Synchronic Automatic Sign Language Recognition

Zidong Xu(zx2507), Caiwu Chen(cc4786)

1. Synopsis

a.overview

The speech-impaired have difficulty in communicating and interacting with people. Bridging the incapacity of communication between the sign language user and spoken language user, the Automatic Sign Language Recognition (ASLR) system holds tremendous promise. The process of translating sign language into written text can bring convenience for the hearing-impaired community and enhance accessibility and social unity to a large extent. With the development of computer vision and deep learning, the ASLR system has become feasible and capable by utilizing models, such as Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs). The Long Short-Term Memory (LSTM) is a classic structure for processing dynamic sign language recognition in videos [1]. CNN + Vision Transformer (ViT): ViT is better at modeling long-distance dependencies in image processing and can be used in combination with CNN [2,3]

We will build upon the project previously investigated by Zidong Xu, which demonstrated the capability to recognize sign language against a pure background using CNNs and MLPs at the image level. For this project, we successfully developed a program capable of identifying sign languages and translating them into text in real-time. Due to the diversity and unique cultural interpretation of sign language regarding its area, we chose the character-based American Sign Language dataset to fit into our designed CNN-MLP concatenated model with the MediaPipe tool to dominate noise, aiming to enhance the robustness and generality of the model. We also built an Android-level ASLR app while contributing to improved human interaction.

B. Value to the community

The development of ASLR offers meaningful contributions to both the technical community and society at large. For the hearing-impaired community, our ASLR system provides a practical solution for real-time communication, reducing barriers in daily interactions, education, and public services. Our implementation demonstrated the feasibility and effectiveness of the combination of CNNs and MLPs with the Mediapipe, the hand-tracking tool in the field of synchronous sign language translation. The deployment of a mobile platform serves as a potential ASLR solution.

2. Related work and Methodology

a. CNN

Convolutional Neural Networks (CNNs) have been widely used in image classification tasks due to their ability to extract visual features. In Automatic Sign Language Recognition (ASLR),we use CNNs for capturing differences in hand position and shapes from image data. In

our work, we implement a custom CNN branch composed of three convolutional blocks, see Fig. 1, each followed by batch normalization, pooling, and dropout. The CNNs' output is then flattened and processed with the output of the MLP model.

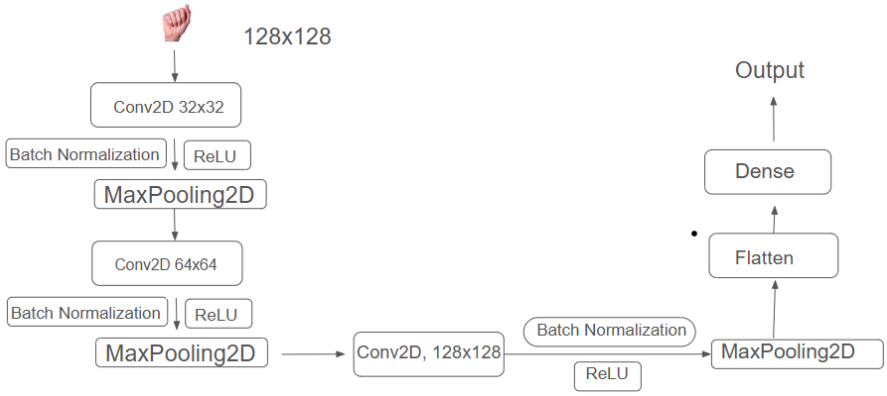

Fig1. Three Blocks CNNs

b. MediaPipe Hand Landmarker

The MediaPipe Hand Landmarker is a cross-platform, Google-developed framework. The Hand Landmarker operates on image data with a machine learning model as a continuous stream and outputs hand landmarks in world coordinates, for both left and right hands. In Automatic Sign Language Recognition, the MediaPipe Hand Landmarker detects 21 3D landmarks per hand with high accuracy, see Fig. 2, even though some joints are eclipsed or in rapid movement. Through tracking the movement of 21 predefined keypoint features to picture the gesture of the whole hand, it provides robust information for the MLP branch of our model and reduces the influence of the noisy background considering the complexity and distinctiveness of the situation of the ASLR.

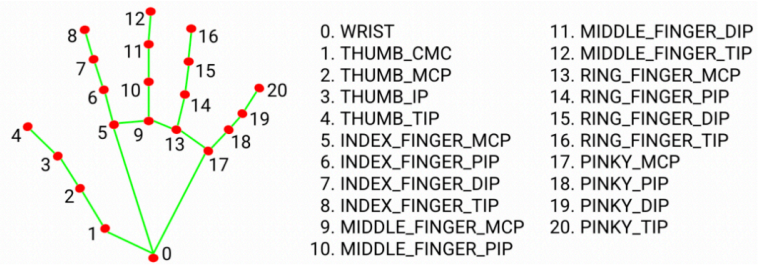

Fig2. MediaPipe

c. MLP

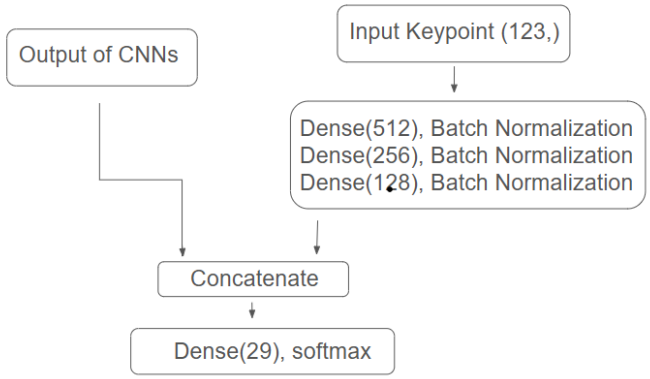

Fig3. MLP model

Multi-Layer Perceptrons (MLPs) are effective in handling structured, low-dimensional input features. In our model, MLP handles the keypoint generated from MediaPipe and passes through multiple fully connected layers with dropout and batch normalization to produce a compact representation of the hand pose, see Fig. 3.

Unlike approaches that solely rely on CNNs, our dual-stream design captures complementary information from both raw RGB images and structured hand landmark data. This hybrid architecture improves recognition accuracy and resilience to noisy input, especially in uncontrolled environments.

3. Research Question

1. How to detect the change of sign language among a series of images, i.e., video?

The question refers to the foundation of this project, which is to build a synchronic automatic sign language recognition system that leverages the well-trained model that we design, and instead of predicting a single image, but predicts a continuous stream of images. To realize this, we defined a model combining CNNs taking real-time video processing image frames, and MLPs relying on the MediaPipe Hand Landmarker to produce keypoint features. The system detects when a new sign begins and ends by analyzing pattern changes across the video stream.

2. How can the sign language be identified from the noisy background?

Since the original ASL dataset was created in a pure background. We initially built a pure CNN model, but it turns out that the initial version of the CNN model cannot achieve a reasonable result. The CNN alone model struggled with the noisy background rather than solely focusing on the hand information. To address this problem, we introduce MediaPipe's Hand LandMarker module to extract the hand landmarks and tracking keypoints. The structured keypoints were then fed into an MLP alongside the CNN image, enabling the model to focus on

hand shape and motion while ignoring irrelevant background details. This hybrid architecture significantly improved robustness and accuracy in more complex visual environments.

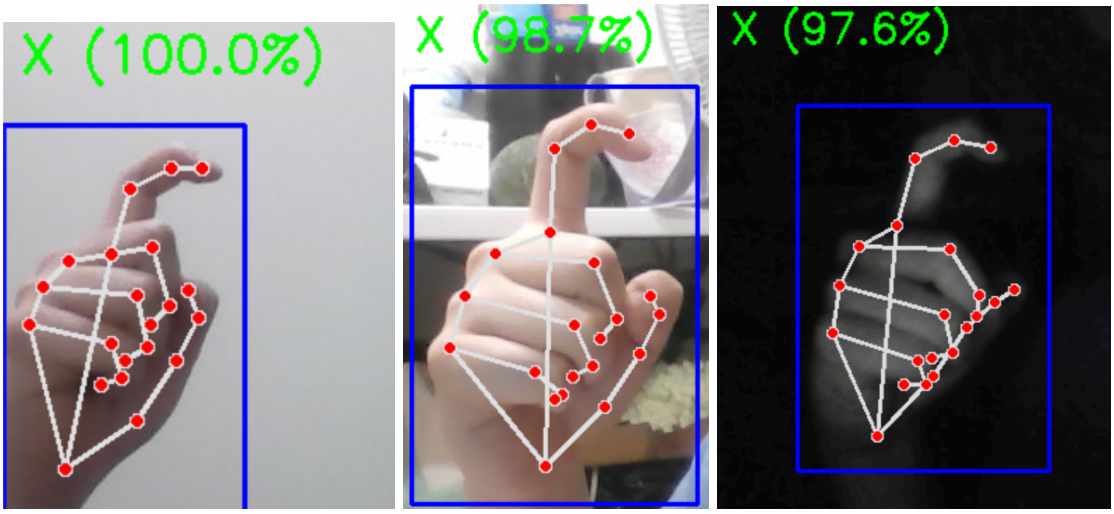

Fig 4. Synchronic automatic sign language recognition under different circumstances

However, even though we take advantage of the MediaPipe to reduce the effect of noisy background, it is no doubt that this factor may decrease the confidence of our model's predictions of characters, see Fig. 4. The image from left to right demonstrates our model's performance under the situation of pure background, noisy background, and in a dark situation, and the gesture performed character "X" in both situations. The result shows that though our model can predict all characters correctly in all circumstances, the confidence drops from 100% to 98.7% to 97.6%. This demonstrates that while MediaPipe greatly mitigates the effect of background noise, it does not fully eliminate its influence, particularly on prediction certainty.

3. How to embed the sign language translator into a personal computer and mobile platform?

To ensure that our sign language recognition system is accessible in personal computers, we developed and tested the model in Python using TensorFlow and integrated it with a real-time webcam interface. This setup allows users to perform gestures in front of their computer's camera and receive immediate textual translations. The write-up logic in our implementation has great potential to be developed and grants great flexibility for users to design according to their preferences. Given our beginner-level fluency in performing sign language, we designed the system to only write down a predicted character if it maintains the highest confidence for over one second. However, considering the familiarity of the user with the ASL alphabet, we recommend that users fine-tune the write-up logic to best suit their habits.

For mobile deployment, we converted our trained CNN-MLP model into TensorFlow Lite (TFLite) format. We then embedded the TFLite model into a custom-built Android application that interfaces with and grants the device's camera in real time. Hand landmarks are also extracted using MediaPipe's mobile-optimized hand tracking module. To save power and fit in a

smaller mobile phone screen, we don't show the 21 keypoint features generated by MeidaPipe on screen visually. This cross-platform compatibility broadens the reach of our ASLR system and enables real-time accessibility for users in various daily environments.

4. How can the robustness of an automatic sign language recognition model be improved for accurate translation?

Robustness is critical for deploying an ASLR system on either a hardware platform or a mobile platform. In real-life applications, factors such as lighting variation, background clutter, hand occlusion, and diverse user behavior can affect recognition accuracy. To improve the robustness of our model, we design a hybrid architecture that combines visual features extracted by a CNN with structured hand landmark data provided by MediaPipe, processed through an MLP. Our beginner-friendly system enables users to fine-tune recording logic and write down ideal characters, given enough time for the system to determine.

Admittedly, our current model cannot always predict with perfect accuracy. This is due to several factors: limited training data diversity, occasional instability in hand landmark detection, and the difficulty in distinguishing visually similar gestures. These challenges highlight the need for continued refinement through larger datasets, improving preprocessing, or integrating temporal models like LSTMs or Transformers for better context awareness over time.

Deliverable:

The code of the framework and dataset are included in this GitHub repository link: https://github.com/xuefeng16513/6156_Final_Project. The README file also contains a short description of the project and usage instructions. Due to GitHub's file size limitations, the full dataset, trained model files, and Android deployment package are hosted on Google Drive. Users can access and download these resources directly via the links provided in the README.

Contribution and Self-Evaluation

Caiwu Chen:

Through the development of our Synchronic Automatic Sign Language Recognition (ASLR) system, we gained substantial experience in applying machine learning and computer vision techniques to real-world accessibility challenges. We are proud of our successful implementation of a dual-input model that combines CNN-based image recognition with structured hand landmark data using MediaPipe, and successfully transplanted an Android app to make it more accessible to the community.

However, our implementation was not successful at first. Since this project was inherited from image-level CNNs and MLPs image classification tasks. So we first also apply the same idea but change the input as a sequence of image streams. However, it turns out that it cannot give a reasonable prediction for sign language. We analyze that the reason may be because of

the limited trained dataset. But even when we changed to a larger training dataset, the result did not improve. While we were searching online to see if there was a solution for this issue, MediaPipe caught our eye. Even though we originally planned to do either the CNN model or the MLP model, we ended up with the hybrid model, achieving a high accuracy and the robustness of the system across varying backgrounds and lighting conditions.

At first, we simply took it as a desktop-level system. However, since the Professor mentioned considering embedding it into a mobile device, a hearing-impaired person will not have to stand next to a PC to do all the sign language recognition. And it makes sense. Since I lacked experience in mobile development, I was afraid that I could not make it. However, thanks to my background in modern software engineering practices and familiarity with problem-solving through documentation and open-source projects, I approached it step by step. With countless resources and existing GitHub examples available, I gradually figured out how to integrate our TFLite model into an Android interface, handle real-time camera input, and embed MediaPipe's hand tracking effectively. This transition from desktop to mobile not only broadened the reach of our system but also helped me grow significantly as a developer.

Zidong Xu:

Our Synchronic Automatic Sign Language Recognition (ASLR) system has completed high-precision recognition of letter gestures and space and deletion gestures. Initially, I tried to use the CNN model I developed in one of the projects to perform image recognition tasks alone, but the effect was not ideal. Although the model's recognition accuracy on the test set reached about 95%, in the real-time recognition task calling the camera, the model's recognition accuracy was very low, and most of the time it could not correctly recognize the gesture. After discussion, we thought that it might be that the image dataset used to train the model was a low-pixel grayscale image, which could not simulate the images taken in the real-time task, so the model's accuracy was low. So I switched to a larger color image dataset, modified the model structure to make it more capable of learning the complex features of color images, and then retrained the model. As a result, the model's accuracy in real-time tasks did increase, but its accuracy still could not meet the requirements, and there were often recognition errors. I think it is very likely that the model learned the noise in the dataset during the learning process, and then output the results through the characteristics of the noise instead of the characteristics of the hand. So I modified the model structure several times, trying to increase the diversity of images through ImageDataGenerator, simulating the natural changes of gesture images in reality, making the model less dependent on information affected by noise such as fixed position and lighting in the image, and adding Dropout, which randomly discards some neurons during training, forcing the model not to rely on a single feature. This version of the CNN model is also stored in the warehouse, located in the cnn_update branch. But the final result is still not ideal. I think this is the most challenging part. I think the model has done the best in all aspects, but the model still learns more noise than real features, which makes its performance in the test set exceed expectations, but its performance in actual tasks is not satisfactory. This prompted me to constantly seek new solutions.

Later we found that MediaPipe will identify the key point information of the hand in the image and output it as a multi-dimensional vector, which reminded me that the principle of MLP model learning is to simulate the neural network input multi-dimensional vector, and then output the results according to different task requirements. So I tried to introduce the MLP model into our project, let the CNN model recognize the image part, let the MLP model recognize the key point part of the hand, and then fuse the output recognition results. After testing, the fusion model performed well in real-time recognition tasks, and can achieve a recognition accuracy of more than 90%. In this process, I have a deeper understanding of the structure and principles of the CNN model and the MLP model, and learned how to design the model structure according to different task requirements and the skills in processing data sets. The complete construction of the sign language recognition system from training to deployment allowed me to understand how the CNN model and the MLP model can be integrated and coordinated, and also allowed me to learn how to apply new technologies such as MediaPipe.

After applying the trained model to the recognition task, I also improved the output interface, adjusted the output letter board to the appropriate size, and implemented the automatic line wrapping, scrolling and multi-frame voting recognition mechanism to avoid misjudgment due to single-frame jitter. After discussion, we decided to port the project to the mobile terminal. Caiwu and I were not familiar with mobile terminal development, but we finally completed the porting work. Caiwu completed most of the porting work, and I only conducted the final usability test.

Reference:

1. S. Mhatre, S. Joshi and H. B. Kulkarni, "Sign Language Detection using LSTM," *2022 IEEE International Conference on Current Development in Engineering and Technology (CCET)*, Bhopal, India, 2022, pp. 1-6, doi: 10.1109/CCET56606.2022.10080705. Sign Language Detection using LSTM | IEEE Conference Publication | IEEE Xplore
2. Nimisha, KP, and Agnes Jacob. “A brief review of the recent trends in Sign language recognition.” *2020 International Conference on Communication and Signal Processing (ICCSP)*, July 2021, pp. 186–190, https://doi.org/10.1109/iccsp48568.2020.9182351.
3. Shin, Jungpil, et al. “Korean sign language recognition using transformer-based deep neural network.” *Applied Sciences*, vol. 13, no. 5, 27 Feb. 2023, p. 3029, https://doi.org/10.3390/app13053029.
4. D. Li, C. R. Opazo, X. Yu, and H. Li, "Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison," *CoRR*, vol. abs/1910.11006, 2019. [Online]. Available: http://arxiv.org/abs/1910.11006.
5. A. C. Duarte, S. Palaskar, D. Ghadiyaram, K. DeHaan, F. Metze, J. Torres, and X. Giró-i-Nieto, "How2Sign: A Large-scale Multimodal Dataset for Continuous American Sign Language," *CoRR*, vol. abs/2008.08143, 2020. [Online]. Available: https://arxiv.org/abs/2008.08143.
6. A. Tayade and A. Halder, "Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning," May 2021. DOI: 10.13140/RG.2.2.32364.03203. Available: https://www.researchgate.net/publication/369945035_Real-time_Vernacular_Sign_Language_Recognition_using_MediaPipe_and_Machine_Learning