CS230

Real-time ASL to English text translation

Neha Keshari
Department of Computer Science
Stanford University
nehakesh@stanford.edu

Anil Kumar Srikantham
Department of Computer Science
Stanford University
asrik@stanford.edu

Gerardo Gomez Martinez
Department of Computer Science
Stanford University
ggomezm@stanford.edu

Abstract

With the recent progress on applying Deep Learning to natural language processing and machine translation a portion of the population, those who are deaf, has not been able to benefit from some of these advancements. Here we propose a system that will enable this portion of the population to benefit from these advancements. For our system, transfer learning is used to accelerate training and improve results. Building on pre-trained models we were able to obtain a 25% accuracy for translating ASL to English.

1 Introduction

ASL is one of the most popular sign languages around the world. It is used by around 500,000 individuals in the US and Canada. Very often most of the users, who are deaf and use sign language as a means of communication, feel left out in a new environment because of the lack of common communication medium. Sign language is a different language in that it can't be spoken. If it can't be spoken, then it can't be translated by any current language translation software. Hence we propose ASL to English text automated translator which can help ASL users communicate in scenarios like when they pay a visit to a doctor or participate in a conference.

There are existing models, including deep learning models that can convert ASL to English text. But the efficacy of such models in real life is limited because the training data is usually on images that cannot be scaled to real time scenarios.

2 Related Work

This project was inspired by the dataset and methodology suggested in the MS-ASL: A Large-Scale Data Set and Benchmark for Understanding American Sign Language paper by Microsoft. We make use of the dataset provided there with our own pre-processing to adapt it to our use. We use the model and evaluation metrics suggested in the paper.

CS230: Deep Learning, Winter 2020, Stanford University, CA.

3 Dataset

We used the following dataset from Microsoft research.
`https://www.microsoft.com/en-us/research/project/ms-asl/downloads`

The dataset includes around 1000 classes covering around 25000 videos for train, validation and test datasets. As suggested in the MS-ASL paper , we plan to use per class accuracy on the test dataset as the benchmark.

Most of the videos have different signers which presented a challenge in terms of training the data. Our model required us to capture ASL from different signers. This later translates to better real world performance as its not fitted to a specific signer.

We have uniformly separated our data to ensure that the training, validation and test data sets come from similar distributions. Since our dataset consists of videos with different signers and different FPS values we have used this as a way to profile our dataset. Figure 1 shows the distributions in respect to the different FPS values and the different signers in the video

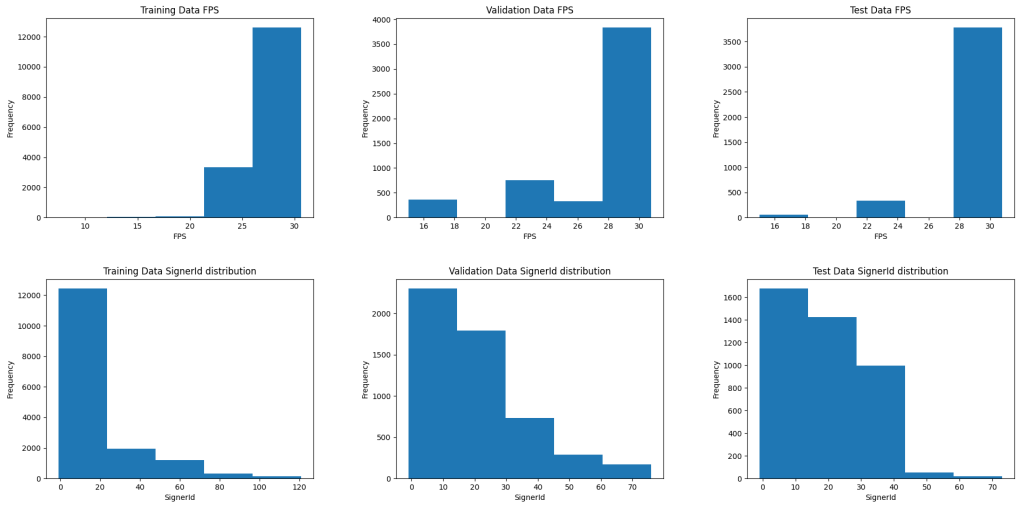

Figure 1: Charts showing FPS and Signer ID of data used for training, validation, and test sets

4 Preprocessing

To train the model, we are using MS-ASL dataset. The dataset is essentially a collection of youtube links to ASL videos. There is a total of 6452 distinct youtube videos. Each video can in turn have multiple ASL words. The MS-ASL dataset had the start and end times and bounding box dimensions for a word against each video file.

We took various preprocessing steps to get the right video for a sign. The dataset had videos with text for the signal or additional unwanted space. For this we used a bounding box already present in the dataset, to crop the videos. There were many videos with signs for more than one text, and required to be trimmed by start and end time and separated for different signs. For this, we used the start and end times provided in the dataset to be considered for each sign in the video. Based on these we trimmed the videos as needed. We also converted these videos to frames in order to input them into our network. We have used the image resolution of (224, 224, 3). Figure 2 shows the data processing pipeline. Also, we did some data augmentation by shifting the width and height of the RGB images by 0.2 scale. This was done to improve the validation accuracy. We have followed the same train, validation and test sets distribution as we got from the MS-ASL dataset, which is 63% train/21% validation/16% test . We have performed our analysis on a dataset with 200 classes.

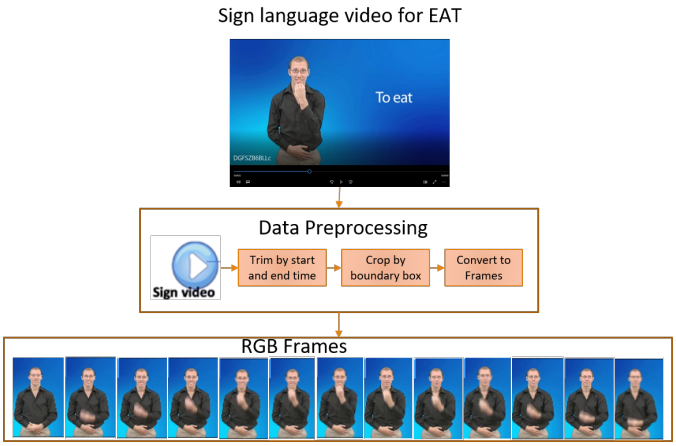

Figure 2: Chart showing data processing pipeline

As a preprocessing step, we downloaded all the videos and next applied trimming and cropping techniques using a ffmpeg library. After preprocessing below is the distribution of the video data

Set	Class Count	Video Count
Train	1047	10872
Validation	1017	3689
Test	1027	3560

Table 1: Dataset distribution after preprocessing

5 Methods

For the model, we are using the I3D RGB Inception architecture model that the MS-ASL paper suggested. We used pretrained weights for the model. Our metric of evaluation during the training process was the validation dataset accuracy.

With this approach, we believe we can make the ASL translation more adaptable and thereby have an increased participation from ASL users with a real-world use case.

5.1 I3D Model

Using video data adds an extra dimension when compared to just image data, this dimension also has a temporal element associated with it. The I3D model is especially helpful for this sort of situation. It takes elements such as filters and pooling layers from 2D model architectures and adds an additional dimension to them. This allows using popular 2D model architectures in a 3D space.

We used the pretrained I3D Model training on 100 classes to start and later expanded to 200 classes. We have used the pretrained weights from the I3D models in order to achieve higher accuracy with our limited training data and have frozen all layers in the model except the last layer or 50 layers where we have added our own layers to train on our data.

5.2 Experiments

We used the pre-trained I3D model trained on Kinetics dataset and ImageNet. The pre-trained model considers RGB channels to get spatial representations and has 185 layers. We replaced the last layer with a Conv3D and softmax activation that can be configured to output the required number of classes. Our metric of evaluation is validation accuracy. In all of our experiments, we trained the model for 50 epochs

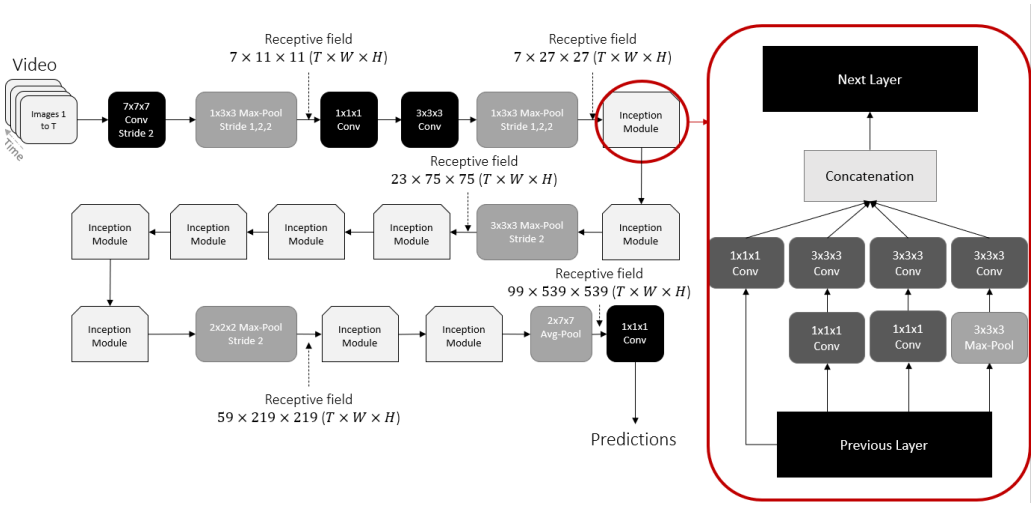

Figure 3: I3D model architecture

5.2.1 Without Preprocessing

First, we trained the model without any cropping. The model did overfit to the training set with a high train accuracy and very low validation accuracy. Figure 4 shows the results after using 0.5 for dropout as a regularization.

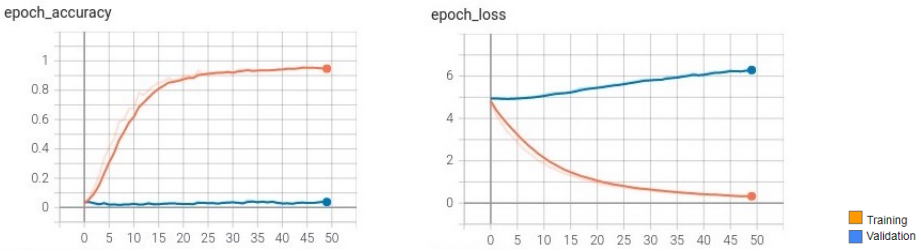

Figure 4: Accuracy on raw data

5.2.2 Fine-Tuning Hyperparameters

As a second iteration, we trained the model by changing parameters for Adam optimizer with a learning rate of 0.0001 with a decay of 1e-6. This improved the validation accuracy

5.2.3 Cropping Video

In the next iteration, we cropped the video to the bounding box dimensions as specified in the dataset and trained the model. The gave a good boost to the whole validation accuracy. The validation accuracy reached double digits of 0.14.

5.2.4 Fine Tuning Last Layers

Next instead of just training the model for weights on the last or the topmost layer, we froze the first 135 layers and trained on the last 50 layers. This improved the validation accuracy to 0.19.

5.2.5 Data Augmentation

As one last step, we tried augmenting the data by randomly shifting the frame by 20% on width and height of each frame of the video. With this step, the validation accuracy improved to 0.25.

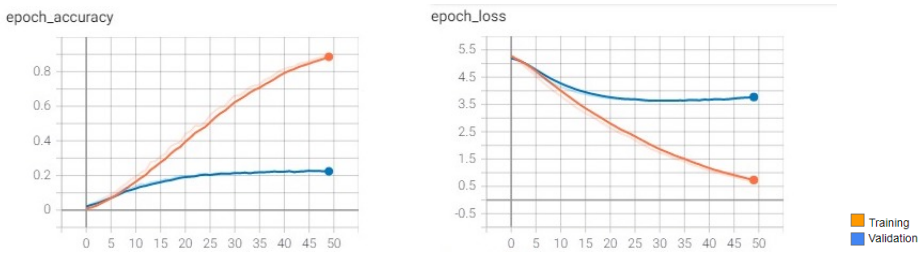

Figure 5: Accuracy With data augmentation

Iteration	Train 100 Acc	Val 100 Acc	Train 200 Acc	Val 200 Acc
No trimming and cropping	0.98	0.02	NA	NA
Trimming and cropping	0.99	0.12	NA	NA
Changing parameters for Adam	0.95	0.14	0.95	0.14
Training last 50 layers	0.95	0.19	0.99	0.18
Data Augmentation	0.96	0.25	0.9	0.22

Table 2: Accuracy results from different iterations

6 Conclusion and Future Work

I3D model is a good start for working on action based video datasets. We were able to iterate with different approaches and achieved very good training accuracy. We did iterations to improve on validation accuracy. The best approach, given the high training accuracy, is to apply data augmentation techniques. We tried random shifts as augmentation and it improved the validation accuracy. By working on this project, we were able to learn the complexities of processing video data for training deep neural networks. Training such a large dataset is computationally expensive and a real challenge.

We would also like to enhance the model further by doing the following:

1. Train for all 1000 classes using a bigger machine considering 64 frames.
2. Try other data augmentation techniques like a random crop to improve on regularization of the model.
3. We would like to use MediaPipe in order to enhance the real time translation of ASL to English.
4. We would like to convert the model into ONNX format which offers interoperability. `https://onnx.ai/`
5. Since we have ASL translated to English, for further translation natural language processing models can be used to translate to different spoken languages. Since natural language processing models are large and need large infrastructure the challenge is to get this working on a mobile platform, in order to solve this, we would like to use TinyBERT model for a real time scenario.

7 Contributions

We worked together in evaluating current similar approaches for ASL to English translation using Deep Learning and finding a model that would work for our use case. Neha focused primarily on data pre-processing needed before being able to feed into the model. Anil focused on training the initial model that we used as a baseline model. Gerardo focused on logistics and aggregating the obtained data into this report. Once we had the initial base model we worked together to optimize it.

8 Code

The code for our model and pre-processing scripts can be downloaded from: `https://github.com/gerardodekay/Real-time-ASL-to-English-text-translation/`

References

[1] “EfficientNet-Lite.” GitHub, Tensorflow, github.com/tensorflow/tpu/blob/master/models/official/efficientnet/lite/README.md.

[2] “MediaPipe Hands.” MediaPipe, Google, google.github.io/mediapipe/solutions/hands.

[3] “Open Neural Network Exchange.” ONNX, onnx.ai/.

[4] "Translating Sign Language in Real Time With AI" towardsdatascience.com/using-ai-to-translate-sign-language-in-real-time

[5] “MS-ASL American Sign Language Dataset.” Microsoft Research, 5 Aug. 2019, www.microsoft.com/en-us/research/project/ms-asl/downloads.

[6] "i3d keras" https://github.com/OanaIgnat/i3d_keras

[7] "Training a neural network with an image sequence" https://medium.com/smileinnovation/training-neural-network-with-image-sequence-an-example-with-video-as-input-c3407f7a0b0f

[8] "Understanding the Backbone of Video Classification: The I3D Architecture" https://towardsdatascience.com/understanding-the-backbone-of-video-classification-the-i3d-architecture-d4011391692

[9] "TinyBERT: Distilling BERT for Natural Language Understanding" https://arxiv.org/abs/1909.10351

[9] "Understanding the Backbone of Video Classification: The I3D Architecture" https://towardsdatascience.com/understanding-the-backbone-of-video-classification-the-i3d-architecture-d4011391692

[10] "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset" https://arxiv.org/pdf/1705.07750.pdf