NITHIN S
221IT085

IT204 MINI PROJECT

Visual Hand Gesture Recognition with Convolutional Neural Network (CNN)

Designing Proposed Method

Tech Stack :

- Linux Environment
- Python Programming Language
- CNN Architecture
- Open CV
- Tensorflow
- Keras
- Numpy

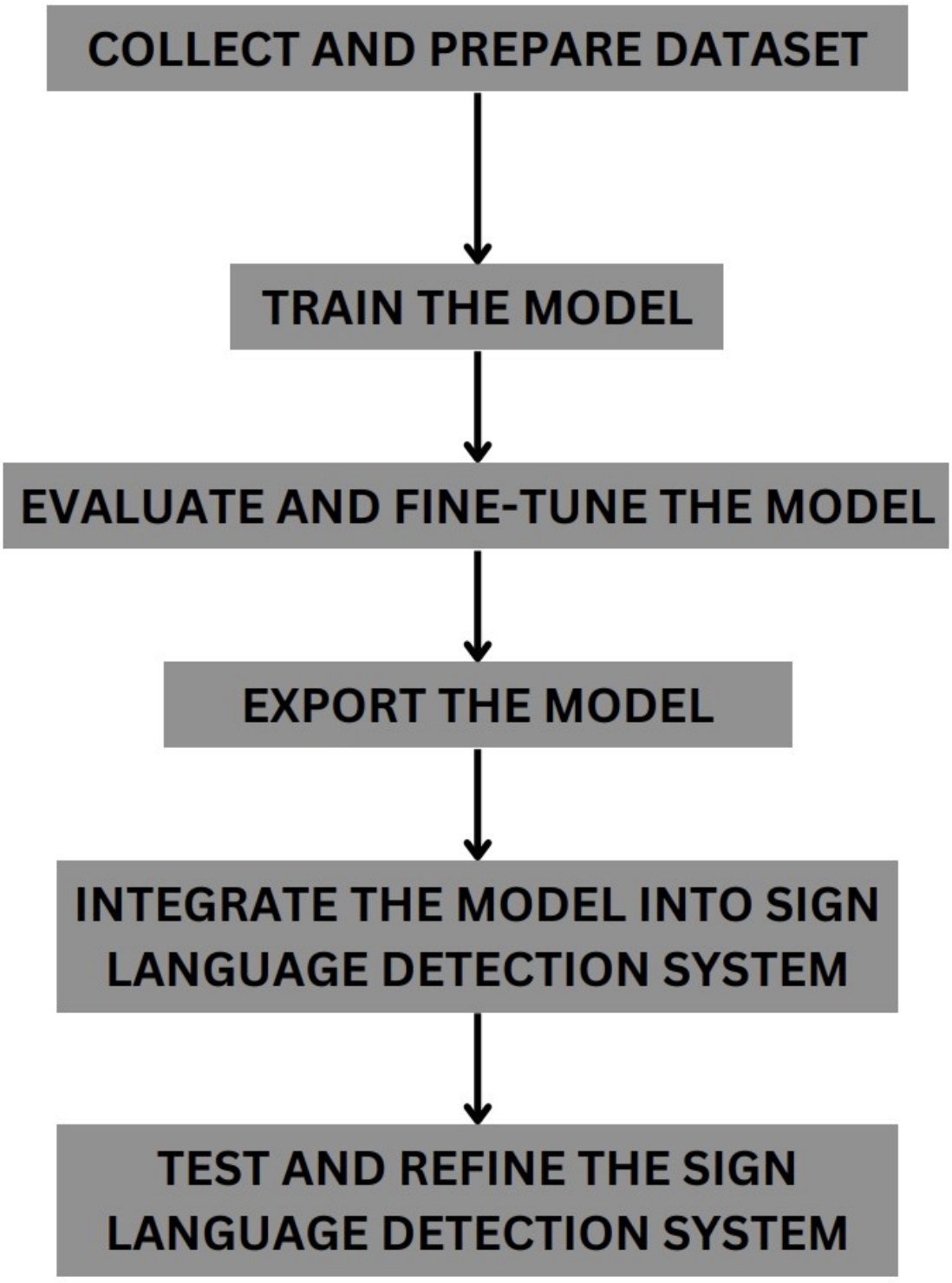
Research Paper Referred:

S. Meshram, R. Singh, P. Pal and S. K. Singh, "Convolution Neural Network based Hand Gesture Recognition System," 2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 2023, pp. 1-5, doi: 10.1109/ICAECT57570.2023.10118267.

Proposed Method:

- We have created a robust dataset of hand gestures for all alphabets having 3000 images for each letter.
- All these images will be convoluted using CNN architecture which will eventually train a keras .h5 model.
- This trained ML model will recognize hand gestures of the **American Sign Langauge** in real time through the webcam.

Simple Block Diagram of our proposed method



Description of the block diagram

Step 1: Collect and Prepare Dataset:

Gather a dataset of sign language gestures, which includes images of different sign gestures performed by various sign language users. Label the dataset, assigning appropriate class labels to each sign gesture to indicate the corresponding sign being performed. Split the dataset into training, validation, and testing sets for model training and evaluation.

Step 3: Train the Model:

Use the train.py to train your sign language detection model using the uploaded dataset. Experiment with different model architectures and hyperparameters to optimize model performance. Monitor the training process and iterate as needed to achieve satisfactory model accuracy.

Step 4: Evaluate and Fine-tune the Model:

Evaluate the trained model using the validation and testing sets to assess its accuracy and generalization performance. Fine-tune the model by adjusting model parameters or collecting more data, if necessary, to improve model accuracy.

Step 5: Export the Model:

Once satisfied with the model's performance, export the trained model from Teachable Machine in a format suitable for your target platform (e.g., TensorFlow, TensorFlow.js, or a custom format). Follow the instructions provided by Teachable Machine to export and download the model files.

Step 6: Integrate the Model into the Sign Language Detection System:

Incorporate the exported model into your sign language detection system, using the appropriate libraries or frameworks for your target platform (e.g.,

TensorFlow.js for web-based applications or TensorFlow for Python-based applications). Implement the necessary preprocessing steps, such as hand detection and tracking, in your system to prepare input data for the model. Utilize the exported model to perform real-time sign language gesture recognition on the captured video or image data in your system.

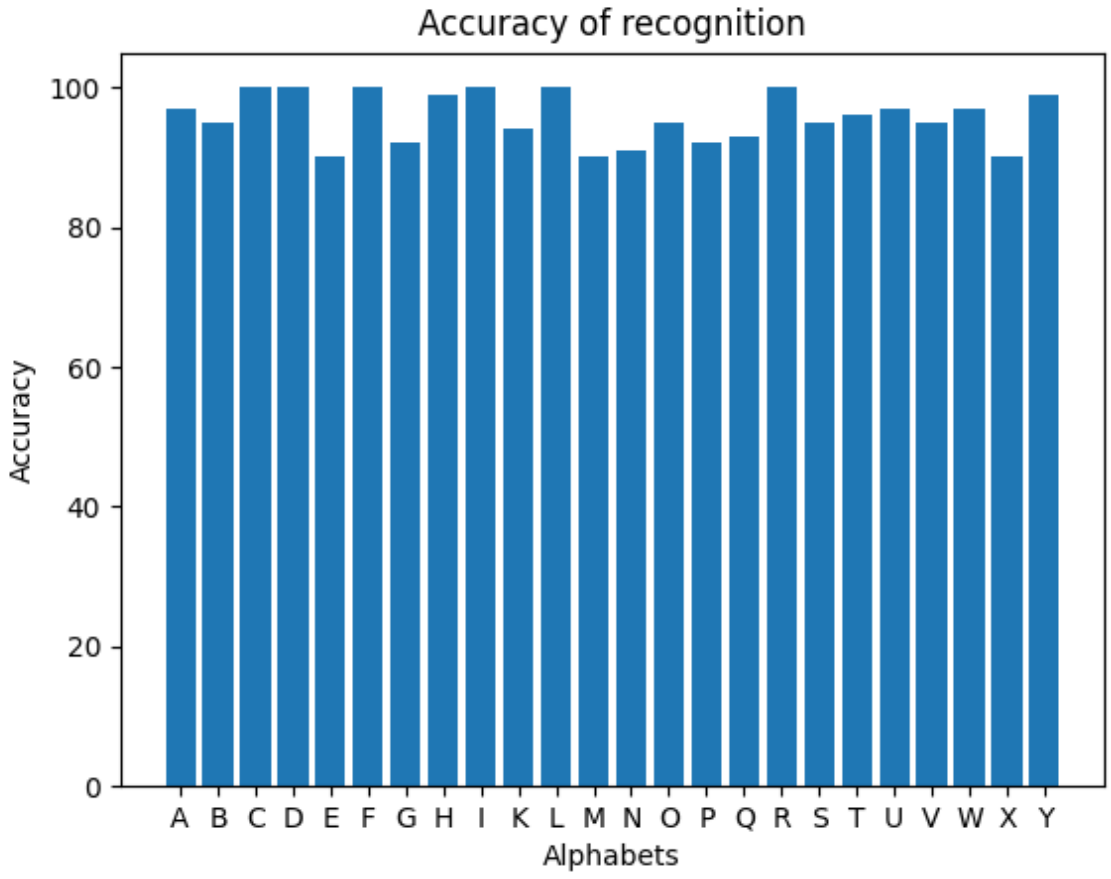
Step 7: Test and Refine the Sign Language Detection System:

Test the sign language detection system with real-world sign language gestures to assess itsaccuracy and performance.Refine the system as needed based on user feedback and evaluation results, such as improving hand detection, tracking, or gesture recognition accuracy.

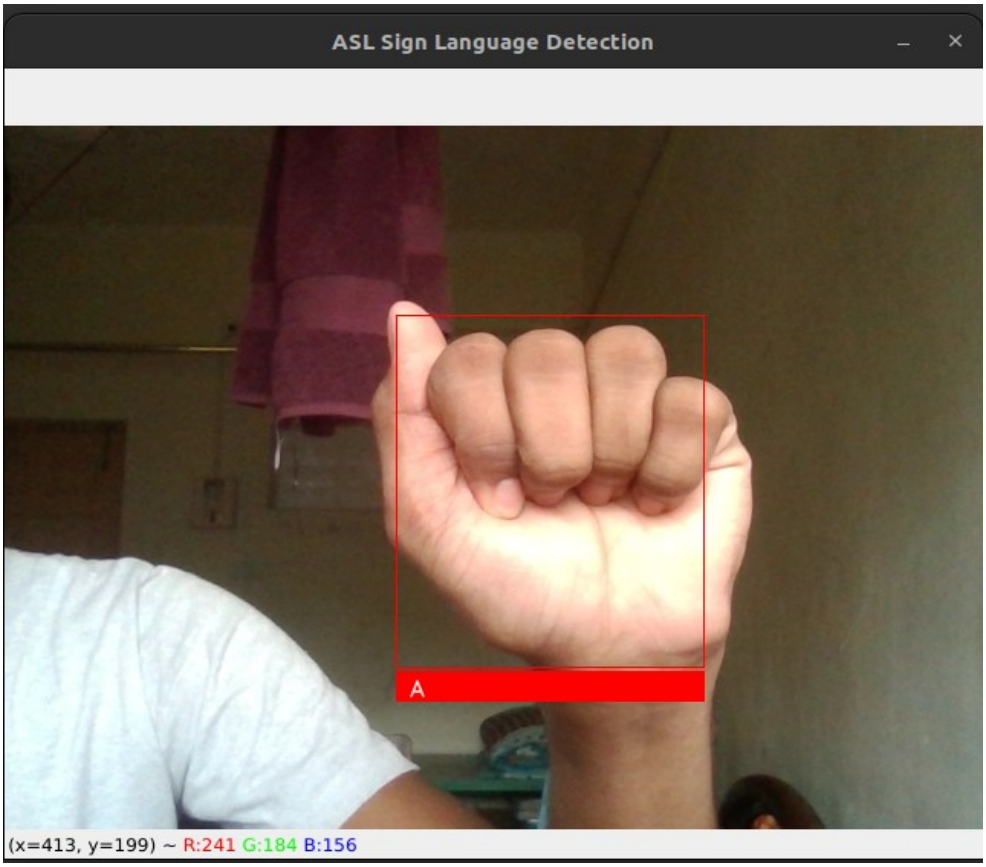
Insights:

The methodology presented above boasts an impressive accuracy rate of 96%. In a rigorous trial involving 100 signs, the ML Model demonstrated its prowess by accurately recognizing 96 of them. This exceptional performance can be attributed to the robust dataset comprising over 78,000 images, with a substantial 3,000 images dedicated to each alphabet.

Dataset


Accuracy of recognition
Accuracy
100
80
60
40
20
0
A B C D E F G H I K L M N O P Q R S T U V W X Y
Alphabets

Results


ASL Sign Language Detection
A
(x=413, y=199) ~ R:241 G:184 B:156


ASL Sign Language Detection
R
(x=413, y=199) ~ R:51 G:31 B:11


ASL Sign Language Detection
K
(x=413, y=199) ~ R:53 G:33 B:17


ASL Sign Language Detection
W
(x=413, y=199) ~ R:64 G:38 B:17