

# CS 230 Project Final Writeup

## ASL Optical Flow

Diego Celis, Ella Hofmann-Coyle

dcelis [at] stanford [dot] edu

ellahofm [at] stanford [dot] edu

### Abstract

In society there is a discrepancy between the number of people who do not understand ASL and the number of people who use ASL. To bridge this gap, we have created a pipeline that inputs ASL signage and outputs the corresponding English translation. Specifically, we employed a 2D CNN to process raw RGB images of ASL participants signing, as well as a 3D CNN to process stacked optical flow frames of the aforementioned. In our results, we saw that using RGB images resulted in more accurate results than using optical flow frames to predict ASL alphabet signs.

### Introduction

Today there are countless language translators for nearly all spoken languages. However, no such translator exists to bridge the language barrier between those who sign American Sign Language (ASL) and those who speak English, leaving both parties at a disadvantage. We hope to bridge this language barrier by creating a translator. Our system takes in video input of a person signing and uses a convolutional neural network to output the sign used. In another case we also generate optical flow data from the images prior to running the convolutional neural network.

### Related Work

The majority of the papers all employ CNN's to capture body movements. One exception is the paper by Cao et. al. for OpenPose, which uses part affinity fields to learn to associate body parts with individuals in the image [4]. The other exception is the paper by Farnebäck, which describes the mathematics behind the optical flow we employed in our project [5].

To solve the action recognition issue, we first wanted to use the OpenPose software based on "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields" [4]. This approach creates vectors to recognize hand and body poses by generating confidence maps and affinity fields and then bipartite matching them to associated body parts. OpenPose's state-of-the-art approach is similar to that reflected by our own since both rely on looking at pixel changes to make inferences, but differ in how we act on that data. In the OpenPose approach, this information was used to further build onto a more complex system to calculate part affinity fields, whereas in our approach we tried extracting features from the optical flow frames using CNN's.

When we were unable to install the software provided (after numerous weeks, and trying several operating systems and computers), we pivoted to solve the challenge of action recognition ourselves. To initially approach this problem, we read "Two-Stream Convolutional Networks for Action Recognition in Videos," which achieved some of the best results in this area [1]. In their approach they created two ConvNets, one trained on video frames and the other on stacked

temporal flow inputs of the video, and finally fused the two outputs to generate a score. For our optical flow frame approach, we took inspiration from their temporal flow ConvNet in particular, from which we formed our own. However, we decided to not create a spatial flow ConvNet to work alongside it. Instead, we took inspiration from the architecture by creating our own spatial flow ConvNet to work on its own. One of the setbacks from this particular strategy lies on the fact that running two streams is computationally expensive, which would not translate well to forming a real-time ASL translation solution, which would ideally serve as the end-goal for further development of our project.

We also read “Learning Spatiotemporal Features with 3D Convolutional Networks” which created a 3D ConvNet architecture that takes stacked unprocessed video frames as input [3]. While this approach does retain important spatio-temporal information from the input structure, it does not utilize sequential optical flow data. “Convolutional Two-Stream Network Fusion for Video Action Recognition” used a similar 2-stream approach that added multiple points to fuse the temporal and spatial scores [2]. The paper expands further on alternate fusion methods. While we did not use a spatial ConvNet, we took inspiration primarily from the first paper in developing the 3D architecture by modifying their approach to the temporal stream. We also took inspiration from the paper by Tran et. al. in how we structured our input data [3].

There have been attempts at tackling this problem in the past. In particular, we took interest in “A Web-Based Sign Language Translator Using 3D Video Processing,” which took a different approach [6]. The largest difference between our approach and that used in the paper by Li et. al. is that their team relied on a 3D sensing camera, like a Kinect, to understand motion. In contrast, our system aims at being successful using a regular camera that captures 2D images. This presents a large difference in potential—their approach requires a hefty price in creating a system that recognizes ASL whereas ours can use any camera with enough detail.

## **Dataset and Features**

We are currently working with the RVL-SLLL ASL database from Purdue University, which is comprised of 14 different participants signing a collection of messages that span a portion of ASL. More specifically, the database is broken down into five sections:

1. alphabet,
2. numbers 1-20,
3. handshapes with two examples each,
4. isolated signs to show motions, and
5. paragraphs to show connected discourse.



*Samples frame from data set*

*(We can not release samples as per our confidentiality agreement with Purdue)*

In total, the database contains 1,950 items, all of which total a size of 274.19GB. Holistically, the database encompasses signage that ranges from simple to complex use cases. In addition, some portions of the database offer contrast and diffused subject lighting as options. The data is mostly labeled and is presented colorized. The frames are  $640 \times 480$ p. Please refer to the references section of this document for a proper citation to the database.

In one approach we apply layered optical flow images: 3 dimensional input - the two image dimensions, a dimension for stacking the frames, and the image channels used to represent angle and magnitude generated through optical flow.

In another we applied a CNN to several raw RGB frames per sign from the segmented data set. We saw this as appropriate as our data was good at localizing the movement of the signage, meaning that a CNN and optical flow would both derive good results from such a setup.

For both approaches, we mainly focused on an alphabet comprised of 25 different frames per letter, which yields a total of 650 frames. There was no data augmentation, and the only preprocessing that we implemented was for the optical flow 3D CNN, in which we calculated optical flow frames to pass into our model. We split our data with 70% encompassing our training set and 30% encompassing our testing set. We thought this division point would be appropriate given the size of the dataset we could parse from the database.

For the 3D CNN, however, because the preprocessing and the 3D convolutions were too computationally expensive, we were only able to use a smaller fraction of the data to run tests. For this approach, we decided to split our data subset with 61% encompassing our training set and 39% encompassing our testing set in order to account for this difference in size.

In both approaches we did not use a development set and we normalized the input data by dividing the matrix images by their highest possible value.

## **Methods**

In our first attempt at this problem we fed in raw RGB images into a 3-layer CNN using 2D convolutions. For this approach, we attempted to minimize the cost via softmax cross-entropy. We implemented this approach using tensorflow.

Our architecture for the approach is as follows:

*CONV2D (s=1) → RELU → MAXPOOL (8x8 s=8) → CONV2D (s=1) → RELU → MAXPOOL (4x4 s=4) → FLATTENED FULLY CONNECTED*

In our other attempt we stacked 5 optical flow images generated from 5 frames of a signing clip as input to a 3D ConvNet architecture. We generated our optical flow images using openCV's implementation of Farneback's algorithm for optical flow. The optical flow is represented similarly to how an image is represented, however instead of three channels there are two, one that represents the angle of motion, the second representing the vector's magnitude. Our input shape to our model is 4-dimensional:

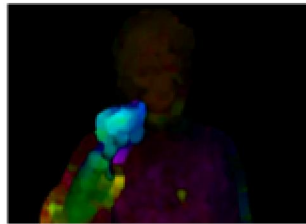
*(5, 480, 640, 2)*

*(# of stacked frames, frame axis=0, frame axis=1, optical flow channels)*

These frames were selected through a filtering process to reduce motion noise. To construct this model, we used keras as our framework and attempted to minimize categorical cross-entropy.

The architecture is as follows:

*CONV3D (f=40 3X3X3 s=2 norm) → RELU → MAXPOOL3D (2x2x2 s=2) → CONV3D (f=2 1x1x1 s=2 norm) → MAXPOOL3D (f=2 1x1x1 s=2, 1x2x2 s=2) → FLATTENED FULLY CONNECTED → FLATTENED DENSE*



*Examples of visualized optical flow data*

In both models, our softmax output classified the 26 characters of the alphabet.

## **Experiments/Results/Discussion**

After requesting and receiving access to the database, the first step we took was creating a program `video_downloader.py` to download the entirety of the database onto our machines. Please note that in order to respect the database usage agreement, the username and password provided to us for database access have been redacted from `video_downloader.py`, and as a result the program will not run.

We then entertained OpenPose, a real-time multi-person system to jointly detect human body, hand, and facial keypoints. From there, we created a model to test the utility of the database in a “simple” case.

Our project took a very different form than what we had initially had hoped. When scoping this problem, we planned on using OpenPose, a software from CMU, that vectorizes hand positions and then feeding this into a network. We spent 4 weeks attempting to install the openPose software,

trying installation on partitions of different kinds of Linux operating systems, Windows 10. We also tried installation on two different computers. Unfortunately we were not able to install the software, and thus took on the challenge of action recognition, a much more challenging problem, but a great learning opportunity.

Taking inspiration from “Two-Stream Convolutional Networks for Action Recognition in Videos” [1], we searched for an open-source pre-trained two-stream convolutional neural network. We found <https://github.com/feichtenhofer/twostreamfusion> (a two stream model developed in MATLAB) and made great progress in installing all the required system requirements, but in the end could not use the model due the fact that the program required to generate the input to the temporal network could not run on our operating system.

We then turned to building our own architecture to solve the action recognition problem. We based our inspiration for looking at optical flow images from the two-stream CNN model “Two-Stream Convolutional Networks for Action Recognition in Videos” [1]. We added a new dimension to the our input to our CNN, whereas the paper increased the size of the image channels. The poor results from running a 3D CNN were likely due to difficulties from optical flow in deciphering hand movement. This is caused by finger positioning (covering other fingers) and granularity issues. Our model would have also likely benefited from transfer learning from the twostreamfusion pretrained model.

As we invested most of this quarter into getting OpenPose and then a pre-trained model to work, we did not have as much time as we would have liked to fine tune our models. For the RGB model we used a learning rate of .009.

Here are our results.

<u>model</u>	<u>training accuracy</u>	<u>testing accuracy</u>
CNN RGB	.991	.875
3D NN	.181	.143

Despite not using OpenPose or a pretrained model, we received favorable results with the 2D CNN RGB. For the 3D CNN, our efforts were largely hindered by lack of computational power alongside architectural challenges.

## **Conclusion/Future Work**

In our project we used two approaches to solve the action recognition problem. We built two models, a 3D optical flow model and a 2D RGB model. Our 3D optical flow model saw poor results likely due to lack of granularity required to distinguish rotated poses of hand signs from optical flow data. Our RGB model, however, did very well in classifying signs. However, this model struggles to classify dynamic signs (letters ‘J’ and ‘Z’). From these results it is clear that the temporal data is

essential in building a robust model. Our results would greatly be improved, by using either a 2-stream CNN with scores fused from a temporal and spatial stream, or the OpenPose vectorization.

## Contributions

We have worked in conjunction to design our infrastructure and execute it. We worked together on the installation process of openPose and the pre-trained models, the neural network design and execution of the models.

## References

### [1] SIMONYAN, K. AND ZISSERMAN, A.

Two-Stream Convolutional Networks for Action Recognition in Videos

Simonyan, K. and Zisserman, A. (2014). *Two-Stream Convolutional Networks for Action Recognition in Videos*. [online] Arxiv.org. Available at: <https://arxiv.org/abs/1406.2199>.

### [2] FEICHTENHOFER, C., PINZ, A. AND ZISSERMAN, A.

Convolutional Two-Stream Network Fusion for Video Action Recognition

Feichtenhofer, C., Pinz, A. and Zisserman, A. (2016). *Convolutional Two-Stream Network Fusion for Video Action Recognition*. [online] Arxiv.org. Available at: <https://arxiv.org/abs/1604.06573>.

### [3] TRAN, D., BOURDEV, L., FERGUS, R., TORRESANI, L. AND PALURI, M.

Learning Spatiotemporal Features with 3D Convolutional Networks

Tran, D., Bourdev, L., Fergus, R., Torresani, L. and Paluri, M. (2015). Learning Spatiotemporal Features with 3D Convolutional Networks. *2015 IEEE International Conference on Computer Vision (ICCV)*. [online] Available at: <https://arxiv.org/abs/1412.0767>.

### [4] CAO, Z., SIMON, T., WEI, S. AND SHEIKH, Y.

Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields

Cao, Z., Simon, T., Wei, S. and Sheikh, Y. (2017). Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [online] Available at: <https://arxiv.org/abs/1611.08050>.

### [5] FARNEBÄCK, G.

Two-Frame Motion Estimation Based on Polynomial Expansion

Farneback, G. (2003). Two-Frame Motion Estimation Based on Polynomial Expansion. *Image Analysis*, [online] pp.363-370. Available at: <http://www.diva-portal.org/smash/get/diva2:273847/FULLTEXT01.pdf>.

### [6] LI, K. F., LOTHROP, K., GILL, E. AND LAU, S.

A Web-Based Sign Language Translator Using 3D Video Processing

Li, K., Lothrop, K., Gill, E. and Lau, S. (2011). A Web-Based Sign Language Translator Using 3D Video Processing. *2011 14th International Conference on Network-Based Information Systems*. [online] Available at: <https://ieeexplore.ieee.org/document/6041939/>.

### **RVL-SLLL American Sign Language Database**

R. B. Wilbur and A. C. Kak, "Purdue RVL-SLLL American Sign Language Database," School of Electrical and Computer Engineering Technical Report, TR-06-12, Purdue University, W. Lafayette, IN, 2006. (Electronic copy available from: [http://RVL.ecn.purdue.edu/databases/Wilbur\\_Kak.html](http://RVL.ecn.purdue.edu/databases/Wilbur_Kak.html))

### **TensorFlow**

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from [tensorflow.org](http://tensorflow.org)

### **Keras**

Chollet, François and others, 2015, <https://keras.io>

### **OpenCV**

Bradski, G., The OpenCV Library, 2000, <https://github.com/itseez/opencv>