers
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = Dense(27, activation='softmax')(x)

# Define the model
model = Model(inputs=inputs, outputs=outputs)
model.summary()
```

[74]

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_6 (InputLayer)         [(None, 50, 50, 3)]       0         []

conv2d_35 (Conv2D)           (None, 50, 50, 64)        1792      ['input_6[0][0]']

batch_normalization_35 (BatchN  (None, 50, 50, 64)     256       ['conv2d_35[0][0]']
ormalization)

re_lu_35 (ReLU)              (None, 50, 50, 64)        0         ['batch_normalization_35[0][0]']

conv2d_36 (Conv2D)           (None, 50, 50, 64)        36928     ['re_lu_35[0][0]']

batch_normalization_36 (BatchN  (None, 50, 50, 64)     256       ['conv2d_36[0][0]']
ormalization)

re_lu_36 (ReLU)              (None, 50, 50, 64)        0         ['batch_normalization_36[0][0]']

conv2d_37 (Conv2D)           (None, 50, 50, 64)        36928     ['re_lu_36[0][0]']

batch_normalization_37 (BatchN  (None, 50, 50, 64)     256       ['conv2d_37[0][0]']
ormalization)

...
Total params: 523,675
Trainable params: 522,779
Non-trainable params: 896
```

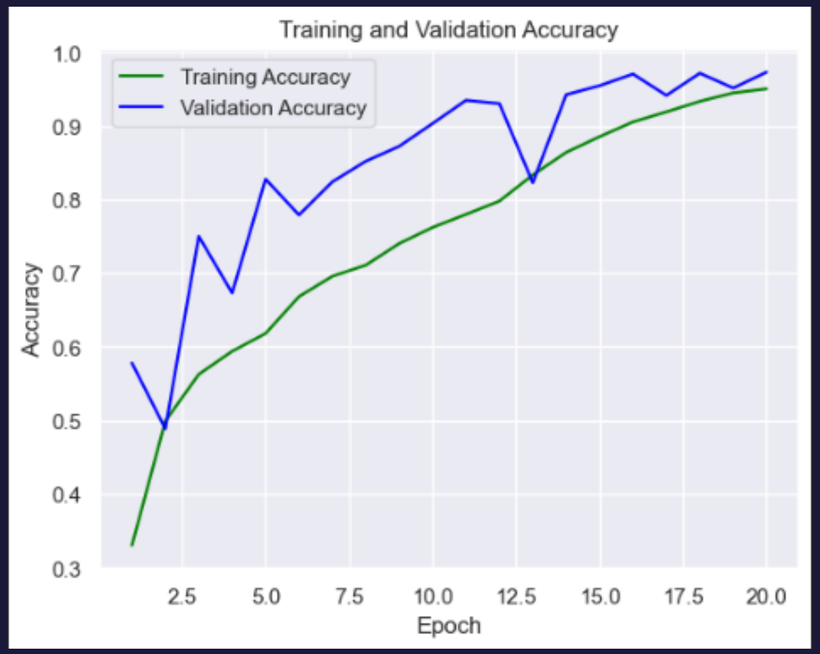
- The code here showcases the resnet architecture that has been deployed using the residual blocks. The results of this code give us the total number of parameters and the trainable parameters

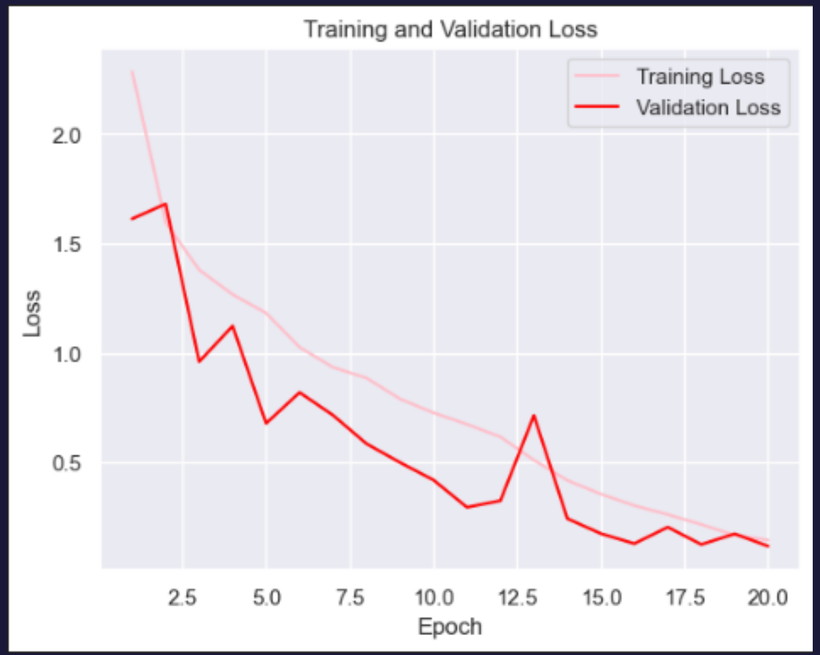
RESULTS

- Our model has an adaptive learning rate and sparse categorical crossentropy loss function.
- On evaluating our data, our model gives us an accuracy of 97.28%.

```
[30]  model.compile(optimizer='Adam', metrics=['accuracy'], loss='sparse_categorical_crossentropy')

[31]  call = tf.keras.callbacks.EarlyStopping(
          monitor="val_loss",
          patience=5,
          restore_best_weights=True)

[32]  fit= model.fit(train_data,validation_data=val_data,epochs=20,callbacks=[call])
```

```
[33]  model.evaluate(val_data)

...   254/254 [==============================] - 25s 99ms/step - loss: 0.1182 - accuracy: 0.9728

      [0.11816398054361343, 0.9728395342826843]
```

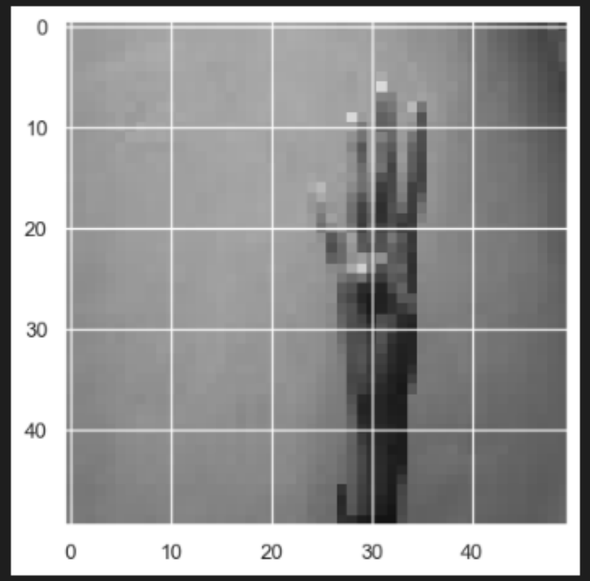




- Our model is able to predict the sign language gestures in around 25 ms and with different lighting backgrounds which shows the robustness of our model.

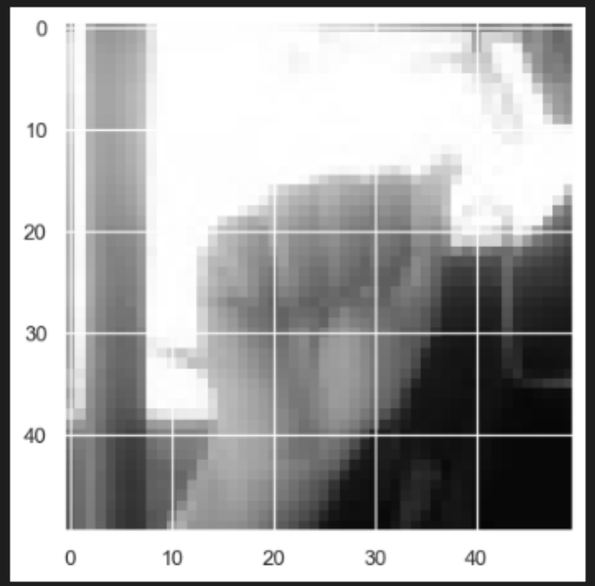
```
image_path = "C:/AU/AU third year/6th sem/cv/archive/Sign Language for Alphabets/b/b_10.jpg"
new_img = image.load_img(image_path, target_size=(50, 50))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
prediction = model.predict(img)
prediction = np.argmax(prediction,axis=1)
print(prediction)
print(dict_labels[prediction[0]])
plt.imshow(new_img)
```

```
1/1 [==============================] - 0s 24ms/step
[1]
b

<matplotlib.image.AxesImage at 0x1c4d14ce560>
```




```
image_path = "C:/AU/AU third year/6th sem/cv/archive/Sign Language for Alphabets/a/a_10.jpg"
new_img = image.load_img(image_path, target_size=(50, 50))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
prediction = model.predict(img)
prediction = np.argmax(prediction,axis=1)
print(prediction)
print(dict_labels[prediction[0]])
plt.imshow(new_img)
```

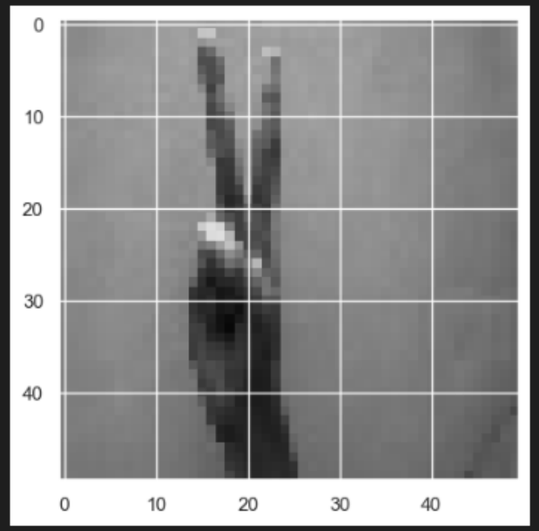
```
1/1 [==============================] - 0s 26ms/step
[0]
a

<matplotlib.image.AxesImage at 0x1c4d131ace0>
```



```
image_path = "C:/AU/AU third year/6th sem/cv/archive/Sign Language for Alphabets/v/v_10.jpg"
new_img = image.load_img(image_path, target_size=(50, 50))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
prediction = model.predict(img)
prediction = np.argmax(prediction,axis=1)
print(prediction)
print(dict_labels[prediction[0]])
plt.imshow(new_img)
```

[60]

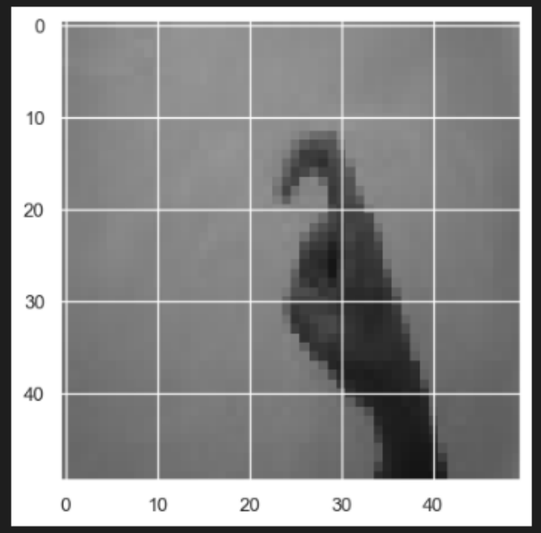
```
1/1 [==============================] - 0s 22ms/step
[22]
v

<matplotlib.image.AxesImage at 0x1c4d4d7e830>
```




```
image_path = "C:/AU/AU third year/6th sem/cv/archive/Sign Language for Alphabets/x/x_10.jpg"
new_img = image.load_img(image_path, target_size=(50, 50))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
prediction = model.predict(img)
prediction = np.argmax(prediction,axis=1)
print(prediction)
print(dict_labels[prediction[0]])
plt.imshow(new_img)
```

[76]

```
1/1 [==============================] - 0s 379ms/step
[24]
x

<matplotlib.image.AxesImage at 0x1c4d937a440>
```



CONCLUSION

- Our project uses ResNet architecture for computer vision to detect ASL alphabet signs.
- Our ResNet-based hand sign language detection model achieves 97% accuracy on the test dataset, indicating its high performance in correctly identifying hand signs from uploaded photos.

ROLE OF EACH GROUP MEMBER

JILL SHAH - Model compilation and resnet architecture.
KHUSHEE VAKIL - collecting dataset and classifying training and validation dataset.
DHRUVI SHAH - Data cleaning and resizing the dataset.

Training and testing of data and weekly reports are collectively done by all.

REFERENCES

1. Akash. (2018, April 22). *Asl alphabet*. Kaggle. Retrieved March 11, 2023, from https://www.kaggle.com/datasets/grassknoted/asl-alphabet
2. Kale, H. V., & Sonawane, R. S. (2021). Sign Language Recognition. Retrieved from ResearchGate: https://www.researchgate.net/publication/354066737_Sign_Language_Recognition
3. *Hand sign detection (ASL)*. Computer Vision Zone. (2022, July 4). Retrieved March 11, 2023, from https://www.computervision.zone/courses/hand-sign-detection-asl/

THANK YOU