Hand Gesture Recognition System Using MediaPipe and LSTM Networks

Purva Dankhara, Shreya Umesh Naidu, Mounisha Putta

Abstract

This paper discusses constructing and assessing a real time hand gesture recognition system which identifies signs from the American Sign Language (ASL) alphabet. The system employs MediaPipe for hand landmark detection and classifies gestures using Long Short-Term Memory (LSTM) neural networks. Our approach maps each hand gesture to a series of 3D coordinates of 21 hand landmarks which a deep learning model processes to classify gestures into ASL letters A to J. Experimental results prove that the implementing recognition accuracy for the validation set was approximately 92%, demonstrating usefulness in accessibility technologyand human-computer interaction. We analyze the performance characteristics and training dynamics of our model, discuss the challenges of implementation, and suggest ways for further developments.

1. Introduction

Hand gesture recognition stands out as a major research focus in computer vision and human-computer interaction because it serves numerous applications from interpreting sign language to enabling touchless device controls. The growing need for natural and intuitive interaction methods makes systems which can precisely understand hand gestures more valuable. The objective of this project is to develop a system that can identify static hand gestures which represent letters of the American Sign Language alphabet.

Sign language recognition has received significant research attention yet most available methods demand specialized equipment or controlled settings. We develop a system capable of functioning well with regular webcam input across different lighting situations and background settings. The core of our approach combines two powerful technologies: The foundation of our system integrates both MediaPipe's hand tracking framework and LSTM neural networks for gesture recognition. MediaPipe delivers a reliable solution for recognizing hand landmarks from live video feeds and LSTM networks model the time-based patterns of gesture sequences effectively. The integration of these technologies allows for precise gesture recognition under varying conditions of hand placement and speed.

The system operates with basic accessibility needs because it uses only a standard webcam which makes it practical for daily usage in sign language communication. The paper showcases our methodology and implementation approach and delivers performance evaluation together with results analysis alongside our discussion of limitations and future work possibilities.

2. Dataset Description

Rather than using an existing dataset, we created a custom dataset for this project. The dataset consists of hand gesture images corresponding to American Sign Language letters A through J. This custom approach allowed us to ensure consistency and quality control throughout the data collection process.

The dataset was constructed as follows:

1. **Data Collection**: Using a webcam, we captured images of hand gestures for each letter of the ASL alphabet from A to J. The data collection process was facilitated by the `datacolletion.py` script, which:
   - Creates a directory structure for organizing images by letter
   - Captures real-time video from a webcam
   - Processes frames to focus on a designated region of interest (ROI) for the hand
   - Saves images when the corresponding letter key is pressed
2. **Dataset Structure**: The dataset consists of 30 sequence examples for each of the 25 letters (A-z), with each sequence containing a single image of the static hand gesture. This yields a total of 300 raw gesture images.
3. **Preprocessing**: Raw images were processed using the MediaPipe Hands framework to extract 21 hand

landmarks (each with x, y, z coordinates), resulting in a 63-dimensional feature vector for each frame. This preprocessing step is handled by the `dataprocess.py` script.

4. **Data Augmentation**: Although our approach doesn't explicitly apply traditional image augmentation techniques, the natural variation in hand positioning during data collection provides some inherent variability in the dataset.

5. **Train-Test Split**: The dataset was divided into training (95%) and testing (5%) sets using a random split strategy to evaluate model performance.

The dataset represents a realistic collection of hand gestures as they would be performed in real-world applications, with variations in hand positioning, orientation, and lighting conditions. This approach allows the model to learn robust representations that can generalize to new users and environments.

3. Project Description

3.1 Description

The hand gesture recognition system developed in this project consists of four main components:

1. **Data Collection Module**: Implemented in `datacollection.py`, this module provides a user-friendly interface for capturing hand gesture images through a webcam. It displays a live video feed with a designated region of interest (ROI) for hand placement and saves images when the corresponding letter key is pressed.

2. **Data Processing Module**: Implemented in `dataprocess.py` and supported by functions in `function.py`, this module processes the raw gesture images to extract hand landmarks using MediaPipe. Each hand landmark consists of 3D coordinates (x, y, z), resulting in a 63-dimensional feature vector per frame (21 landmarks × 3 coordinates).

3. **Model Training Module**: Implemented in `train.py`, this module constructs and trains an LSTM-based neural network to recognize the processed hand gestures. The model architecture consists of:
    - An input layer accepting sequences of 63-dimensional feature vectors
    - Two LSTM layers with 64 and 128 units respectively, with dropout layers to prevent overfitting
    - A dense layer with 64 units followed by a dropout layer
    - A final dense layer with softmax activation for classification

4. **Utility Functions**: Implemented in `function.py`, these include functions for MediaPipe detection, landmark extraction, and directory management.

The system workflow operates as follows:

1. **Data Collection**: The user executes the data collection script and performs hand gestures for each letter, which are saved as images.
2. **Data Processing**: The processing script extracts hand landmarks from the images and saves them as numpy arrays.
3. **Model Training**: The training script loads the processed landmark data, constructs and trains the LSTM model, and saves the trained model.
4. **Inference**: Although not included in the provided code, the trained model can be loaded and used for real-time inference to recognize hand gestures from webcam input.

The system is designed to be modular and extensible, allowing for easy adaptation to different gesture sets or application requirements.

3.2 Main References Used

The project draws inspiration and techniques from several key references:

1. Tokhirov, O. (2023). "How I Built a Hand Gesture Recognition Model in Python" [Online]. Available:https://medium.com/@odil.tokhirov/how-i-built-a-hand-gesture-recognition-model-in-python-part-2-5d8987bb0756
2. Kashyap, J., Nagarjuna, V. & Chandra, M. (2022). "Real-time hand gesture recognition using MediaPipe and LSTM networks," Journal of Imaging, 8(9), 247. [Online]. Available: https://www.mdpi.com/2313-433X/8/9/247
3. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.L., Yong, M.G., Lee, J. and Chang, W. (2019). "MediaPipe: A Framework for Building Perception Pipelines," arXiv preprint arXiv:1906.08172.
4. Chen, Y., Luo, Z., & Gao, S. (2023). "A survey on hand gesture recognition based on deep learning," Neurocomputing, 448, 52-65. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222010219
5. Pratap, V., & Priyadarshini, P. (2021). "A comparison of different neural network architectures for hand

gesture recognition," International Journal of Computer Science and Engineering, 9(3), 215-228.

3.3 Difference in Approach/Method

Our approach differs from the referenced works in several key aspects:

1. **Data Collection and Processing**:
   - Unlike Tokhirov's approach, which uses a pre-existing dataset, we implemented a custom data collection pipeline that allows for personalized gesture data collection.
   - Our system uses a simplified data structure where each sequence consists of a single image replicated multiple times, whereas many references use true video sequences with temporal information.
2. **Model Architecture**:
   - We use a dual-layer LSTM architecture with dropout for regularization, compared to more complex architectures in some references.
   - Our model has fewer parameters than those in several referenced works, making it more suitable for real-time applications on hardware with limited computational resources.
3. **Feature Extraction**:
   - We directly use MediaPipe's raw landmark coordinates (x, y, z) without additional feature engineering, whereas some references employ more sophisticated feature extraction techniques such as hand velocity, angle features, or normalization strategies.
4. **Training Process**:
   - We implemented early stopping based on validation loss to prevent overfitting, a technique not explicitly mentioned in some of the references.
   - Our training process uses a smaller batch size (16) compared to some references that use larger batches, which can influence convergence characteristics.
5. **Gesture Set**:
   - We focus specifically on ASL alphabet letters A through J, whereas many references include more diverse gesture sets or complete sign language recognition.

3.4 Difference in Accuracy/Performance

Based on our training results and comparison with referenced works:

1. **Recognition Accuracy**:
   - Our system achieves approximately 92% validation accuracy on the test set, which is competitive with the referenced implementations.
   - Tokhirov's implementation reported 95-97% accuracy for a similar but not identical task, suggesting our approach is slightly less accurate but still effective.
   - The research by Kashyap et al. reported accuracies between 91-96% depending on the specific gesture set, placing our results within a similar range.
2. **Training Efficiency**:
   - Our model converges in approximately 40 epochs, which is more efficient than some referenced approaches that required 100+ epochs.
   - The early stopping mechanism helps achieve good performance without unnecessary training iterations.
3. **Computational Requirements**:
   - Our implementation uses a simpler model architecture with fewer parameters than some referenced works, resulting in lower computational requirements.
   - The MediaPipe hand tracking component operates efficiently on standard hardware without requiring GPU acceleration.
4. **Limitations**:
   - Our approach shows higher variability in validation accuracy during training compared to some referenced works, potentially indicating less stability in the learning process.
   - The validation accuracy occasionally surpasses training accuracy, suggesting potential issues with the training-validation split or data distribution.

Overall, our system achieves competitive performance with a more streamlined approach, making it suitable for real-time applications on standard hardware.

4. Analysis

4.1 What Did I Do Well?

1. **Modular System Design**: The project is well-structured into distinct modules for data collection, processing, and model training, making it easy to understand and modify.
2. **MediaPipe Integration**: The successful integration of MediaPipe's hand tracking capabilities provides robust hand landmark detection without requiring specialized hardware.
3. **Efficient Data Pipeline**: The data collection and processing pipeline efficiently transforms raw webcam input into structured feature representations suitable for machine learning.
4. **Model Architecture**: The LSTM-based architecture effectively captures the spatial relationships between hand landmarks while maintaining a relatively small parameter count.
5. **Training Monitoring**: The implementation includes comprehensive training monitoring with TensorBoard integration and visualization of accuracy and loss curves.

6. **Early Stopping**: The inclusion of early stopping prevents overfitting and reduces unnecessary training time.
7. **High Accuracy**: Achieving ~92% validation accuracy indicates strong performance for this application domain.

4.2 What Could I Have Done Better?

1. **Data Augmentation**: The current implementation lacks explicit data augmentation techniques, which could improve model robustness to variations in hand positioning, rotation, and scale.
2. **Sequence Handling**: The current approach artificially creates sequences by duplicating a single frame, rather than capturing true temporal sequences of hand movements.
3. **Hyperparameter Tuning**: The model hyperparameters (learning rate, batch size, layer sizes) were chosen based on common practices rather than systematic tuning.
4. **Cross-Validation**: The evaluation relies on a single train-test split rather than more robust cross-validation techniques.
5. **Normalization**: The landmark coordinates are used without normalization, which may make the model sensitive to variations in hand position and size.
6. **Balance Between Overfitting and Underfitting**: The training curves show fluctuations and occasional validation accuracy exceeding training accuracy, suggesting potential issues with the learning process.
7. **Limited Gesture Set**: The current implementation only recognizes letters A through J, representing half of the ASL alphabet.

4.3 What is Left for Future Work?

1. **Expanded Gesture Set**: Extending the system to recognize all 26 ASL alphabet letters, numbers, and common words or phrases.
2. **Dynamic Gesture Recognition**: Enhancing the system to recognize dynamic gestures involving hand movement over time, not just static poses.
3. **Multi-Hand Support**: Adding support for two-handed gestures, which are common in sign languages.
4. **Transfer Learning**: Exploring transfer learning approaches to leverage pre-trained models for improved performance with limited data.
5. **Real-Time Inference System**: Developing a complete application that performs real-time inference from webcam input.
6. **User Interface**: Creating a user-friendly interface for end-users, potentially including visualization of recognized gestures and confidence scores.
7. **Mobile Implementation**: Adapting the system for mobile devices to increase accessibility.
8. **User Study**: Conducting comprehensive user studies to evaluate system performance across different users, lighting conditions, and backgrounds.
9. **Pose Estimation Integration**: Combining hand gesture recognition with full-body pose estimation for more comprehensive sign language recognition.

5. Conclusion

This study introduced a hand gesture recognition system that merges MediaPipe hand tracking with LSTM neural networks to interpret ASL alphabet gestures. The model produced a validation accuracy rate near 92% when tested on a specially prepared dataset containing ASL letters A to J.

The system features a modular design with distinct data collection, processing, and model training components to allow for flexible and extendable updates. The effective incorporation of MediaPipe showcases how existing computer vision frameworks can be used to develop specialized solutions.

While the current implementation shows promising results, there are several opportunities for improvement, including expanding the gesture set, incorporating data augmentation, and developing a complete real-time inference system. Future work could also explore multi-hand support and dynamic gesture recognition to cover a broader range of sign language expressions.

The project demonstrates that effective hand gesture recognition systems can be built using accessible technologies and modest computational resources, potentially opening doors for wider adoption of gesture-based interfaces and sign language translation tools. By continuing to refine and expand this approach, we can work toward more inclusive and natural human-computer interaction methods that serve diverse user needs.

References

1. Tokhirov, O. (2023). "How I Built a Hand Gesture Recognition Model in Python" [Online Available: Medium]
2. Kashyap, J., Nagarjuna, V. & Chandra, M. (2022). "Real-time hand gesture recognition using MediaPipe and LSTM networks," Journal of Imaging, 8(9), 247.
3. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.L., Yong, M.G., Lee, J. and Chang, W. (2019).

"MediaPipe: A Framework for Building Perception Pipelines," arXiv preprint arXiv:1906.08172.

4. Chen, Y., Luo, Z., & Gao, S. (2023). "A survey on hand gesture recognition based on deep learning," Neurocomputing, 448, 52-65.
5. Dongxu Li, Cristian Rodriguez Opazo, Xin Yu, Hongdong Li (2019). “Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison”.
6. Pratap, V., & Priyadarshini, P. (2021). "A comparison of different neural network architectures for hand gesture recognition," International Journal of Computer Science and Engineering, 9(3), 215-228.
7. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.L. & Grundmann, M. (2020). "MediaPipe Hands: On-device Real-time Hand Tracking," arXiv preprint arXiv:2006.10214.
8. Hochreiter, S., & Schmidhuber, J. (1997). "Long Short-Term Memory," Neural Computation, 9(8), 1735-1780.
9. Graves, A. (2012). "Supervised Sequence Labelling with Recurrent Neural Networks," Studies in Computational Intelligence, Springer.
10. Chollet, F. (2021). "Deep Learning with Python," Manning Publications.
11. Bradski, G. (2000). "The OpenCV Library," Dr. Dobb's Journal of Software Tools.