

ASL2Text - Deep Learning Final Project

Raviv Segal - 315054122

Omer Ziv - 209224328

August 2025

This project has a [Github repository](#).

Introduction

As times and technology advanced, humanity has seen an increase in awareness of and accommodation for disabilities, especially those which affect communication abilities. Still, a fundamental enduring gap regards hard of hearing/non-verbal individuals who must communicate through sign language. They experience difficulties when interacting with able-bodied people who might not understand sign language, which seems like a very difficult communication gap to bridge. In light of this, we decided to focus our project on this issue, and try to train a neural network to decode and convert signed words to written English text.

We aim to perform transfer learning using a preexisting convolutional neural network in hopes of successfully decoding videos of signed words in ASL (American Sign Language) into written english. We found several projects online that have attempted to use CNNs for ASL to English translation, but most focus on letter recognition [3],[1], whereas we intend to use the WLASL database [4] (Word-Level ASL) which contains signed words and not letters¹.

Methods

Data Acquisition and Preparation

The WLASL dataset contains references to many online videos of people demonstrating various words in ASL. The dataset is curated as a *.json* file containing all relevant information regarding each sample, and is accompanied by a python script to automatically download the videos, which we modified slightly to also sort the videos according to the label in the *.json* file.

To save on resources we had to limit the scope of the classification task, so we decided to focus on just the first 100 words in the dataset, which we saw as rich and diverse enough - containing a high variance of hand gestures and words from different semantic fields. The next step was to sterilize the samples we had - we found that the way we divided the samples into their categories yielded nonhomogeneous classes which would have made

¹Initially we wanted to use 'ASL-LEX2.0' [5] (which is also word-level), but we weren't able to download the videos shown in the website. The WLASL dataset had some size and accessibility limitation, but turned out to be a pretty reliable resource for the most part.

direct translation rather challenging. This happened for two reasons; first, there are a few different ways of signing certain words (similar to synonyms in written English), and there are some words which can have different meanings altogether depending on context (for example ‘cool’ can mean cold, awesome etc.). The first problem is solved simply by ‘choosing a dialect’ - committing to one way of signing the word, but the second is more general and affects any translation task. Whereas for traditionally an attention model would be a viable option to try and capture the context of the word, here it can’t be done because we translate the words as standalone instances. We therefore had to also commit to only one possible meaning per signed word, and since we both don’t understand ASL we cross referenced with the ASL-LEX2.0 [5] to tackle both of these issues, and erase the redundant samples from our dataset.

Working with videos poses an issue; while in essence being the very tokens we have to work with, analyzing a full video is very computationally extensive. Due to limited time and computational resources we had to streamline the process somehow. We did not want to deal with designing a sequential network to decode entire videos, so we tried to approximate this notion - by capturing a subset of frames and stacking them as a 3D tensor.

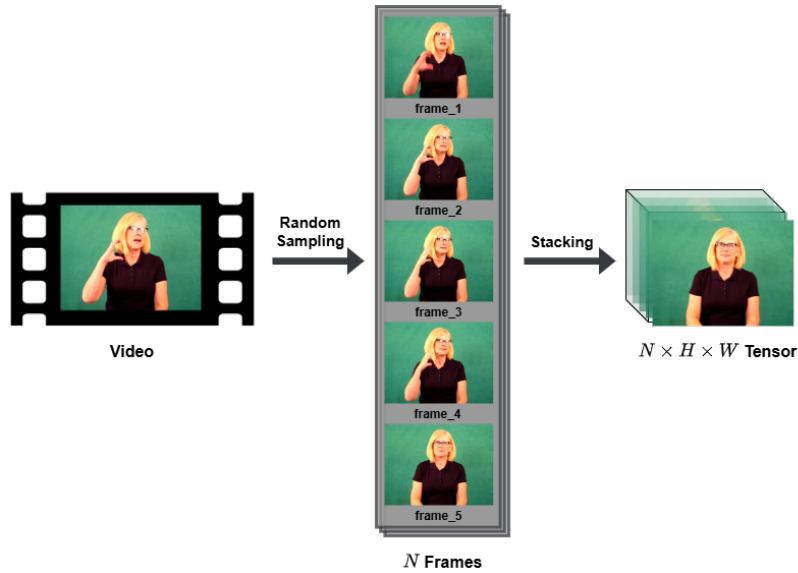


Figure 1: Treating the videos as a static 3D tensor

Preprocessing

Offline Video Sampling

Due to missing/inaccessible online videos, we had some trouble with capturing enough samples, and after the sterilization the dataset was decidedly not diverse enough - with some words having only 3 different samples. The way we went about this was by adding randomization to the frame sampling process in an attempt to widen the variance. We thought of three different ways to do this:

1. Pure random sampling; we could just randomly pick N frames (with or without repetitions) from the video and sort them by temporal order. Though this is a simple way of generating a lot of distinct samples per word, there's no guarantee that these resulting samples will be highly correlated in any sense - for signs which consist of fast/wide movements or different facial expressions we could get vastly different samples for the same words.
2. Simulating a counting process and sampling a frame at each occurrence - this would result in approximately uniformly distributed sampled frames across the duration of the video, which would maintain correlation with high probability for all associated samples. The downside though is that then we'd be working with a variable number of frames per sample, which adds back the layer of complexity we wanted to eliminate by treating each video as a static tensor.
3. The third option, and the one we ultimately chose, is sampling at noisy intervals - fixing the number of frames we wish to capture we calculated equidistant intervals for sampling that number of frames, and then sampled at a random offset. This would result in rather similar samples, varying only slightly. The motivation for this method was noticing that the demonstrators in the videos often did not keep their hands steady, and that most gestures required abrupt precise gestures - so by slightly tweaking and taking close frames we could mimic a dataset with much higher variance. With that we had a larger dataset, which hopefully could emulate a richer one provided we had the resources.

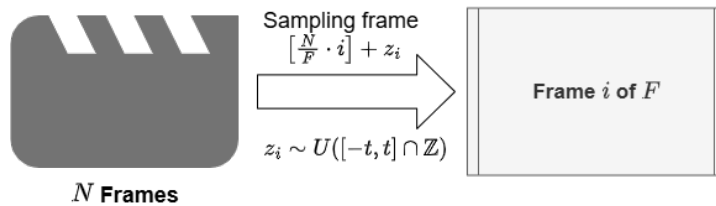


Figure 2: Sampling frames at approximately fixed intervals, offset by a random number of frames per frame

In-Code Augmentations

We added the following augmentations to the code in hopes of mitigating preexisting biases of ResNet and increasing invariances to certain transformations:

1. **Color Jittering:** attempting to reduce biases which are related to colors in the videos. Each demonstrator was wearing a shirt of a certain color and was filmed

on a background of a certain color, and we had to minimize the effect of those factors. At first we considered sidestepping this problem altogether by converting the frames to grayscale, but we decided against it in the hopes of being able to scale the model in the future, and figured the randomness introduced by the jittering will suffice.

2. **Random Horizontal Flip:** We noticed some of the demonstrators were signing with their right or left hand, which of course looks completely different when reading pixel values of an image. We added this augmentation therefore to increase the model’s invariance to this phenomenon and encourage it to detect both possible orientations as the same.

Other than that, we also uniformly rescaled the frames to 224×224 pixels and normalized about $\mu = (0.485, 0.456, 0.406)$, $\sigma = (0.229, 0.224, 0.225)$, as recommended in the pytorch documentation for ResNet.

Architecture and Training

We decided to use the publicly available CNN Resnet-18 whose architecture is presented in [2] as a backbone, pretrained on Imagenet. The resulting sequences of features are fed into a GRU with a hidden layer size of 256, and finally a fully connected linear layer for the final classification. In each of the experiments described in the upcoming segment

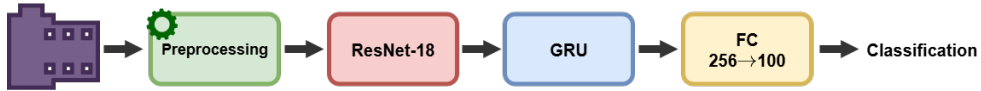


Figure 3: The architecture we used

we used the ADAM optimizer with a fixed learning rate of 10^{-4} and a batch size of 16. Each experiment was run for 10 epochs, and evaluated using Cross-Entropy loss, which is a standard loss function when trying to solve a multi-class classification problem such as this one. Videos were sampled at five frames per clip (intervals of 0.25) and with a range of $t = 5$ to keep runs computationally feasible with the resources we had available.

Experimentation, Iteration and Results

Our attempts started by running the model as is, with a standard train-test split of 80%-20% for each label, maintaining the same sample distribution for both datasets.

First Attempt

Running for 10 epochs with the settings specified in the previous segment, we got a train accuracy of 100% and a test accuracy of 99.29%. This was very surprising, as we didn't expect to reach such high accuracy out of the gate, and we suspected this was due to overfitting; since our dataset was very limited even after the added sampling noise, samples generated from the same videos were very similar to each other - having the same background and demonstrator meant that most of the pixels across different samples would remain identical. Since we were using a pretrained Resnet-18, its prelearned internal features might have led to a bias of classifying based on these pixels.

In an attempt to test this and figure out whether our network actually falls into this bias or not, we manually went over our database and set aside one demonstrator to function as our test set. The one we chose shares its background color with almost none of the other videos, and appears the most in our dataset. We then repeated the process, while ensuring the number of samples per word is the same. The results for this run were 100% train accuracy and 0% test accuracy, which presents a (very) strong case for overfitting.

Second Attempt - Difference Frames

In light of the limited data we had available we had to find some way to make our backbone overcome these inherent biases. The idea we tried is to take a move from RNNs and put the emphasis on the movements themselves. Instead of feeding the frames themselves to the network, we used the differences between two consecutive frames - the effect of which is twofold; both eliminating static pixels like the background, and body parts which aren't supposed to move for a particular sign, as well as emphasizing what exactly was the movement that took place between two consecutive frames - filling in the gaps. Adjusting our samples like this and running the trials again we got 100% train

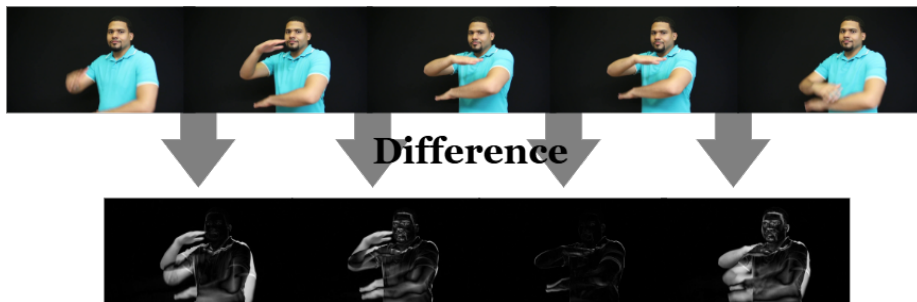


Figure 4: Using the differences between consecutive frames for the second round of trials

accuracy and 96.11% test accuracy for the classical train-test split, non isolated dataset, and 100% train accuracy, 0.53% test accuracy for the isolated dataset.

	Normal Frames	Difference Frames
Train-Test Split	99.29%	96.11%
Isolated Dataset	0%	0.53%

Table 1: Test accuracies across all four trials

Conclusion

Though we failed to successfully use Resnet-18 for transfer learning on translating words the WLASL database to written English, we can see in Table 1 that using the differences augmentation helped reduce overfitting and improve generalization ever so slightly, decreasing the test accuracy for the classical split trial, and increasing it for the isolated dataset. This shows the importance of using data-conscious augmentations, especially when working with limited data/resources.

Discussion

We’d like to acknowledge some of the shortcomings of our project in regards to the translation task itself; we ultimately failed to present a model which successfully classifies the videos of the signed words, and we’d like to suggest a few alternate directions we could have taken and methods we could have employed:

1. **Choice of Backbone:** ResNet-18 is a robust, 18 layer CNN which was designed to differentiate between many more different classes, which have distinctly different shapes and colors. Due to its relative complexity, and affinity to capturing geometric features/textures, it makes it very easy for ResNet-18 to overfit our relatively small and homogeneous dataset. A straightforward way to sidestep this would have been to use a simpler network or write our own, but perhaps freezing could have also helped us classify better - especially with limited data.
2. **Color Invariance:** If our suspicion that our model is biased towards detecting colors is correct, we could have applied more ‘aggressive’ color jittering, allowing more extreme hue shifts for example, and repeating a few times to generate more samples. Alternatively, we could randomize color jittering per frame of each sample instead of the current implementation which applies it to the entire tensor (sample). Also, we could use the same convolution kernel for each channel, effectively working in grayscale.
3. **Data:** We mentioned a few times the limitations of the WLASL database. Acquiring plentiful and more diverse data would have helped us to ‘outscale’ ResNet, reducing the chance of overfitting. Capturing our own data or combining different datasets are two options, but these unfortunately weren’t viable due to shortage of time and resources².
4. **Sequential Data:** The main point holding this project back, even if we were to solve the previous points, is its neglect of videos as a sequential data type. Video frames are not context-free, and by treating them as such we set ourselves for failure compared to sequential models which take advantage of this structure. The attempts we made at capturing this context, of training in difference frames, and the minor effect it had on overfitting and generalization, suggest that where we were to use a sequential model, or at least better direct our pretrained model by other

²even combining datasets would require checking for repeats somehow.

preprocessing steps which reveal information about the video as a time-dependent object, for example by feeding it a velocity field of the calculated approximate velocities of each pixel on screen, etc.

5. **Feature Extraction:** While inspecting our dataset, we picked on some ASL along the way, and with it we noticed that similarly to written language, words that are semantically close to one another may also be structurally similar. For example the signs for 'man' and 'woman' (Figure 5) share the same general hand movements, but one starts at the forehead, and the other at the chin. This gave us the idea that it might be beneficial to add a feature extraction step and to embed videos in some simplified lower dimension feature space based on these 'atoms', but that is beyond the scope of this project.

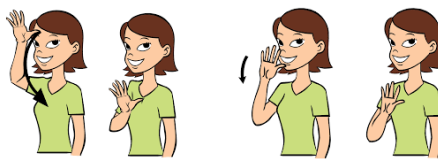


Figure 5: The ASL signs for 'man' (left) and 'woman' (right)

Future Directions

Provided we, or whatever party that may encounter this report, will want to complete and expand on it, here are a couple of suggestions for possible improvements/directions to take the project:

- Expand to sentence-level translation - decode videos of words signed one after the other
- Develop bidirectional translation (text-to-ASL) - accompanied by some graphics simulation to do the signings
- Improve processing efficiency for real-time applications
- Test with more diverse signing styles and backgrounds - being able to extend the model's capabilities would perhaps allow implementing it as a tool for people to use daily
- Other sign languages besides ASL - a lot of countries around the world have their own sign language, and it'd be great to get the model to be able to decode more of them

Closing Remarks

Even though our project was not what you'd traditionally call 'a success' we had a lot of fun, and learned a lot from our mistakes.

References

- [1] Nikhil Gupta. *Sign-Language-To-Text-Conversion*. <https://github.com/emnikhil/Sign-Language-To-Text-Conversion>. 2025.
- [2] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: [1512.03385](https://arxiv.org/abs/1512.03385) [cs.CV]. URL: <https://arxiv.org/abs/1512.03385>.
- [3] Mrigank Khandelwal. *ASL-2-Text*. <https://github.com/Mrigankkh/ASL-2-Text>. 2025.
- [4] Dongxu Li et al. “Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison”. In: *The IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 1459–1469.
- [5] Zed Sevcikova Sehyr et al. “The ASL-LEX 2.0 Project: A Database of Lexical and Phonological Properties for 2,723 Signs in American Sign Language”. In: *The Journal of Deaf Studies and Deaf Education* 26.2 (Feb. 2021), pp. 263–277. ISSN: 1081-4159. DOI: [10.1093/deafed/enaa038](https://doi.org/10.1093/deafed/enaa038). eprint: <https://academic.oup.com/jdsde/article-pdf/26/2/263/36643382/enaa038.pdf>. URL: <https://doi.org/10.1093/deafed/enaa038>.

Ethics Statement

1

Students: Omer Ziv & Raviv Segal

Project Name: ASL2Text

Description: We take the ResNet-18 architecture introduced in 2015's paper [2], and use it to classify signings from the WLASL database.

2

The following answers are by OpenAI's ChatGPT 5-mini:

a. List 3 types of stakeholders that will be affected by the project.

1. **Deaf and Hard-of-Hearing Users** – Individuals who will use the ASL2Text system to communicate more easily with hearing people.
2. **Researchers and Developers** – People working on improving sign language recognition, AI models, or related accessibility technologies.
3. **Educational Institutions / Schools for the Deaf** – Organizations that may integrate this technology into classrooms to assist in teaching or communication.

b. What will an explanation that is given to each stakeholder look like?

1. **Deaf and Hard-of-Hearing Users:** "Our ASL2Text system uses a neural network model to translate American Sign Language gestures captured on video into text in real time. This means you can communicate more easily with hearing individuals, as your signs are automatically converted into written words that can be read instantly."
2. **Researchers and Developers:** "We employ a ResNet-18-based convolutional neural network trained on the WLASL dataset to classify ASL signs from video frames. The architecture, training procedure, and dataset choice were optimized for accuracy and efficiency, providing a strong foundation for further research in gesture recognition and accessibility AI."
3. **Educational Institutions / Schools for the Deaf:** "ASL2Text converts students' sign language into text in real time, enabling more inclusive classroom interactions. Teachers can receive immediate written feedback, and students can better communicate in group activities or digital lessons."

c. Who is responsible for giving the explanation to each stakeholder?

1. **Deaf and Hard-of-Hearing Users:** Product team or user experience specialists should present the explanation through simple guides, demos, or tutorials, ensuring clarity and accessibility.
2. **Researchers and Developers:** Project leads or technical team members are responsible for detailed explanations, ideally through academic papers, presentations, or technical documentation.
3. **Educational Institutions / Schools for the Deaf:** Project managers or outreach coordinators should communicate with schools via demonstrations, workshops, or instructional materials that highlight practical classroom applications.

3

The answer we got seems reasonable and generic enough. We found it weird that though we didn't mention anything about real-time sign decoding (which in fact we don't do), ChatGPT decided to include that in its explanation for the less 'tech-savvy' stakeholders. That could be due to existing works/some other internal bias, but when using LLMs for this sort of input, perhaps it would be better for the models to ask for missing information.