Hand Gestures Recognition (ASL) With ESP32-CAM

Ahmed-Marwan Mohamed-Yasser Shehab-Essam Mostafa-Karam - Wael-Ahmed-EL Zuhairy

Under Supervision Prof Amgad Bayoumy

Abstract:

Communication plays a vital role in human interaction, and American Sign Language (ASL) is a critical medium for the deaf and hard-of-hearing communities. This project aims to develop a robust deep learning-based system for recognizing ASL alphabets using Convolutional Neural Networks (CNNs). The system processes hand gesture images, converting them into corresponding alphabetical characters to improve accessibility for individuals with speech or hearing impairments.

The system design integrates advanced data preprocessing techniques, including image rescaling and one-hot encoding, alongside CNN architecture optimized with dropout and data augmentation for enhanced accuracy. A TensorFlow Lite model is deployed on an ESP-32 CAM module, enabling real-time recognition and offline processing on edge devices. The pipeline includes image capture, preprocessing, gesture classification, and output display via an LCD.

Results demonstrate the effectiveness of the proposed system, achieving notable accuracy in gesture recognition and highlighting its potential for improving communication accessibility. Future work includes expanding the dataset to accommodate dynamic signs and exploring further optimizations.

1.INTRODUCTION

Communication is a cornerstone of human interaction, and language serves as a critical tool for mutual understanding. For the deaf and hard-of-hearing communities, American Sign Language (ASL) offers a vital means of communication. However, the lack of accessibility to ASL interpreters and the complexity of manual translation pose significant challenges. This research focuses on developing a deep learning-based system to bridge this gap by recognizing ASL alphabet gestures from images. By leveraging advancements in artificial intelligence, this system aims to enhance communication capabilities for individuals with speech or hearing impairments.

2.SYSTEM DESIGN

2.1 Dataset and Preprocessing

The system's effectiveness relies on robust data preprocessing and organization, which include the following steps:

Dataset Loading: Images are stored in folders labeled with corresponding ASL letters, such as A, B, and C. A metadata DataFrame maps each image to its respective label for streamlined processing.

Data Splitting: To ensure reliable evaluation, the dataset is divided into training (70%), validation (15%), and testing (15%) subsets. Stratified sampling is used to maintain class balance across splits.

Image Rescaling: Input images are normalized to a pixel range of [0, 1] to optimize convergence during model training.

Data Augmentation: Techniques such as rotation, flipping, and zooming are applied to enrich the training dataset and improve the model's generalization capability.

One-Hot Encoding: Labels are transformed into one-hot encoded vectors, enabling the classification model to predict probabilities for all classes.

2.2 Model Architecture

The Convolutional Neural Network (CNN) serves as the core framework for ASL recognition due to its proven efficiency in image classification tasks. The architecture comprises the following layers:

Input Layer: Accepts 64x64 pre-processed images as input.

Convolutional Layers: Extract spatial features by applying multiple filters, learning essential patterns in ASL gestures.

Pooling Layers: Reduce spatial dimensions to minimize computational complexity while retaining significant features.

Fully Connected Layers: Aggregate the extracted features and map them to the output alphabet classes.

Output Layer: Implements a softmax activation function to generate a probabilistic distribution over 24 ASL classes.

Hyperparameter tuning, including learning rate optimization, batch size adjustment, and epoch selection, is conducted to ensure maximum model efficiency.

2.3 Hardware Components

The hardware setup integrates the following components:

ESP-32 CAM Module: A compact, low-power microcontroller with integrated Wi-Fi and Bluetooth, equipped with a camera for capturing real-time hand gesture images. The module's support for TensorFlow Lite enables efficient on-device processing.

Power Supply: A rechargeable battery provides reliable power to the ESP-32 CAM module.

Display: A 16x2 Liquid Crystal Display (LCD) visualizes the recognized ASL gestures, offering immediate feedback to users.

2.4 Software Components

The software infrastructure focuses on seamless integration between the deep learning model and hardware components:

TensorFlow Lite Model: A lightweight, optimized version of the CNN model designed for efficient inference on edge devices with constrained resources.

Model Conversion: The trained CNN model is converted into TensorFlow Lite format to ensure compatibility with the ESP-32 CAM module, enabling real-time gesture recognition.

3.METHDOLOGY

1. **Problem Definition:** Clearly define the objective of creating a system capable of recognizing ASL alphabets to enhance accessibility for individuals with hearing or speech impairments.
2. **Dataset Preparation:** Collect and preprocess a comprehensive dataset of ASL hand gesture images. Ensure the dataset covers all 24 excluding (J&Z) alphabet gestures with sufficient class balance and diversity.
3. **Model Development:** Design a CNN-based model tailored for ASL recognition. Configure the model with convolutional and pooling layers to extract features and optimize fully connected layers for classification.
4. **Training and Validation:** Train the model using the prepared dataset, applying data augmentation and regularization techniques like dropout to prevent overfitting. Validate the model on unseen data to fine-tune hyperparameters.
5. **Model Optimization:** Convert the trained model into TensorFlow Lite format to enable deployment on edge devices like the ESP-32 CAM. Optimize for performance and efficiency.
6. **Evaluation:** Evaluate the system's performance using metrics such as accuracy, F1 Score and precision Visualize results through confusion matrices and learning curves
7. **Hardware Integration:** Program the ESP-32 CAM module to capture images and process them using the TensorFlow Lite model. Ensure seamless integration between hardware components and software.
8. **Deployment:** Deploy the system for real-world use, ensuring stability and usability. Test on real-time scenarios to validate its effectiveness

4. IMPLEMENTATION

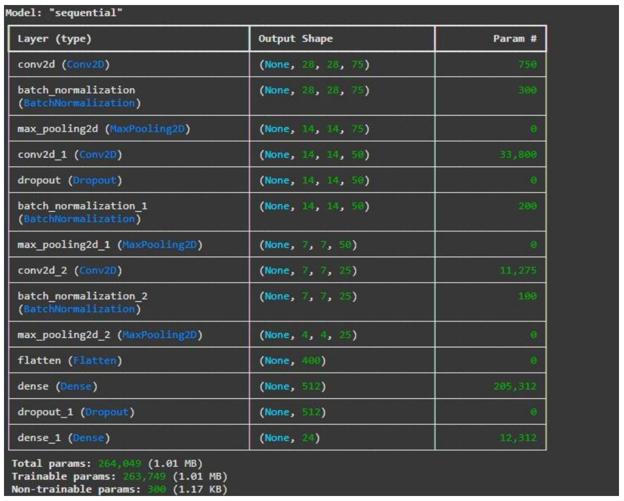
1. **Image Capture:** The ESP-32 CAM captures high-resolution images frames of hand gestures in real time.
2. **Preprocessing:** Captured images undergo resizing, normalization, and other transformations to match the input requirements of the TensorFlow Lite model.

3. **Gesture Recognition:** The preprocessed images are classified by the trained CNN model running on the ESP-32 CAM. The model predicts the ASL letter corresponding to the gesture.

4. **Output Display:** Recognized letters are displayed on the LCD, facilitating real-time feedback and potential integration with additional applications, such as text or speech generation.

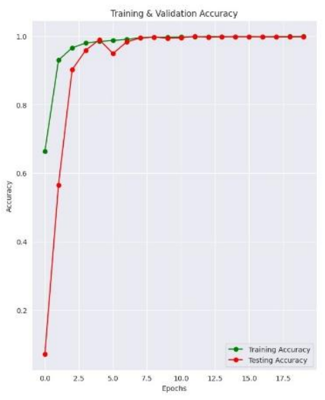
5. RESULTS:

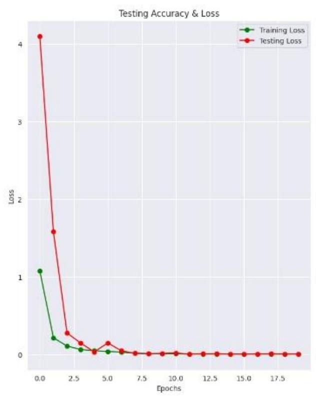
Model Architecture:



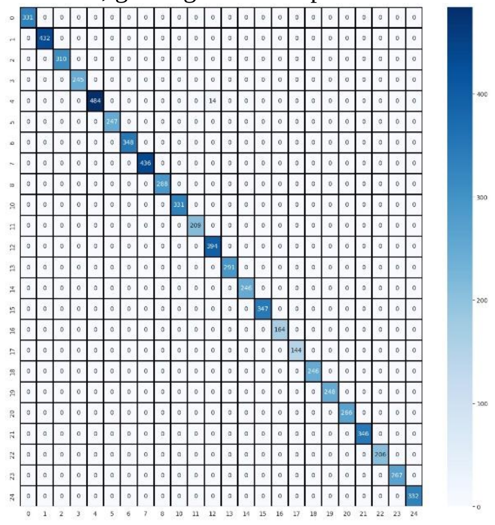
```
Model: "sequential"
┌──────────────────────────────┬─────────────────────┬──────────┐
│ Layer (type)                 │ Output Shape        │  Param # │
├──────────────────────────────┼─────────────────────┼──────────┤
│ conv2d (Conv2D)              │ (None, 28, 28, 75)  │      750 │
├──────────────────────────────┼─────────────────────┼──────────┤
│ batch_normalization          │ (None, 28, 28, 75)  │      300 │
│ (BatchNormalization)         │                     │          │
├──────────────────────────────┼─────────────────────┼──────────┤
│ max_pooling2d (MaxPooling2D) │ (None, 14, 14, 75)  │        0 │
├──────────────────────────────┼─────────────────────┼──────────┤
│ conv2d_1 (Conv2D)            │ (None, 14, 14, 50)  │   33,800 │
├──────────────────────────────┼─────────────────────┼──────────┤
│ dropout (Dropout)            │ (None, 14, 14, 50)  │        0 │
├──────────────────────────────┼─────────────────────┼──────────┤
│ batch_normalization_1        │ (None, 14, 14, 50)  │      200 │
│ (BatchNormalization)         │                     │          │
├──────────────────────────────┼─────────────────────┼──────────┤
│ max_pooling2d_1 (MaxPooling2D)│ (None, 7, 7, 50)   │        0 │
├──────────────────────────────┼─────────────────────┼──────────┤
│ conv2d_2 (Conv2D)            │ (None, 7, 7, 25)    │   11,275 │
├──────────────────────────────┼─────────────────────┼──────────┤
│ batch_normalization_2        │ (None, 7, 7, 25)    │      100 │
│ (BatchNormalization)         │                     │          │
├──────────────────────────────┼─────────────────────┼──────────┤
│ max_pooling2d_2 (MaxPooling2D)│ (None, 4, 4, 25)   │        0 │
├──────────────────────────────┼─────────────────────┼──────────┤
│ flatten (Flatten)            │ (None, 400)         │        0 │
├──────────────────────────────┼─────────────────────┼──────────┤
│ dense (Dense)                │ (None, 512)         │  205,312 │
├──────────────────────────────┼─────────────────────┼──────────┤
│ dropout_1 (Dropout)          │ (None, 512)         │        0 │
├──────────────────────────────┼─────────────────────┼──────────┤
│ dense_1 (Dense)              │ (None, 24)          │   12,312 │
└──────────────────────────────┴─────────────────────┴──────────┘
 Total params: 264,049 (1.01 MB)
 Trainable params: 263,749 (1.01 MB)
 Non-trainable params: 300 (1.17 KB)
```

Training Metrics: A learning curve demonstrated consistent improvement in accuracy while minimizing loss, indicating effective training.

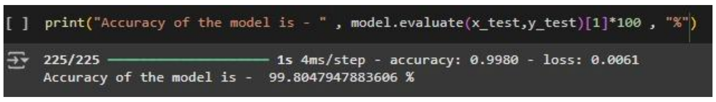




Confusion Matrix: Visualization of classification results highlighted the model's strengths and occasional misclassifications, guiding further improvements

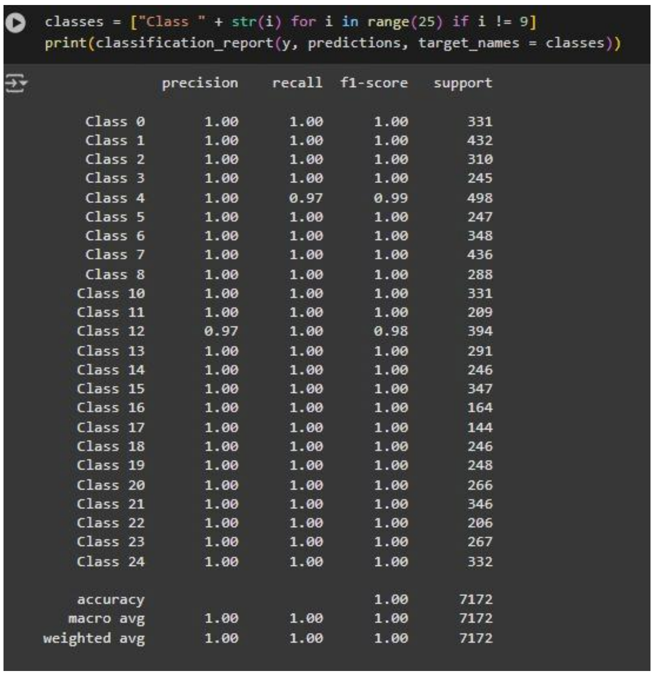


Test Accuracy: The system achieved a final accuracy of 98%, validating its robustness on unseen data.



```
[ ]  print("Accuracy of the model is - " , model.evaluate(x_test,y_test)[1]*100 , "%")

     225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9980 - loss: 0.0061
     Accuracy of the model is -  99.8047947883606 %
```



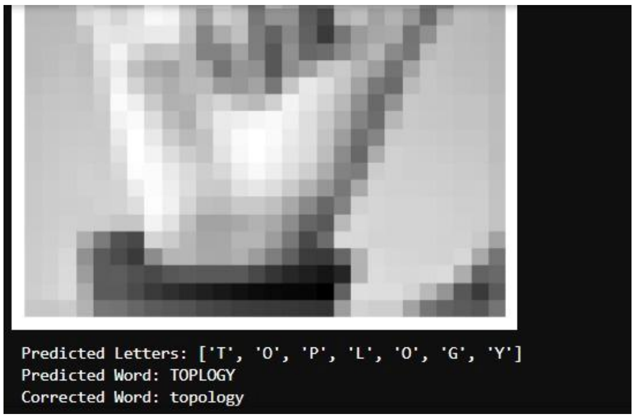
```
classes = ["Class " + str(i) for i in range(25) if i != 9]
print(classification_report(y, predictions, target_names = classes))

              precision    recall  f1-score   support

     Class 0       1.00      1.00      1.00       331
     Class 1       1.00      1.00      1.00       432
     Class 2       1.00      1.00      1.00       310
     Class 3       1.00      1.00      1.00       245
     Class 4       1.00      0.97      0.99       498
     Class 5       1.00      1.00      1.00       247
     Class 6       1.00      1.00      1.00       348
     Class 7       1.00      1.00      1.00       436
     Class 8       1.00      1.00      1.00       288
    Class 10       1.00      1.00      1.00       331
    Class 11       1.00      1.00      1.00       209
    Class 12       0.97      1.00      0.98       394
    Class 13       1.00      1.00      1.00       291
    Class 14       1.00      1.00      1.00       246
    Class 15       1.00      1.00      1.00       347
    Class 16       1.00      1.00      1.00       164
    Class 17       1.00      1.00      1.00       144
    Class 18       1.00      1.00      1.00       246
    Class 19       1.00      1.00      1.00       248
    Class 20       1.00      1.00      1.00       266
    Class 21       1.00      1.00      1.00       346
    Class 22       1.00      1.00      1.00       206
    Class 23       1.00      1.00      1.00       267
    Class 24       1.00      1.00      1.00       332

    accuracy                           1.00      7172
   macro avg       1.00      1.00      1.00      7172
weighted avg       1.00      1.00      1.00      7172
```

Test Result:



```
Predicted Letters: ['T', 'O', 'P', 'L', 'O', 'G', 'Y']
Predicted Word: TOPLOGY
Corrected Word: topology
```

CONCLUSION

The project successfully demonstrates the development of a deep learning-based system for recognizing American Sign Language (ASL) alphabets. By employing a Convolutional Neural Network (CNN) architecture, optimized with techniques like data augmentation and dropout, the model achieved notable accuracy in classifying ASL gestures.
This innovation enhances communication accessibility for deaf and hard-of-hearing communities. Future improvements could include expanding the dataset to dynamic signs

REFRANCES

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
2. Chollet, F. (2017). Deep Learning with Python. Manning Publications.
3. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
4. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
5. https://www.kaggle.com/datasets/datamunge/sign-language-mnist?select=american_sign_language.PNG

6.Fritz AI. (n.d.). *Exploring sign language recognition techniques with machine learning*. Retrieved from https://fritz.ai/sign-language-recognition-techniques-with-machine-learning/

7.Stanford University. (2024). *Sign language recognition with convolutional neural networks*. Retrieved from https://cs231n.stanford.edu/2024/papers/sign-language-recognition-with-convolutional-neural-networks.pdf