Sequential Sign Language Recognition & Transcription

Aman Jalan (ajalan), Serena Wang (wserena), Yajwin Jain (yajwin), Hogan Mastanduno (hoganmas)

1. Introduction

Communication is essential to navigating through this world. Communication is especially challenging for those who are deaf or hard of hearing—a condition that affects about 28 million people in America and many more worldwide [1]. A common mode of communication used between deaf and hearing individuals is American Sign Language (ASL). Although sign language has become more well known, especially with more schools offering ASL courses, most people do not know ASL or are fluent in it. This makes it difficult for someone who relies on communicating in ASL as their primary mode of communication to communicate with others in their daily lives. Individuals with hearing disabilities also face challenges in getting an education and finding and maintaining a job. This makes the need for efficient and accessible sign language interpretation services even more of a priority. To help make communication easier for the deaf and hard of hearing community, we will explore and attempt to use computer vision with deep learning to recognize and transcribe ASL fingerspelling.

1.1. ASL Fingerspelling

Sign language is a visual language that uses gestures and hand shapes to represent concepts or ideas [1]. One end of the sign language spectrum is American Sign Language (ASL) which has its own syntax and grammar. One element of sign language is fingerspelling. Fingerspelling is where words are spelled out one at a time by handshapes. In ASL, fingerspelling is primarily used to indicate places, names, or ideas for which there is no official sign [2].

1.2. ASL and Computer Vision

Technology has helped mitigate some of the challenges that deaf and hard of hearing people face with communication. The internet and electronic devices like smartphones have made it possible so that speaking and hearing are no longer required to do some everyday tasks. Despite this, there is still a strong need for interpreters to help hearing impaired people that rely on sign language to converse with others who do not know sign language. Hiring an interpreter is often very expensive and unaffordable for most people. Fortunately, the advancements in computer vision has made accessible, computer driven ASL interpreting in real time possible. The computer vision community has made lots of developments in image recognition and classification, making it possible for models to be trained to interpret and understand sequential gestures. We will use these computer vision techniques to create and train our own model for interpreting ASL fingerspelling.

1.3. Related Works

Currently, there are a few ASL interpreting apps out on the market. One of which includes Google's GnoSys application that translates hand gestures into text or speech using computer vision and deep learning [3]. Unfortunately, there are still a lot of improvements left to be made with these ASL interpreting apps. There are still issues with accuracy and reliability with the translations and the apps are often not fast enough to keep up with a conversation in real time. With GnoSys, only hand gestures are detected [3]. It misses any facial expressions or speeds of signing that can change the meaning of what is being discussed [3]. We are seeing just the first phases of computer vision driven ASL interpretation.

Some neural networks that have been used for image and video recognition include CNN, RNN, and ResNet. We have investigated these different neural networks and evaluated its usefulness in gesture recognition. We will explore them further in our method approach.

1.4. Method

In an effort to make communication in the real world easier for the deaf and hard of hearing community, we will build a model that uses sequential sign language gestures to accurately predict their translations. Each sign language gesture represents an alphabet, and a sequence of gestures represents a word. When we try predicting words from single gesture models, we lose a lot of accuracy because many gestures are very similar in sign language. To get a more accurate prediction on sign language translations, we will be looking at sequences of gestures, as that gives us extra information such as co-articulation and spontaneous sign production.

2. Approach

In this experiment, we want to determine whether a 3D ResNet model is efficient at sign language (SL) recognition. To determine whether it is efficient, we will compare it to two baselines:

1. A baseline convolutional neural network (CNN)
2. Completely random selection.

2.1. ResNet 3D

A ResNet 3D model is a neural network that uses residual functions, and 3D convolutions to train its parameters to perform the task of classifying instances. This project utilizes the R3D_18 torch vision model which adopts the architecture of a 3D ResNet 18 model. Usually by adding more layers to a network in trying to make it more performant, the accuracy of a neural network model seems to fall. There are many research papers that highlight this idea. To resolve this problem, residual networks are used. Residual networks use skipped connections to skip certain layers that are detrimental to the performance of the network. They are functions that formulate the layers of a neural network by having a reference to the input through these connections, as shown in Figure 1 below. These are the building blocks to the Resnet model that we use for our video classification.

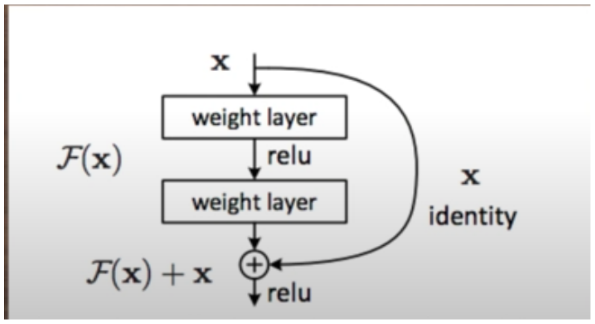
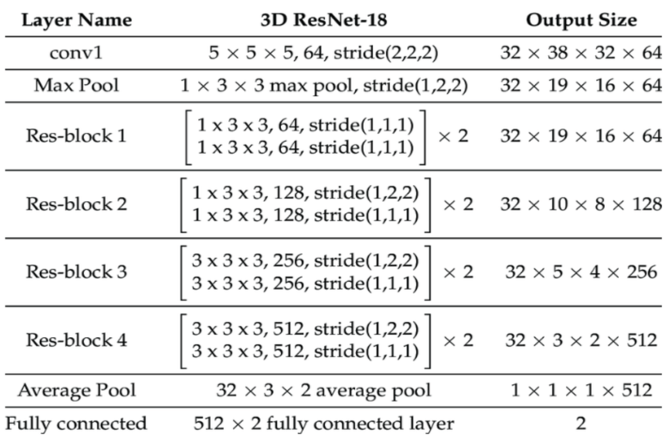

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 1: Skipped connections

In addition to this, R3D_18 uses 3D convolution to deal with the spatial and temporal dimensions for our dataset. Since we are classifying ASL videos, the model takes input in an extra temporal dimension and performs 3D convolution. This represents the number of frames for our videos. The details of this are explained in the baseline section of this report.

The architecture for the R3D_18 model is complex and it uses many layers. The first few layers include the convolution and Max Pool layer. Following that are different Res-blocks that use a combination of Convolution 3D layers, Batch Normalization 3D layers, and ReLU layers. The model has an average pooling layer, and a fully connected layer as the last few layers of the network. The input to our model is a dataset that has 8 frames, 3 in-channels, and 64x64 image dimensions. In figure 2, we can see these layers and the dimensions with an input size of (5,5,5,64) instead.



Layer Name	3D ResNet-18	Output Size
conv1	$5 \times 5 \times 5$, 64, stride(2,2,2)	$32 \times 38 \times 32 \times 64$
Max Pool	$1 \times 3 \times 3$ max pool, stride(1,2,2)	$32 \times 19 \times 16 \times 64$
Res-block 1	$\begin{bmatrix} 1 \times 3 \times 3, 64, \text{stride}(1,1,1) \\ 1 \times 3 \times 3, 64, \text{stride}(1,1,1) \end{bmatrix} \times 2$	$32 \times 19 \times 16 \times 64$
Res-block 2	$\begin{bmatrix} 1 \times 3 \times 3, 128, \text{stride}(1,2,2) \\ 1 \times 3 \times 3, 128, \text{stride}(1,1,1) \end{bmatrix} \times 2$	$32 \times 10 \times 8 \times 128$
Res-block 3	$\begin{bmatrix} 3 \times 3 \times 3, 256, \text{stride}(1,2,2) \\ 3 \times 3 \times 3, 256, \text{stride}(1,1,1) \end{bmatrix} \times 2$	$32 \times 5 \times 4 \times 256$
Res-block 4	$\begin{bmatrix} 3 \times 3 \times 3, 512, \text{stride}(1,2,2) \\ 3 \times 3 \times 3, 512, \text{stride}(1,1,1) \end{bmatrix} \times 2$	$32 \times 3 \times 2 \times 512$
Average Pool	$32 \times 3 \times 2$ average pool	$1 \times 1 \times 1 \times 512$
Fully connected	512×2 fully connected layer	2

Figure 2: Internal structure of R3D_18

2.2. ChicagoFSWild Dataset

The ChicagoFSWild dataset consists of 7304 videos of fingerspelling sequences along with their corresponding transcriptions. This dataset originates from the team of researchers at the University of Chicago and the Toyota Technological Institute at Chicago, who were also responsible for translating each sequence.

In order to use the entire dataset, we created a custom data structure to act as an intermediary between the neural networks and the ChicagoFSWild files.

This data structure was optimized to minimize the workload on the disk, by reducing the number of files and directories that were read by the program. This involved reducing the spatial resolution to 64x64 pixels and 8 frames. We chose this optimization metric in response to the massive size of the dataset (~14 GB); our progress early on was hampered by absurdly long disk reads.

2.3. Data Processing

Theoretically, our best model would be one that classifies a video or sequence of images into 1 of 26 categories, each corresponding to a single English letter. We would run this model on a set of image sequences, each depicting the fingerspelling of a single letter.

However, the ChicagoFSWild dataset consists of sequences that almost entirely depict a string of letters, making this approach difficult. Furthermore, the RNNs we used were built to classify each sequence into a single category, not multiple. Our workaround to this was to instead use our models as word classifiers.

Another issue is that the ChicagoFSWild consists of 3906 distinct labels, many of which had only a single sequence in the entire dataset. On average there are roughly two sequences per label in this dataset, meaning that training a word classifier on this set would be exceptionally prone to *overfitting*. (Overfitting occurs when a model is fitted for a dataset that is not large enough. Overfitted models tend to fail at predicting interpolations.)

To avoid overfitting our data, we decreased the number of unique labels in the dataset down to the 10 most common labels in the entire dataset, cutting out all sequences that do not correspond to the 10 most common labels. This had the added benefit of reducing the number of datapoints from 7304 to ~1050. This reduction in the size of our working dataset allowed for the process of training our models to be much more efficient. This ultimately let us run the training process in under one hour.

Of the roughly 1050 sequences that we filtered out from the original ChicagoFSWild dataset, we divided this subset into an 80:20 train-to-test split, yielding 210 test sequences. We took advantage of the fact that the original dataset had existing subdivisions for training and testing. We assured that these original subdivisions featured near-equal distributions of labels.

2.4. Baselines

We chose our two baselines in order to answer two key questions concerning the use of RNNs on SL recognition:

1. Does a RNN model work for SL recognition?
2. How well does a RNN work for SL recognition when compared to traditional CNNs?

To address the former, we will consider a model that simply guesses randomly. If our RNN model works for SL recognition, then it should make predictions that are significantly more accurate than mere random guesses.

To address the latter, we will use a simple CNN. This CNN is composed of a single 3D convolutional layer, followed by a single linear layer. If our RNN model works better than CNNs for SL recognition, then it should make predictions significantly more accurate than those of even the simplest CNN.

The baseline CNN uses 3D convolution. An example of that is shown in figure 3 below. A convolution is performed for a stack of frames to form a 3D cube. This is then changed into a linear layer to achieve our classification results.

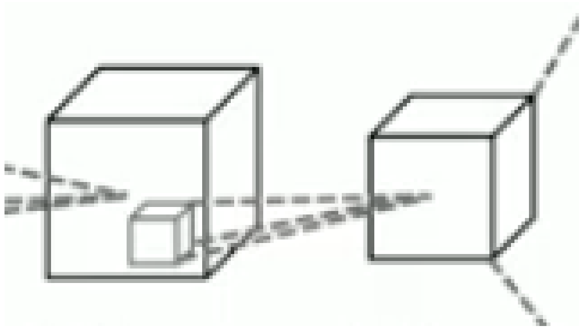


Figure 3: 3D convolution

2.5. Training Our Models

We used an Adam optimizer to train both our baseline and ResNet models. These optimizers used the same hyperparameters:

- learning rate = 10^{-3}
- weight decay = 10^{-4}
- training batch size = 16
- number of epochs = 10

Lastly, note that due to our limited computational resources, we used external servers to train our models, specifically using Google Colab and Project Jupyter: two extremely potent web-based computing platforms.

3. Experiments

Our main metric of success was the accuracy of our results, as our main goal was to maximize the correct number of predictions for our problem statement. Thus, we focused on quantitative evaluation more than qualitative evaluation.

3.1. Data Analysis

When we ran our model against the test set, below were the accuracies that we achieved. We also measured the time it took to train our models, so that we could evaluate and compare the complexity between the two. As seen below, the training time for the ResNet model was dramatically larger than our baseline model, indicative of the added complexity of the multiple layers of the ResNet model. This data can be seen in Figure 4.

Metric	Random Selection	Baseline Model	ResNet Model
Accuracy (%)	10	9.56	13.97
Training Time (s)	-	316	8512
P-value	-	0.58	*0.028*

Figure 4

Our ResNet model performed much better than our baseline random selection, to a *statistically significant* degree. Furthermore, the baseline model oddly performed worse than random selection.

Also note that the baseline model trained much more quickly than the ResNet model by a factor of 27.

3.2. Discussion

Why did the baseline model perform worse than random selection?

Our best explanation is that the baseline model does not efficiently account for the added temporal dimension in the way that a ResNet does. Although the baseline consists of mostly 3D convolutions, 3D convolutions rely on temporal locality: that is, that information is shared across adjacent frames in a video. ResNets do not suffer from this aspect to the same degree, as they utilize skipped connections to relate information between non-adjacent frames in a video. This highlights a key benefit to using a ResNet for video recognition and by extent most SL recognition.

Why did the ResNet model not perform better?

Although the ResNet model did perform statistically significantly better than random selection, its accuracy was not high enough to be applicable for many real-life situations, which often require accuracies greater than at least 14%.

Here we can note the low resolution of our dataset. Given our available computational resources, in order for our training process to run in a reasonable amount of time we were forced to drastically reduce both the spatial and temporal resolution of the fingerspelling sequences. This caused a loss in information and thus demolish the accuracy of both models. Note that this low resolution can be partially blamed on the original ChicagoFSWild dataset, which featured several sequences consisting of 2-3 frames per letter.

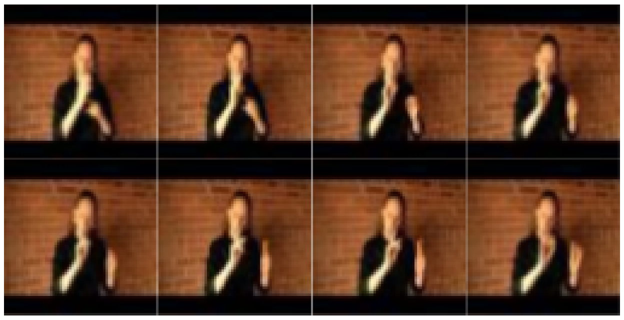


Figure 5: Example fingerspelling sequence

As a demonstration, Figure 5 showcases a single sequence, spelling the word “ok”. Observe how much fine detail in the hand formations is lost in the process of downsampling each image to 64x64 pixels. However, note that there is enough visual information there to detect the hand movement across the 8 frames, which appears to have been the essential piece for the statistically significant performance improvement seen in our ResNet model.

3.3. Conclusion

In spite of our limited resources forcing us to simplify our procedure and dataset, we nonetheless observed a clear performance increase when using the ResNet model. It appears to have a clear advantage in video classification over a 3D convolutional neural network (CNN). Due to the prevalence of video classification in the field of SL recognition, we can conclude that overall ResNet models both work efficiently for SL recognition and also provide an advantage over CNNs in the case of video classification.

4. Implementation

The following external resources were used

1. ResNet 3D model (credits to torchvision Python library)
2. ChicagoFSWild dataset [4] (credits to creators at TTI-Chicago and U. Chicago)

References

[1] Jay, Michelle. “American Sign Language: START ASL.” *Start ASL | Learn American Sign Language with Our Complete 3-Level Course!*, Michelle Jay Https://Www.startasl.com/Wp-Content/Uploads/StartASLlogoFinal-1.Png, 15 Feb. 2021,

https://www.startasl.com/american-sign-language/.

[2] ASL, Start. “Fingerspelling in American Sign Language: START ASL.” *Start ASL | Learn American Sign Language with Our Complete 3-Level Course!*, Start ASL Https://Www.startasl.com/Wp-Content/Uploads/StartASLlogoFinal-1.Png, 15 Feb. 2021, https://www.startasl.com/fingerspelling.

[3] “Google Sign Language AI Turns Hand Gestures into Speech.” *BBC News*, BBC, 20 Aug. 2019, https://www.bbc.com/news/technology-49410945.

[4] “American Sign Language Fingerspelling Recognition in the Wild.” *Chicago Fingerspelling in the Wild Data Sets (Chicagofswild, ChicagoFSWild+)*, Dec. 2018, https://home.ttic.edu/~klivescu/ChicagoFSWild.htm.

[5] "ResNet-18 Architecture. Network Architecture for 3D ResNet-18 Model..." *ResearchGate*, 1 Dec. 2021, www.researchgate.net/figure/ResNet-18-architecture-Network-architecture-for-3D-ResNet-18-model-used-for-corr-fMRI_tbl1_342127487.