Project Title: American Sign Language (ASL) Detection Using Machine Learning
Name: Muhammed Saneen Ak
Internship: Machine Learning Internship
Institution: Unified Mentor Pvt. Ltd
Date: 15/06/2025

Acknowledgment

I would like to express my sincere gratitude to my mentors and team for guiding me through this project. This work helped me understand how AI can assist in accessibility, especially in bridging communication gaps for the hearing-impaired.

Abstract

This project aims to detect American Sign Language hand gestures using a machine learning model trained on image data. The goal is to recognize signs corresponding to alphabets and enable real-time communication assistance. A convolutional neural network (CNN) was used to achieve high recognition accuracy from static hand gesture images.

Table of Contents

1. Introduction

Sign language is a vital means of communication for people with hearing and speech impairments. This project introduces an AI-based solution to detect and translate American Sign Language (ASL) gestures, helping bridge the gap between sign language users and the general population.

2. Problem Statement

There is a lack of real-time, low-cost tools to translate ASL into text or speech, limiting accessibility. This project addresses this problem using a machine learning-based classification approach.

3. Objective

- To classify hand gestures corresponding to ASL alphabets.
- To develop an image classification model using CNN.
- To achieve high accuracy on validation data and prepare for real-time use.

4. Tools and Technologies Used

- **Language**: Python
- **Libraries**: TensorFlow, Keras, OpenCV, NumPy, Matplotlib
- **Platform**: Jupyter Notebook
- **Model Type**: CNN
- **Hardware**: CPU/GPU (Laptop)

5. Dataset Description

- The dataset contains labelled images of ASL alphabet hand gestures (A-Z and del, space, nothing).
- Each class has 1000+ images with various backgrounds and lighting.
- Preprocessing included grayscale conversion, resizing, and normalization.
- Split the data in to train and test as X,y

```
img_path = glob.glob('asl_alphabet_train/*/*.jpg')
image = []
label = []
for path in img_path:
    lbs = os.path.basename(os.path.dirname(path))
    img = Image.open(path).convert('RGB')
    img = img.resize((64,64))
    image.append(np.array(img))
    label.append(lbs)
label_encoder = LabelEncoder()
encoded = label_encoder.fit_transform(label)
```

```
X = np.array(image).astype('float32') / 255.0
y = np.array(encoded)
```

6. Methodology

- Data preprocessing and augmentation
- CNN model development
- Training and tuning using callbacks
- Validation using accuracy and confusion matrix

7. Implementation

- Used a CNN architecture with convolutional, pooling, and dense layers.
- Batch normalization and dropout were applied to reduce overfitting.
- Model trained with categorical cross-entropy loss and Adam optimizer.

```
from keras import Sequential,layers
from tensorflow.keras import regularizers
```

```
model = Sequential()
model.add(layers.Conv2D(128,(3,3),activation='relu',input_shape = (64,64,3)))
model.add(layers.MaxPool2D(2,2))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(0.3))

model.add(layers.Conv2D(64,(3,3),activation='relu',kernel_regularizer = regularizers.l2(0.01)))
model.add(layers.MaxPool2D(2,2))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(0.5))

model.add(layers.Flatten())
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dropout(0.6))
model.add(layers.Dense(30,activation='softmax'))
```

8. Challenges Faced

- Variability in hand sizes and lighting conditions
- Overfitting on small training sets
- Confusion between similar-looking signs (e.g., M, N, T)

```
model.compile(optimizer='adam',loss = 'sparse_categorical_crossentropy',metrics=['accuracy'])
model.fit(xtrain,ytrain,epochs=20,validation_split=0.2)
```

```
Epoch 1/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 87s 144ms/step - accuracy: 0.1241 - loss: 3.7009 - val_accuracy: 0.1157 - val_loss: 9.7804
Epoch 2/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 79s 137ms/step - accuracy: 0.2080 - loss: 2.6855 - val_accuracy: 0.3403 - val_loss: 1.9964
Epoch 3/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 79s 136ms/step - accuracy: 0.2414 - loss: 2.4447 - val_accuracy: 0.1981 - val_loss: 6.5382
Epoch 4/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 79s 137ms/step - accuracy: 0.2756 - loss: 2.3049 - val_accuracy: 0.3869 - val_loss: 1.7445
Epoch 5/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 80s 137ms/step - accuracy: 0.3018 - loss: 2.2087 - val_accuracy: 0.4972 - val_loss: 1.6053
```

After :-

```
Epoch 15/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 80s 139ms/step - accuracy: 0.5724 - loss: 1.3188 - val_accuracy: 0.6754 - val_loss: 1.0104
Epoch 16/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 80s 138ms/step - accuracy: 0.6081 - loss: 1.2380 - val_accuracy: 0.6935 - val_loss: 1.0782
Epoch 17/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 80s 138ms/step - accuracy: 0.6408 - loss: 1.1398 - val_accuracy: 0.6231 - val_loss: 1.4998
Epoch 18/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 80s 138ms/step - accuracy: 0.6601 - loss: 1.0764 - val_accuracy: 0.7957 - val_loss: 0.7124
Epoch 19/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 81s 139ms/step - accuracy: 0.6771 - loss: 1.0371 - val_accuracy: 0.8560 - val_loss: 0.6211
Epoch 20/20
580/580 ━━━━━━━━━━━━━━━━━━━━ 81s 139ms/step - accuracy: 0.6866 - loss: 0.9866 - val_accuracy: 0.8793 - val_loss: 0.4196
```

9. Results and Evaluation

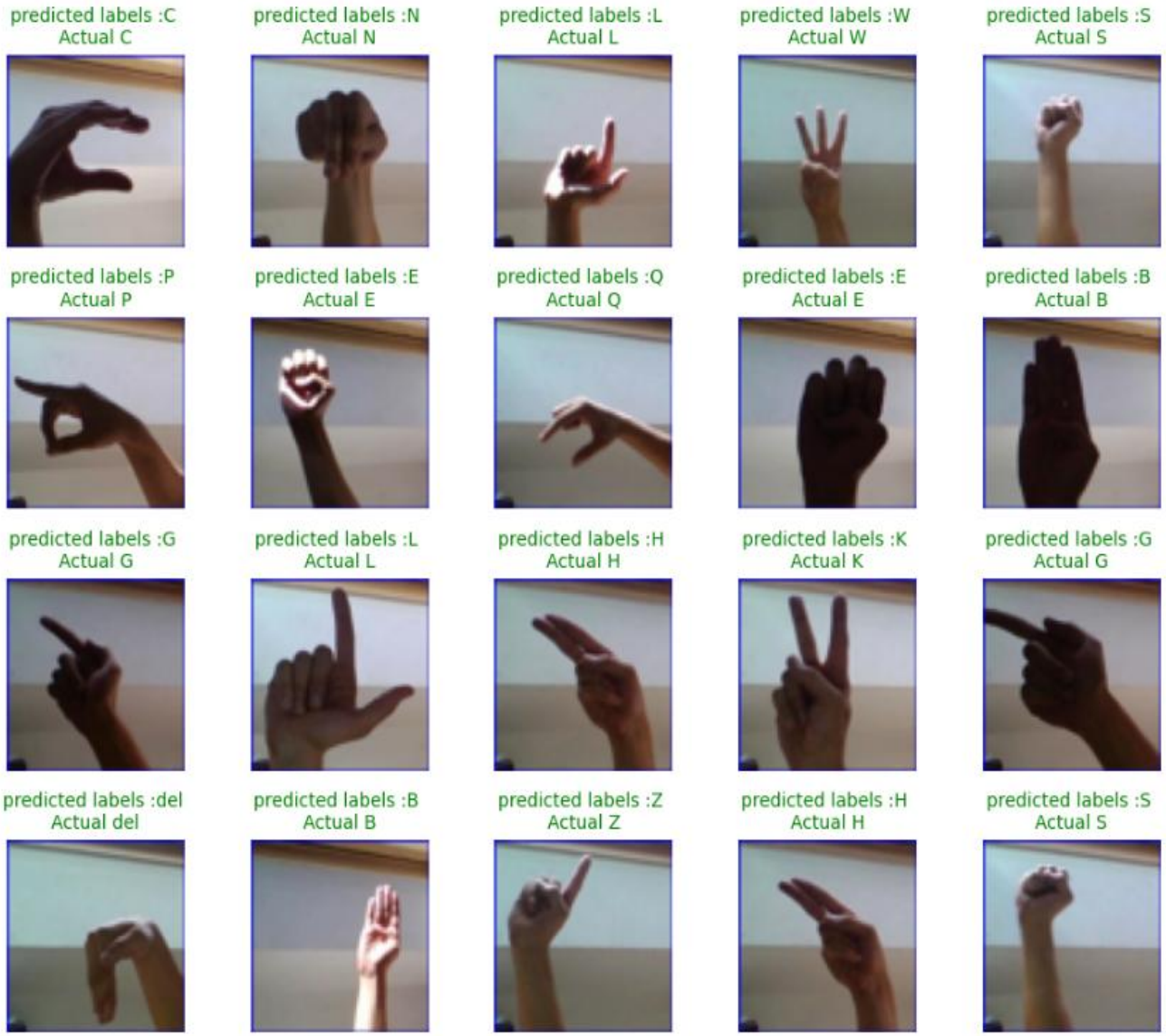
- **Training Accuracy**: ~76%
- **Testing Accuracy**: ~87%
- **Confusion Matrix**: High accuracy for most signs, with minor confusion in a few
- Real-time test cases performed well under controlled lighting

Visualize Actual and Predicted Label

Plot Here :-

Code :

```
plt.figure(figsize=(10,8))
for i in range(20):
    color = 'green' if decoded[i] == actual[i] else 'red'
    plt.subplot(4,5,i+1)
    plt.imshow(xtest[i])
    plt.title(f'predicted labels :{decoded[i]} \nActual {actual[i]}', color = color, fontsize = 10)
    plt.tight_layout()
    plt.axis('off')
```



10. Project Scope and Future Work

- Extend to real-time ASL-to-text translation using webcam input (OpenCV)
- Support dynamic gestures (J, Z) using video frames or RNN/LSTM
- A future extension of this model includes real-time ASL detection using OpenCV via webcam input, allowing gesture-to-text conversion for accessibility tools.

11. Conclusion

The project successfully built a robust classifier for ASL hand gestures using deep learning. This has practical implications for accessibility tools and human-computer interaction in the future.

12. References

- TensorFlow/Keras Documentation
- ASL Dataset (Kaggle / OpenASL)
- Research papers on gesture recognition
- OpenCV Documentation