Deep Learning American Sign Language Translator

## By Ella Bar, Naomi Green and Daniela Simonovsky

### Problem Statement

Sign languages, such as American Sign Language (ASL), are the primary mode of communication for millions of Deaf and hard-of-hearing individuals. However, barriers still exist between signers and non-signers, often leading to difficulties in everyday communication, education, and accessibility. Traditional solutions such as interpreters or subtitles are not always available, affordable, or scalable.

Our project aims to build a deep learning–based system that can take as input a short video of a single ASL sign and output its corresponding English word. By focusing on isolated signs rather than full sentences, the system provides a simplified but practical step toward real-time ASL translation. A tool such as ours could serve as a foundation for assistive technologies, educational platforms, or communication aids that bridge the gap between signers and non-signers.

An inherent challenge of this task is that signs involve many variations in hand shape, movement, personal style etc., making this a non-trivial task even for humans.

### Approach and Existing Frameworks

For our translator, we leveraged existing computer vision and sequence modeling frameworks to handle the challenges of gesture recognition. Our approach centered on two main components: pose landmark extraction and temporal sequence modeling.

For landmark extraction, we employed MediaPipe, an open-source framework developed by Google that provides real-time, cross-platform perception solutions. MediaPipe’s holistic model enables the detection and tracking of human body landmarks, including hands, arms, and facial features. Specifically, we used it to obtain precise 3D coordinates of key body joints and hand landmarks from each frame of the input video. A datapoint extracted from a single frame consists of 77 3D vectors: 21 for each hand, 33 for the pose, and 2 vectors describing a bounding box around the face. This representation dramatically reduces input complexity by abstracting away raw pixel data and focusing on the geometric structure of the signer’s movements.

To model the temporal dynamics of signing, we used a Bidirectional Long Short-Term Memory (BiLSTM) network implemented in PyTorch. LSTMs are recurrent neural networks (RNNs) that maintain a hidden state across time, enabling them to model dependencies between frames. The **bidirectional** variant processes the sequence in both forward and backward directions, which is useful because signs often depend not only on past motion but also on anticipating how the movement concludes. Our BiLSTM consisted of **two stacked layers**, each with a **hidden size of 256 units**, and produced an output embedding for each time step. These embeddings were aggregated and passed through a **fully connected classification layer** with softmax activation to predict the most likely ASL label. The MediaPipe landmark sequences are fed into the BiLSTM, which learns to classify the sequence into the correct ASL label.

This combination of MediaPipe for robust feature extraction and BiLSTM for temporal modeling allowed us to address the key challenges of sign recognition: capturing both the spatial arrangement of landmarks and their evolution over time.

### Dataset

Our main dataset was [WASL Video](https://www.kaggle.com/datasets/risangbaskoro/wlasl-processed). It contains links to short video clips of a single sign, aggregated from online sources, featuring 2,000 different words. The set itself contains 3-40 samples for each word, however we encountered several problems with this dataset:

1. A substantial percentage of the links were no longer available. Even focusing on the words with the most samples, this left us with 10-15 samples per word.
2. Some words have multiple signs (some due to “dialect” and some due to some English words having several meanings that are translated to distinct ASL signs). This meant that choosing the words to focus on required watching all the videos under a specific label and checking whether they really represent the same sign.

As the dataset was more limited than initially thought, we collected the data in two major ways: filming our own videos and collecting more examples from online. This left us with about 40 videos for each of our chosen words, consistently signed. We then enriched our data further by mirroring each video.

### Preprocessing

Using raw videos as training input has two main downsides. The first is that videos contain a lot of irrelevant information – background, lighting, skin color and clothes, etc. An especially meaningful part of this is positioning and size of the person within the video frame. All this means that even two videos of the same person signing the same word can look very different at the pixel level, which makes it harder for an AI model to “focus” on the relevant parts that we wish to train on. The second is that videos are large files, which means that even on our very limited data, training a model on raw videos takes a lot of time and computation power.

To solve these problems, we employed a preprocessing pipeline using the MediaPipe framework. For each video, every frame was processed by MediaPipe Holistic, which extracts 3D coordinates of pose landmarks (points of interest in the human body). Initially, we used the pose, hand and face landmarks, totalling 543 points. However, 468 of those points are facial features, creating a heavy bias towards the facial expression and face movement of the person in the video. As ASL is mostly signed by hand movements, we wanted to reduce this facial emphasis. We did not want to throw away the facial data completely, as relative positioning of face and hands is often meaningful in ASL. To accomplish this, we used the face points to calculate an axis-aligned bounding box for the face, which reduces the face data to two 3D points.

To address the issue of size and positioning, we normalized the MediaPipe output for each frame such that every axis has mean 0 and standard deviation 1. This is appropriate for ASL as the signer does not move apart from hands and face, so normalizing frame by frame does not lose meaningful information.

In ASL, a given movement when signed with the right hand has the same meaning as when signed with the left. To avoid the model viewing the choice of hand as meaningful, we created a mirrored version of the normalized output described above.

### Training

For our final model, we trained for 100 epochs using Google Colab’s CUDA resources (T4 GPU). We used an 80–20 training–validation split, and the best checkpoint to save was selected based on validation accuracy. Typically, after around 60 epochs, the training accuracy approached or even reached 100%, while validation accuracy plateaued around 80%, indicating clear overfitting. This is expected given that modern models can easily memorize a dataset of only ~200 short clips, especially when the input has been reduced to compact landmark features. Our best validation accuracy was 81.4%.

Before employing MediaPipe (i.e., training directly on raw video), we experimented with three different architectures: a convolutional LSTM, a ResNet50 + BiLSTM hybrid, and an attention-based BiLSTM. We also explored learning rate schedulers and varying dropout rates as strategies to mitigate overfitting. These models achieved 60–80% validation accuracy, but training was far more resource-intensive and required significantly longer to converge — often only reaching a plateau after ~80 epochs. In contrast, the MediaPipe-based pipeline not only reduced input dimensionality and training cost but also delivered slightly better performance with far fewer computational requirements.

### End Product

Our project supports both offline and online usage, meaning both predicting a sign from an input video file (using predict.py) or recording a video live and getting a prediction immediately (using predict\_realtime.py file on a device with a camera).

### Git Structure

Something

### Main Takeaways and Future Work

The main challenge we encountered was lack of training data. It proved to be hard to mitigate and tripling the source data with personal videos did not prevent overfitting entirely. This is expected, as reaching very good results from very little data is a problem at the forefront of AI research.

Employing our MediaPipe preprocessing routine was extremely helpful. It allowed us to run many more training sessions and thus experiment freely with different parameters, GD vs. SGD, loss functions and more. Additionally, we think it helped refine the important parts of each sample, which is evident by the faster plateauing. Refinement also reduces the required data scale, although it was still not enough.

One major limitation of our solution is its restricted vocabulary. While the codebase is designed to support an arbitrary number of signs, the current implementation — and the dataset we enriched — covers only three words. A natural next step would be to expand the dataset to include many more signs and retrain the model accordingly.

With a larger vocabulary in place, the system could be extended to handle full ASL sentences. This would require a model capable of segmenting a sentence into its constituent signs and then translating each in sequence. However, because ASL and English differ in both vocabulary and grammar, a complete end-to-end translation system would also need an NLP layer to restructure the recognized signs into fluent English sentences. In this way, our current sign-level translator could serve as a key building block toward a full ASL-to-English translation pipeline.