DeepASL: Classifying ASL Signs using Deep Learning

Andy D. Martinez
University of Central Florida
Orlando, FL
mhdaniel24@knights.ucf.edu

Daniela Florit
University of Central Florida
Orlando, FL
danielaflorit@knights.ucf.edu

Abstract

This work focuses on developing a Deep Learning model capable of using the optical flow produced by the gestures of a signer performing a specific ASL sign and predicting the correct gloss. The work explores models which use spatial and temporal information. The developed DeepASL model is capable of performing much higher than any of the baselines. It is also shown that, in some challenging video datasets with small and non-existent optical flow between pairs of consecutive frames, encoding the entire video as a single optical flow image and extracting features using convolutions provides better results than using the temporal information.

1. Introduction

Sign Language serves as one of the most important forms of communication for the deaf community. It is estimated that about 70 million deaf people use it as their mother tongue [1], and therefore rely on recognizing the gesture patterns in order to communicate. Learning the sign representation of a word tends to be an easy task as dictionaries that map the word to the specific sequence of gestures are readily available. Figure 1 shows an example of simple phrases in American Sign Language (ASL) as described on the NIDCD web page [3]. Unfortunately, due to the visual nature of the language, there exists a lack of resources such as dictionaries to perform the inverse task, mapping from the sign to their English equivalent [4]. This represents an inconvenience for both novice learners and experienced users when encountering a new, unseen sign.

A real-time translator would facilitate the process of learning sign language, and could possibly extend a bridge between the deaf community and the hearing population. However, this represents a challenging task due to considerations such as the availability of data, the point of view of the input, sensitivity to light, background noise, and occlusions, among others. The majority of the research done targeting this or similar tasks has used feature-based approaches [4, 17, 16], and only some have tried to use CNN-based approaches [7, 2, 11].

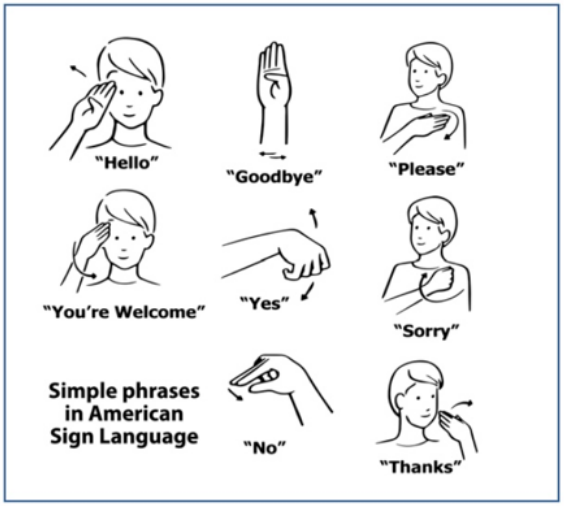

Figure 1. Simple phrases in American Sign Language

In this work, we developed a Deep Learning model which learns to recognize 20 different ASL signs by extracting features from the optical flow produced by the gestures of the signer. We called this model DeepASL. The proposed network is trained end to end using supervised learning. In addition to DeepASL, we proposed some alternative models which build on top of DeepASL.

2. Related Works

The ASL recognition task is a special case of the much broader action recognition problem, which has been a target for computer vision research for a long time, during which, multiple approaches have been proposed. Works by Polana and Nelson [12], for example, used spatio-temporal templates of motion obtained from optical flow images for the recognition of human motion. Bobick *et al*. [5] proposed the use of Hu moments obtained from motion energy images, and Rodriguez *et al*. [13] proposed a modification of

traditional MACH filters to be used for video.

As it's the case in most action recognition tasks, one of the main challenges in ASL recognition is the lack of sufficient data. The American Sign Language Lexicon Video Dataset (ASLLVD) constituted an effort to create such reference dataset for the development of vision-based systems that allow users to look-up ASL signs [4]. Most recently, The National Center for Sign Language and Gesture Resources (NCSLGR) Corpus [9] has been made available to the public, to encourage research on this area. Even with these datasets, however, the lack of sufficient labeled data represents one of the main challenges for this task. Previous works have utilized feature-based approaches [4, 17, 16] to label the signs. Other works such as those in [11, 16] have used the Microsoft Kinect to take advantage of its ability to collect information regarding depth. However, this does not provide a practical solution for real-time systems, such as those intended for mobile devices, which lack depth perception capabilities.

Most recent works have attempted to use Convolutional Neural Networks to solve the problem. Garcia *et al*. [7] proposed a CNN-based system to translate videos of ASL signs into text. However, this work centered on fingerspelled words, where the actual prediction of the CNN was a letter from the alphabet for each gesture provided, and language modeling was used to output the most likely word. Our system, on the other hand, does not include fingerspelled words. Our goal is to classify video sequences of signs into their English word equivalent, or glosses, with no language modeling. Patnaik *et al*. proposed a 3D-CNN that performed cubic convolution kernels in order to learn spatio-temporal features for this task [2]. In contrast, the work here presented intends to explore the use of spatio-temporal features by means of CNN and LSTM networks applied to the input videos optical flow in order to deal with the specific challenges provided by the dataset used.

3. Dataset

DeepASL and all other alternative models were trained using The National Center for Sign Language and Gesture Resources (NCSLGR) Corpus, which is available through Boston University's American Sign Language Linguistic Research Project's Data Access Interface (DAI) [9]. The NCSLGR Corpus consists on preprocessed sample videos obtained from three different points of view (Face, Front, and Side) from 19 short spontaneous stories. The available videos contain only the section pertaining to a specific labeled gloss, and have been preprocessed to contain the label and other relevant information as part of each frame. Since samples for each gloss were obtained from spontaneous stories, the number of occurrences of each gloss is not the same.

Sample videos corresponding to 20 glosses were downloaded. These 20 glosses were selected for being the ones with most occurrences in order to maximize the available number of examples per gloss. Only the "Front" point of view was used. Table 1 shows the selected glosses and their occurrences.

Gloss	Occurrence	Gloss	Occurrence
Really	261	Mother	76
Book	128	Same	70
Buy	124	See	68
Not	115	House	66
Car	114	That	64
Finish	107	Yesterday	55
Who	92	Know	52
In	86	But	48
Future	85	Chocolate	44
Like	81	Deaf	43

Table 1. 20 Glosses and their occurrences

Downloaded videos have been split by frames. Figure 2A shows a sample frame sequence obtained from a video corresponding to "Deaf." As all videos contained at least 36 frames, only the first 36 frames were used. Each frame was then preprocessed, as shown in Figure 2B by programmatically cropping it across its height to exclude the bottom section containing labels and other frame and video descriptions. Additionally, each frame was resized using resized using OpenCVs bilinear interpolation method to reduce the dimensionality of the original image and to guarantee that all frames have the same size.

For each pair of consecutive preprocessed frames F_i and F_{i+1}, the optical flow O_i was computed based on Gunner Farnebacks algorithm denoted by $OF(a, b)$ [6] such that:

$$O_i = OF(F_i, F_{i+1}) \quad \forall i \in [1, 35] \tag{1}$$

The resulting optical flow for each pair was color coded and saved as an image. Figure 2C a shows an example of the two optical flow images obtained from the frame pairs in Figure 2B.

4. Methods

4.1. Models:

In order to to develop our Deep Learning models, the popular framework Keras was used on top of Tensorflow. The experiments were run on CPU, which limited the ability to use deeper and richer networks. The DeepASL model consisted on a set of Convolutional Layers followed by Fully Connected Layers. On the other hand, the LSTM-based models use similar Convolutional and Fully Connected Layers to preprocess each input tensor before they serve as input to the LSTM as a timestep. Both the DeepASL model and the LSTM-based models output a probabil-

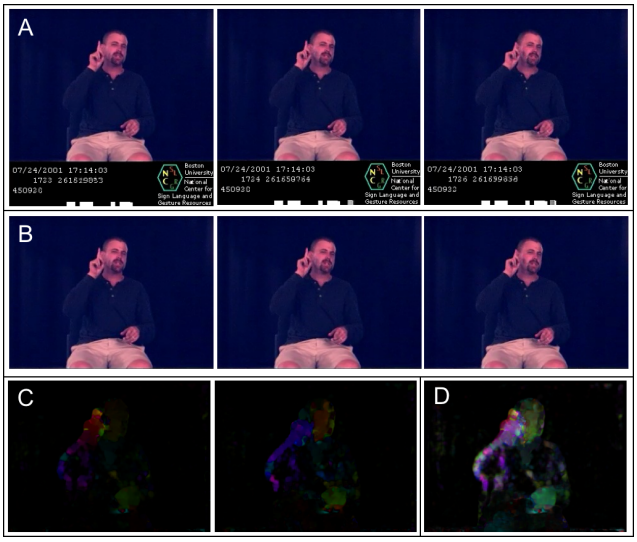

Figure 2. Dataset preprocessing. A: Original frames in video sequence for gloss "Deaf," B: "Preprocessed frames," C: "Optical Flow for pairs of frames" D: "Optical Flow of entire video sequence"

ity distribution over the 20 gloss classes. Additionally, two baselines were developed for comparison purposes: a Random Agent with a 0.05 probability of choosing any gloss class and a bias agent which always picks the gloss with most occurrences.

4.1.1 DeepASL Model:

For the development of the DeepASL architecture, hereafter denoted as D, inspiration was borrowed from the network proposed on [14]. For efficiency purposes, however, our model is a simplification of this network. The specific variations are described below:

The input received by D consists of a tensor I representing the optical flow of a video of 36 frames where $I \in R^{(48x64x3)}$, which is equivalent to an image of 48 columns, 64 rows, and RGB channels. Section 4.3 describes in more detail how this tensor is obtained. Figure 12 shows the DeepASL architecture.

The first layer of D consists of a Convolutional Layer C_1 containing 32 filters such that $C_1(I) \in R^{(48x64x32)}$. It's then followed by a Dropout Layer, and the output of this Dropout Layer, $C_{1d}(I)$, serves as input to a second Convolutional Layer, C_2, containing 64 filters such that $C_2(C_{1d}(I)) \in R^{(48x64x64)}$.

A 2D MaxPooling Layer with kernels of dimension 2x2 condenses the output obtained from the previous Convolutional Network into $Maxpool(C_2(C_{1d}(I))) \in R^{(24x32x64)}$, followed by a second Dropout Layer which outputs $Maxpool_d(C_2(C_{1d}(I)))$ with no dimensionality change. This output is then flattened into a vector of 49,152 dimensions. For simplicity the vectorized representation of $Maxpool_d(C_2(C_{1d}(I)))$ will be called $\phi(I)$ hereafter.

$$\phi(I) = Maxpool_d(C_2(C_{1d}(I))) \tag{3}$$

A Fully Connected Layer, FC_1, receives $\phi(I)$ as input such that $FC_1(\phi(I)) \in R^{512}$. Another Dropout Layer follows producing $FC_{1d}(\phi(I))$ and a final Fully Connected Layer FC_2 produces $FC_2(FC_{1d}(\phi(I)))$ to which a dropout is applied producing $FC_{2d}(FC_{1d}(\phi(I)))$. Finally, a $Softmax$ function is applied to the output of the Dropout Layer producing :

$$D(I) = Softmax(FC_{2d}(FC_{1d}(\phi(I)))) \tag{3}$$

where $D(I) \in R^{20}$, which is a probability distribution over the 20 possible glosses in our dataset.

Throughout the model, all Convolutional Layers contain padding to maintain the size of the input. Additionally, a ReLU activation function is used for all these layers, all Dropout Layers were set to 25%, and L2 regularization was used for all Fully Connected Layers.

4.1.2 LSTM based model

The input of the LSTM-base model, hereafter denoted as L, consists of a sequence of optical flow images, $S = [\Omega_1, \Omega_t, \Omega_i..., \Omega_n]$. Section 4.3] describes in more detail how the Ω_t tensors are obtained. Figure 13 shows the basic LSTM-based model architecture.

For each timestep t s.t. $1 \leq t \leq n$, $\phi(\Omega_t)$ is computed and passed through FC_1. See Section 4.1.1 for more information about these variables and formulas. An LSTM layer then receives $FC_1(\phi(\Omega_t))$ as input for each timestep t.

In order to handle the output of the LSTM, we borrowed ideas from [10], which studies the best way to use LSTM networks for temporal information where each timestep is a frame from a video. Three different approaches were consequently used: The first one consists on using the output of the last timestep from the LSTM; The second and third approaches consist in extracting the LSTM outputs for all timesteps, and performing the average and maximum respectively. Figures 13 14 and 15 show the architectures of the three variations described.

In all three cases, a vector of dimensions 512 is obtained as output. Therefore, for simplicity, all three variations will be referred to as LSTM(). Given an input series S, the output of the LSTM-based model, L, will be :

$$L(S) = Softmax(FC_{2d}(FC_{1d}(LSTM(S)))) \tag{4}$$

where $LSTM(S) \in R^{512}$

4.1.3 Baselines

For comparison purposes, a random and a bias agents were developed as baselines as described below:

Assume the set $L = \{l_1, l_2, ..., l_k\}$ contains all possible labels and $f(l_i)$ is the frequency of label l_i.

For accuracy at rank $n \leq k$, the random agent selects a random label from the set difference $L - P$ where $P = \{p_1, p_2, ..., p_{n-1}\}$ is the set of previous random predictions up to rank $n-1$, $P \subseteq L$.

For accuracy at rank $n \leq k$, the bias agent always picks l_h where $f(l_h) > f(l_i)$ s.t. $l_i \in L - P$ where $P = \{p_1, p_2, ..., p_{n-1}\}$ is the set of all previously selected predictions for ranks $1, .., n-1$.

4.2. Regularization

To avoid overfitting to the training set, two methods of regularization have been used:

- L2 regularization terms are included in all Fully Connected Layers throughout all the networks in order to keep weights small, avoiding large biases towards a specific feature.
- Several 25% Dropout layers are included in the architectures of both the DeepASL and LSTM-based models to also assist in avoiding dependencies of very specific features.

4.3. Training Strategies

The descriptions here included for training strategies concern only the DeepASL and the LSTM-based models, as neither the random nor the bias model required any pretraining in order to make predictions.

All the Deep Learning networks used a data split of 75%, 5%, 20% for training, validation and testing sets respectively. Additionally, Cross Entropy Loss was used in all cases as the loss function to be backpropagated, and the Adam Optimizer [8] was used to minimize this loss function. For all models, training was done for 100 epochs, each with a batch size of 10.

The DeepASL, which is a Convolutional-based model, receives as input on each iteration the correct label l and the optical flow of the entire video represented as a tensor $I \in R^{(48x64x3)}$. This tensor is computed by:

$$I = \sum_{i=1}^{35} O_i \tag{5}$$

where O_i is the optical flow computed from frames F_i and F_{i+1} as explained on Section 3.

A visual representation of I can be seen on Figure 2D for the sequences of preprocessed frames shown in Figure 2B. This representation proved to be very distinctive across all different signs, as it can be seen in Figure 3, making it ideal for classification. By visual inspection on 3, it can be determine which hand is predominant in each specific sign, as well as other details that help humans easily differentiate between the examples.

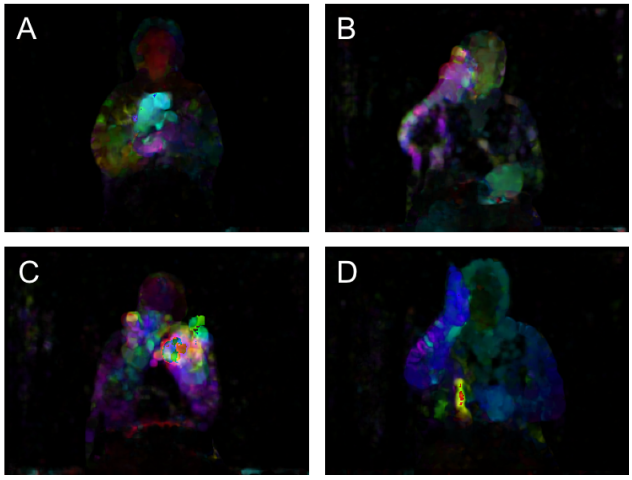

Figure 3. Optical Flow examples for different classes: A:Chocolate, B:Deaf, C:Finish, and D:Future

The DeepASL network described on Section 4.1.1 outputs a probability distribution among all the possible glosses. The 4 highest probabilities were taken for accuracy at ranks 1, 2, 3 and 4 respectively.

A naive approach to the input of the LSTM would be to use the sequence $F = [O_1, O_2, ..., O_{35}$, however, this was not used for this work because there exists only a very small, and in some cases none, optical flow between pairs of consecutive frames from the videos. For this reason, in some cases each O_i provides no information since its values are all zeros or close. Instead, we use as input to the LSTM-based models the sequence $S = [\Omega_1, \Omega_2, ..., \Omega_n]$ where each Ω_i is computed by adding a consecutive sequence of values in F. In addition to producing more meaningful inputs to the LSTM, this approach reduces the number of timesteps. Each Ω_i is computed as follows:

Assuming we are using groups of size k from F,

$$\Omega_i = \begin{cases} \sum_{j=(i-1)k+1}^{ik} (O_i) & \text{if} \quad i < \lceil \frac{35}{k} \rceil \\ \\ \sum_{j=(i-1)k+1}^{35} (O_i) & \text{if} \quad i = \lceil \frac{35}{k} \rceil \end{cases} \tag{6}$$

After performing the steps described on section 4.1.2, it produces a probability distribution output of size 20 over all the possible glosses.

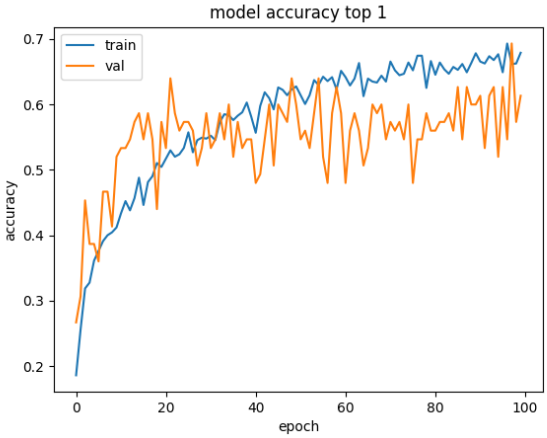

Figure 4. Convolutional Model Top 1 Accuracy per epoch

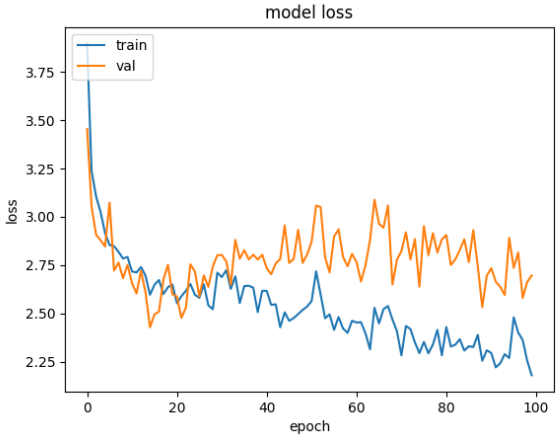

Figure 5. Convolutional Model Loss

4.4. Evaluation methods

Percent accuracy of the models on unseen data was the evaluation metric used in this work. The accuracy was analyzed up to rank 4. In addition to accuracy, the loss convergence rate on both validation and training sets were observed and analyzed. Attention was also paid to how well DeepASL clustered different classes by using t-SNE [15].

5. Results

5.1. DeepASL

During the training of the DeepASL model, the behaviors of the accuracy and loss function for the validation and training sets were tracked. This was done in order to make sure that the model was learning and converging. Figure 4 shows the behavior of both validation and training top 1 accuracies during training. It can be appreciated how both accuracies tend to increase as the DeepASL model fits the training set. As expected, the training accuracy fluctuates more than the validation one since it is the one which is being optimized, as opposed to the validation set which is composed of unseen data.

Same analysis applies to the Cross Entropy Loss plotted on Figure 5, which shows that both losses, the ones from training and validation, tend to decrease. As in the case of the accuracy, the training loss fluctuates less than the validation loss.

In addition to Percent Accuracy at Rank 1, all the accuracies up to Rank 4 were observed. Figure 6 and 7 show the behaviors of these accuracies for training and validation respectively. On both figures, it can be observed that the largest gap occurs between accuracy at rank 1 and accuracy at rank 2. Accuracies at ranks 3 and 4 provide minor improvements over rank 2, meaning that most of the labels

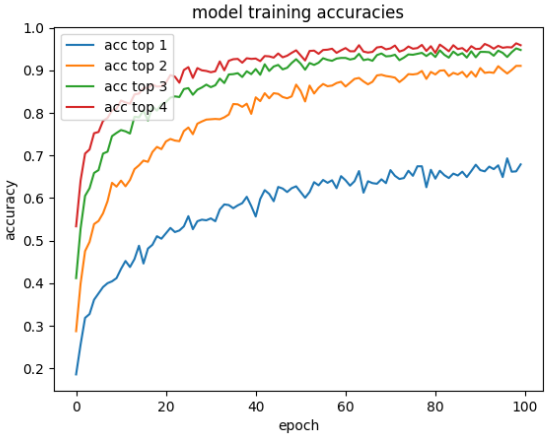

Figure 6. Convolutional Model Training

that the model will predict correctly will be on the top 2 highest probabilities on the final probability distribution of size 20.

Table 2 shows the Percent Accuracies per Rank during testing for each of the models proposed in this work. The Percent Accuracies obtained by the DeepASL model during testing were 40.4% higher than its closest LSTM-based model competitor, and 40.9% higher than the bias agent which provides the best performance out of the two baselines. This shows that the DeepASL model is learning to generalize what is has learned during training to unseen data and that it is the best out of all other implemented methods.

In addition to the Top 4 Accuracies, the confusion matrix obtained from applying DeepASL on the training set is provided on Figure 8. It is possible to observe that that most of the predictions fall on the main diagonal. Additionally, the t-SNE algorithm was used to visualize in 2 dimensions

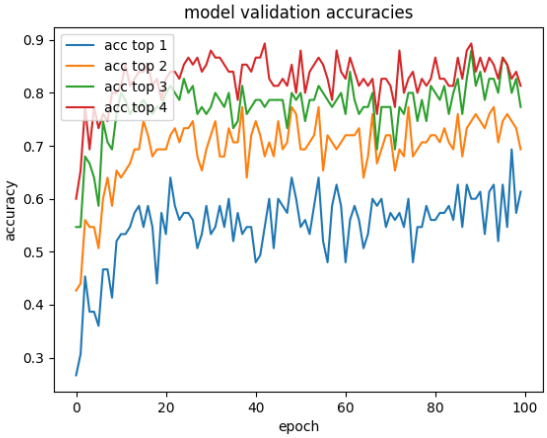

Figure 7. Convolutional Model Validation

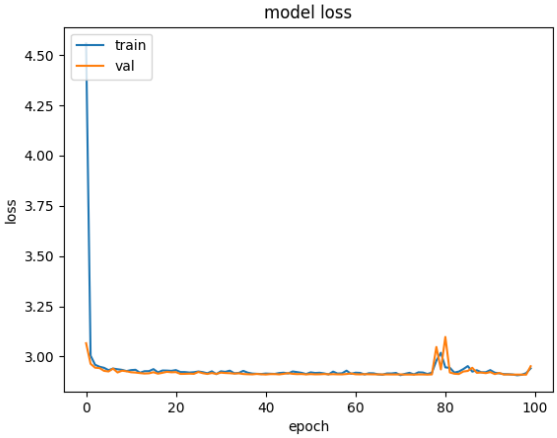

Figure 9. LSTM Model with Max Pooling Loss

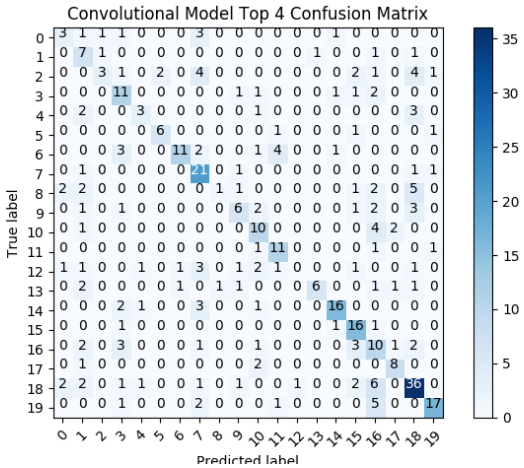

Figure 8. Convolutional Model Confusion Matrix

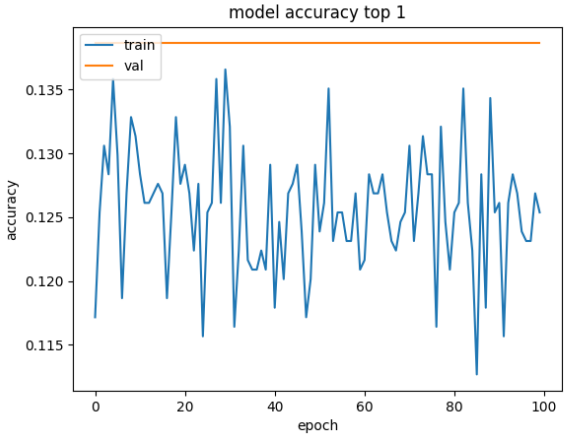

Figure 10. LSTM Model with Max Pooling Accuracy

the clusters formed by the feature vectors D(I) for all I on the training set, as shown in Figure 11. This figure shows that the DeepASL network is capable of clustering together elements that belong to the same class. The easiest ones to be seen are those that belong to classes with more examples such as car represented by blue dots. These separated clusters are to be expected due to how well the model can distinguish between them.

Due to the challenges provided by the dataset and those of the specific area of action recognition, the results obtained by DeepASL seem acceptable, however, there is not any other work on this same specific dataset to provide an accurate comparison.

5.2. LSTM-Based Models

The three LSTM-based model variations were trained as described in Section 4.3, and performance during training was analyzed. Only figures of the model training with Max Pooling after the LSTM are being reported here, however, all of the LSTM models behaved similarly as it can be appreciated on Table 2.

Figures 9 and 10 show the behavior of the Loss and Accuracies for the LSTM-model during training. It can be seen that the LSTM model fails to fit the training data, as it fails to improve the accuracy or reduce the loss from epochs 1 to 100. The performance of any of the 3 LSTM models is similar to that of the bias model. One of the reasons why it may be failing to fit the data is the lack of temporal information patterns on the dataset. This lack of patterns confuses the model while trying to discriminate among classes. Also, more data may be required to train a model that contains an LSTM, however, this seems not to be the main reason for the poor performance since during training it was unable to improve the data fit at all.

5.3. Comparison

As a final comparison, it can be appreciated that the DeepASL model, which only extracts features using convolutions, performs much better than any of the baselines or any of the LSTM-based models that try to capture additional temporal information. This seems to suggest that in datasets in which there is inconsistent temporal information, even if intuitively it may seem that the model can only benefit from it, the use of an LSTM layers can actually decrease its performance. In the cases of videos with really small amounts of change between frames, it seems like a great approach to encode the entire video by the sum of each optical flow computed from pairs of frames and extract features using just convolutions.

6. Future Work

In the task of training models capable of translating words from Sign Language to Text, one of the most relevant future works is the creation of a larger dataset that contains examples with more temporal information in the change from frame to frame. This will help models that take advantage of temporal information to avoid getting confused.

Additionally, deeper models such as VGG can be trained for the purpose of feature extraction. In this work, our network was not as deep as for example VGG due to hardware limitations since CPUs were used instead of GPUs.

Also combining the DeepASL model with a language model which learns how likely a specific word is to appear, given the rest of the sentence, should provide big improvements.

7. Conclusion

In this work we presented a Deep Learning model, DeepASL, which learns a reasonable generalization over 20 different classes of ASL glosses after having seen multiple examples of the optical flow produced by a signer performing such glosses. DeepASL shows a good generalization while learning this 20 glosses, showing great improvement over both of the baselines. Additionally, it shows that, under the specific circumstances of minimal and inconsistent temporal change between frames, an encoding of the optical flows of all the frames as a single image with no temporal information seems to work better than any of the other models tested, which try to encode and learn such temporal information. It is our hope that DeepASL could be used as inspiration and baseline to keep tackling this interesting and challenging problem of translating Sign Language to other languages such as English.

References

[1] Sign language.

[2] Learning hand features for sign language recognition. 2015.

[3] American sign language, Apr. 2017.

[4] V. Athitsos, C. Neidle, S. Sclaroff, J. Nash, A. Stefan, Q. Yuan, and A. Thangali. The asl lexicon video dataset. *CVPR 2008 Workshop on Human Communicative Behavior Analysis (CVPR4HB'08)*, 2008.

[5] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:257–267, 2001.

[6] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, SCIA'03, pages 363–370, Berlin, Heidelberg, 2003. Springer-Verlag.

[7] B. Garcia and S. A. Viesca. Real-time american sign language recognition with convolutional neural networks. *Stanford cs231 course projects*, 2016.

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[9] C. Neidle and C. Vogler. A new web interface to facilitate access to corpora: development of the asllrp data access interface. In *In Proceedings of the International Conference on Language Resources and Evaluation*, 2012.

[10] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, June 2015.

[11] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen. Sign language recognition using convolutional neural networks. *Computer Vision - ECCV 2014 Workshops Lecture Notes in Computer Science*, page 572578, 2015.

[12] R. Polana and R. Nelson. Low level recognition of human motion (or how to get your man without finding his body parts). In *Proceedings of 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 77–82, Nov 1994.

[13] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[15] L. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[16] H.-D. Yang. Sign language recognition with the kinect sensor based on conditional random fields. *Sensors*, 15(1):135–147, 2015.

[17] M. M. Zaki and S. I. Shaheen. Sign language recognition using a combination of new vision based features. *Pattern Recogn. Lett.*, 32(4):572–577, Mar. 2011.

	Rank1	Rank2	Rank3	Rank4
Bias Model	0.1456043956	0.217032967	0.2857142857	0.3489010989
Random Model	0.04395604396	0.09340659341	0.1675824176	0.206043956
Conv Model	0.5549450556	0.6620879127	0.7225274732	0.7747252754
LSTM Max	0.150887574	0.2218934911	0.2958579882	0.3609467456
LSTM Avg	0.150887574	0.224852071	0.2958579882	0.3609467456
LSTM LTS	0.150887574	0.224852071	0.2958579882	0.3609467456

Table 2. Accuracy per rank

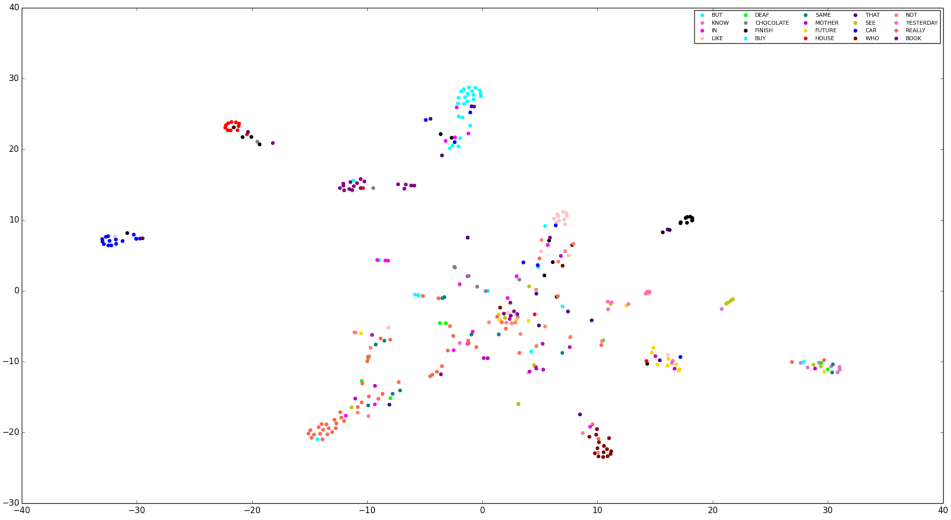

Figure 11. TSNE visual representation

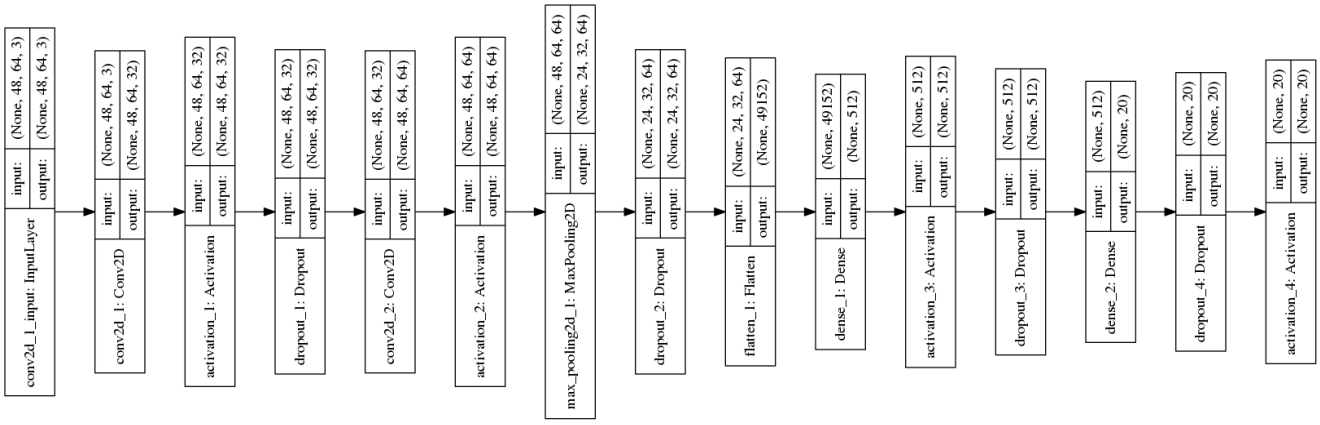

Figure 12. DeepASL Model Architecture

8. Supplemental Images:

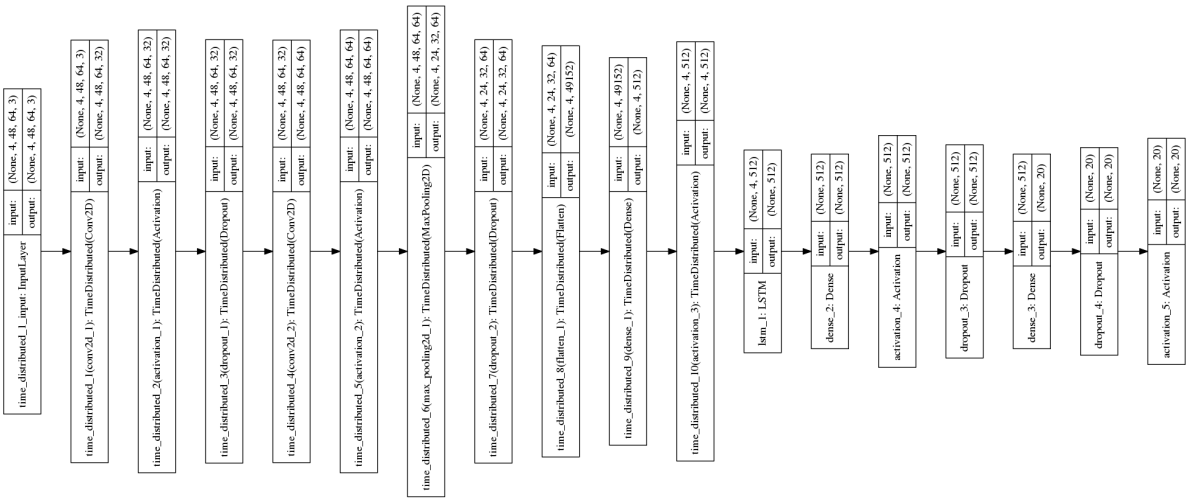

Figure 13. Architecture of LSTM Model with no Pooling

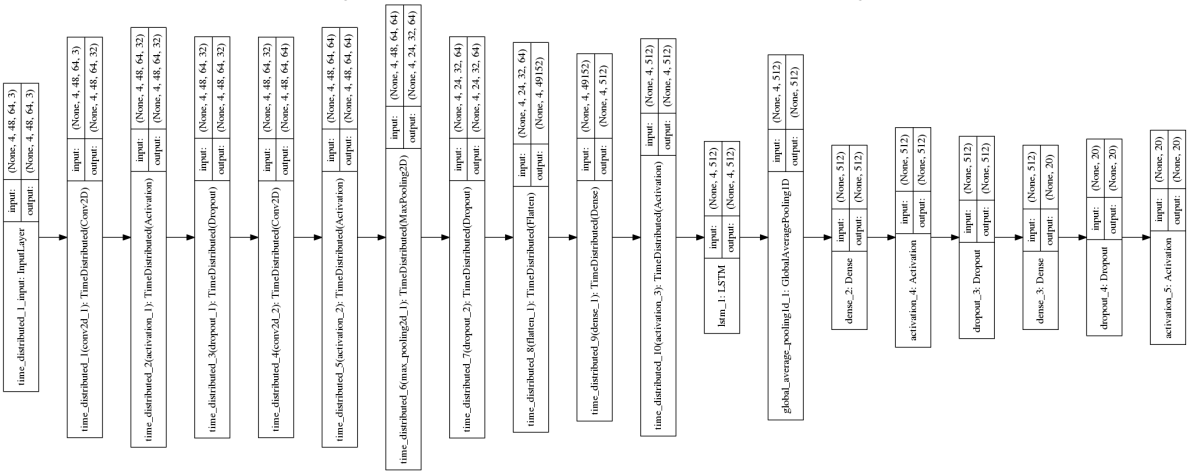

Figure 14. Architecture of LSTM Model with Average Pooling

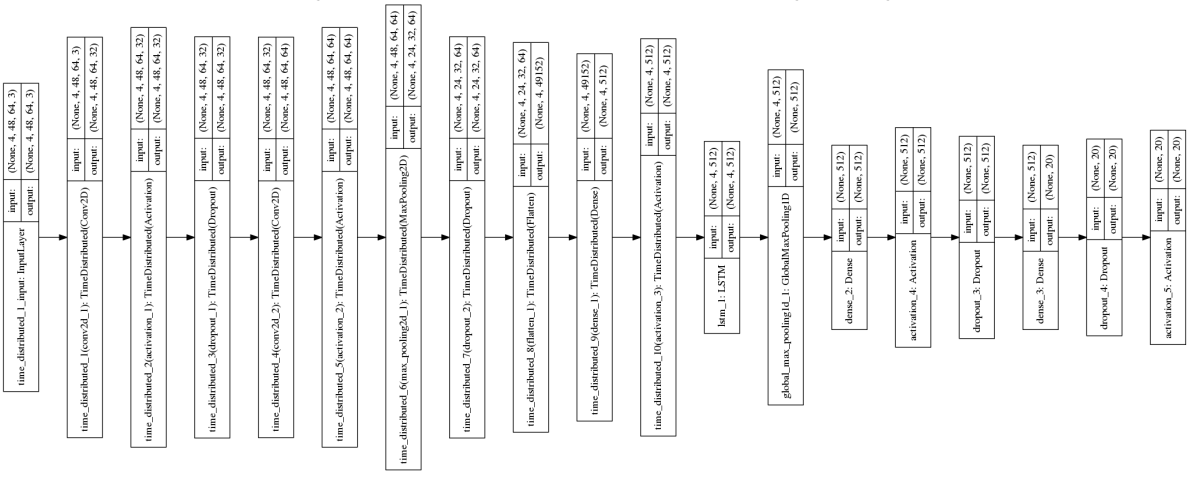

Figure 15. Architecture of LSTM Model with Max Pooling