Learning Hand Features for Sign Language Recognition

Kaushik Patnaik
Georgia Institute of Technology
kpatnaik@gatech.edu

Payam Siyari
Georgia Institute of Technology
psiyari3z@gatech.ed

Vinodh Krishnan
Georgia Institute of Technology
krishnan.vinodh@gmail.com

Vivek Nabhi
Georgia Institute of Technology
viveknabhi@gmail.com

Abstract

Sign languages are primarily used for communication with deaf people and people who can hear but can not physically speak. We intend to propose a system for sign language recognition. In the simplest form, recognition can be interpreted as conversion of sign language to the text in other regular languages, although some systems go one step further by giving spoken language as the output.

We propose to address the spatio-temporal nature of the problem by employing Convolutional Neural Networks that learn features across time. Following recent works, in order to extend traditional CNNs to include temporal information, we use 3D convolutions where common weights are learned over several frames and whole image. Our chief contribution lies in adapting these existing methods for learning sign language features that are both spatial and temporal in nature.

1 Introduction

By recent estimates around 360 million people all over the world suffer from hearing loss. Sign Language is the most popular means of communication for people in this group. This however poses a problem because a majority of the hearing population of the world does not understand sign language. There has been a lot of research put into translating sign language into spoken language but doing this in real time is a complex task.

Sign Languages in general, specifically American Sign Language (ASL) are not simple visual representations of spoken languages but, are fully developed standalone languages having their own syntactic and grammatical rules [1]. Similar to regular languages they have a temporal aspect to them. Furthermore, they have spatial features to convey meanings and concepts as the communication happens through a visual medium. This makes the problem of translating sign language to regular speech or text an extremely challenging one. In addition to the inherent complexity of the language, sign language recognition systems also have to tackle problems present in the data like dynamic backgrounds, data occlusion, decoding gestures which involve not only the hand as a whole but also specific parts of the arms or the signers fingers etc.

Sign Language translation systems typically have 3 major components, tracking hand gestures, recognizing hand shapes and translating the hand shape. These tasks are approached individually by handcrafting features which are common to computer vision problems like SIFT, HOG etc. The features are generally further augmented by human intuition. Our system attempts to recognize words in American Sign Language from videos of experienced signers signing ASL words. For this we use the ASLLVD (American Sign Language Lexicon Video Dataset) which contains more than

3,300 words, each signed by 1-6 experienced ASL signers. We address the spatio-temporal nature of the problem by employing a Convolutional Neural Network (CNN) which uses 3D convolutions to learn features across time. We also compare the performance of the 3D CNN with a 2D CNN. Our contribution lies in adapting the existing methods to learn sign language features which are both spatial and temporal in nature.

We propose a 2D-CNN and a 3D-CNN model for sign language recognition. The difference between these two models is the fact that in 3D-CNN, we use cubic convolution kernels in order to derive feature representations across time. We show that 3D convolutional neural networks are able to learn spatio-temporal features for the problem of sign language recognition. We show this by visualization of the learnt features by 3D-CNN model. 3D-CNN also provides better Top-3 and Top-5 classification accuracy than 2D models.

The outline of the paper is as follows. In section 2 we present a literature review of the current research in the realm of sign language recognition and also discuss the application of deep learning to this problem. An outline of the data used is provided in section 3. In Section 4.1 we discuss the steps taken to pre-process the data set, 4.2 deals with the architectures of the 2D CNN and 3D CNN models and our findings are summarized in section 4.3. We draw conclusions from our observations in section 5 and also present directions for future research.

2 Related Work

Early attempts at applying machine learning to sign translation relied on the signers wearing instrumented gloves or utilizing desktop cameras along with template matching or neural networks in order to recognize gestures. Hidden Markov Models also gained a lot of popularity especially through the works of Starner et. al. [2] who use a HMM model with data from a desk and a wearable computer and achieve 92 and 98 percent word accuracies respectively, using a lexicon containing 40 different words in the ASL.Liu et. al. [3] develop a gesture recognizer where they leverage depth data which allows them to identify gestures without depending on hand segmentation methods like skin color or making the signer wear gloves.

In gesture recognition and translation, Ao Tang et al. [4] employ deep learning approaches to classify hand postures. However they ignore the spatio-temporal nature of the problem and only choose to do classification on single frames of hand postures. Although their models exhibit high accuracy, direct classification from static hand postures is pretty straightforward using deep models. Several different approaches have been proposed in literature [5] [6] to extend traditional CNNs to include temporal information. Work by Shuiwang Ji et al. [5] uses 3D convolutions, where common weights are learned over several frames and whole image. Work by Jeff Donahue et al. [6] uses a model composed of CNNs and LSTMS to carry visual recognition and descriptions. Image frames of the video are fed into multiple CNNs, whose output are fed into LSTMs to generate video descriptions. Both these approaches have shown improvement over feature based approaches in the tasks of video classification [6] [7].

3 The Dataset

This paper used the data from the American Sign Language Lexicon Video Data set (ASLLVD) [8] [9] [10]. The data set contains video sequences in uncompressed raw format for $> 3,300$ words in the ASL. Each word was signed by 1-6 native ASL signers generating a total of 9,800 tokens. The data was collected in Boston university. Each signer was shown a video stimulus and were asked to produce the sign they saw. The videos were collected using 4 synchronized cameras capturing a side view of the signer, a close up of the head, a half speed high resolution front view and a full resolution front view. The video sequences in the data set are processed to produce high fidelity in the hand and face regions as they contain most information about the signed word. Automatic skin segmentation is applied and the resulting frames are cropped to the skin and normalized to ensure uniform brightness. Lexical variants of a particular sign are grouped along with the original sign in the data set. The videos were captured at a frame rate of 60 frames per second but the playback happens at 1/4th of that rate (15 frames per second).

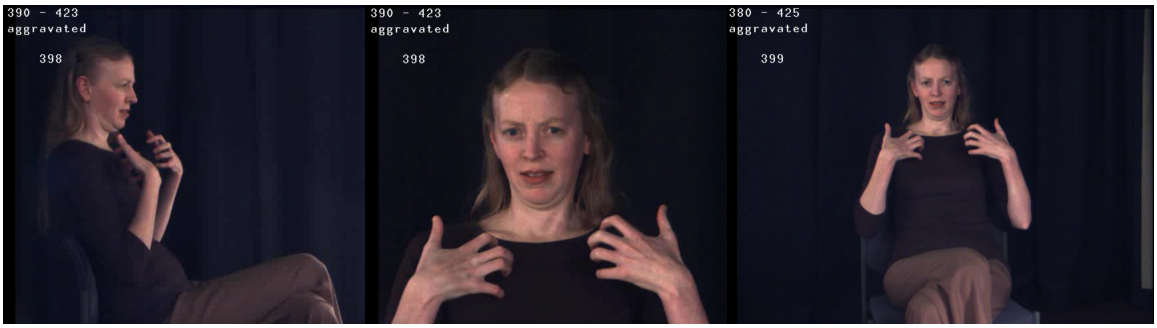


Figure 1: Side, zoomed and front views of the signer as present in the ASLLVD dataset

This paper used a sample of this data set containing 10 classes (words). Each word had around 5-6 videos showing the full resolution front view and the side view of the signers. Fig. 1 shows a sample of the side and front views of the signer as present in the data set.

4 Approach

4.1 Data Preprocessing

We extract the frames from the videos taken for each of the words for every signer. There were a total of 5-6 signers and we extracted data for 10 classes. Once we obtained the frames, we averaged the frames for every user for front and side views separately and subtracted the mean from every image, effectively normalizing them. This ensured that we captured the hand movement effectively and accounted for the background noise. We resized the images to 200x200 and converted it to grayscale. Once the grayscale-resized images were obtained, we eliminated the first 20 and the last 20 images. This is because these images mostly consisted of the hand in resting positions. This left us with about 80 images per user. Since the transition between the images was slower if we take every image, we took one in every 4 consecutive images. This ensured that the sequence of images was able to effectively capture the change in hand position while performing the sign.

We generated additional images by applying:

- Rotation - Here we rotated the images by 5 degrees and 10 degrees on either side.
- Noise - We applied salt and pepper noise on the images with a probability of 0.05 for each pixel. We generated 5 additional images through this for each image.

After applying the above preprocessing and generating techniques, we had about 400 images (200 for front view, and 200 for side view) for each person for each sign. These images were then supplied to the model based on the batch size and the in_time parameter. The batch size parameter stood for how many sequences of images the model required, and the in_time parameter is how many consecutive images it wanted in each sequence. The data preprocessing flow is outlined in Fig. 2

4.2 Model Architectures

4.2.1 Spatial Model: 2D Convolutional Neural Network (2D-CNN)

Classic convolutional neural network (CNN) is the baseline architecture we experiment on. Since convolutions are applied spatially, we call these architectures 2D-CNNs, as opposed to 3D-CNN models (next section) which also apply convolution across time.

The most common CNN architecture includes a series of convolution and Rectified Linear units (Conv-ReLU), possibly along with max pooling stages (MaxPool). We use the same architecture in this paper. As observed in Fig. **??**, our model has four CONV-ReLU layers, three of them followed by MaxPool layers. All CONV filters have been applied with the stride of one and MaxPool filters are applied with window size 2x2 and stride of one. In summary, we have a 200×200 grayscale

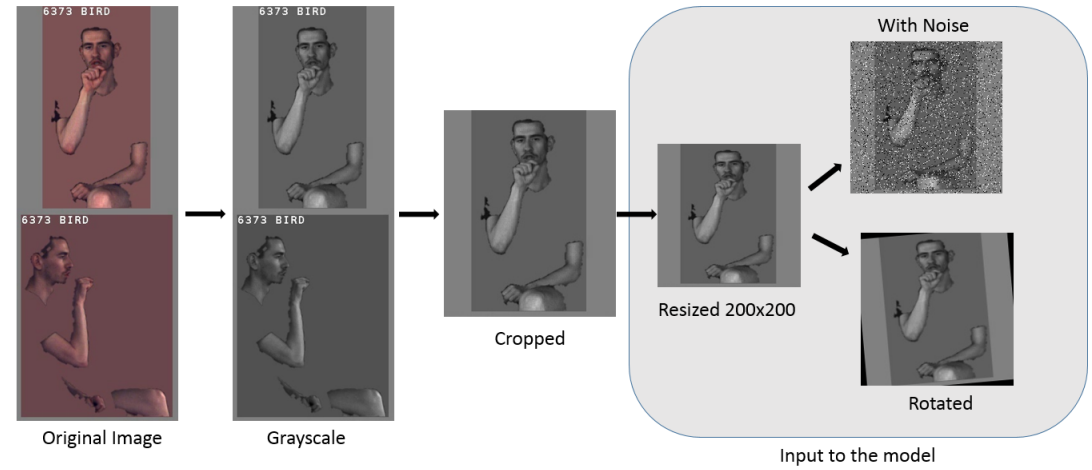


Figure 2: Data preprocessing done on each video frame

input image. Then in the first and second layer, we apply 32 and 64 kernels of size 5×5, respectively and follow all kernels with ReLU filters. Both of these layers are followed by a MaxPool stage. In the third layer, we have a Conv-ReLU stage with 96 kernels of size 3×3. Following foundational works on CNN models (e.g. [11]), we do not use a pooling stage in this layer. Lastly, in the fourth layer, we apply filtering with 128 kernels of size 3×3, followed by a MaxPool stage, resulting in 128 codings of size 21×21. In the fifth layer, as for the classification, we flatten the last layer codes to 1024 neurons by connecting all neurons in the fourth layer to all neurons in fifth layer, following by a ReLU stage. For extracting class scores, we use a softmax layer by fully connecting it to the fifth layer. We use categorical cross-entropy loss as the classification objective.

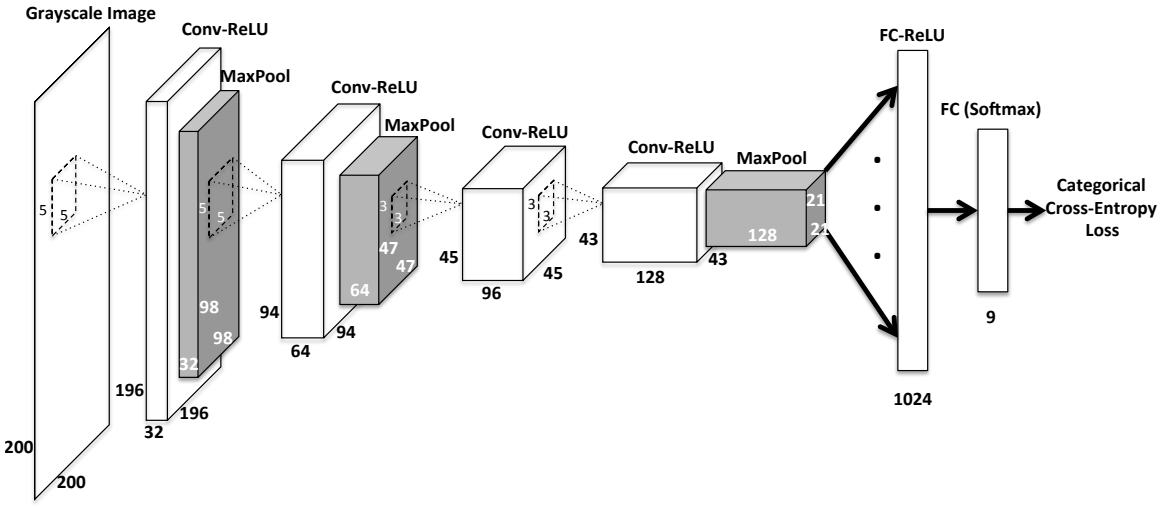
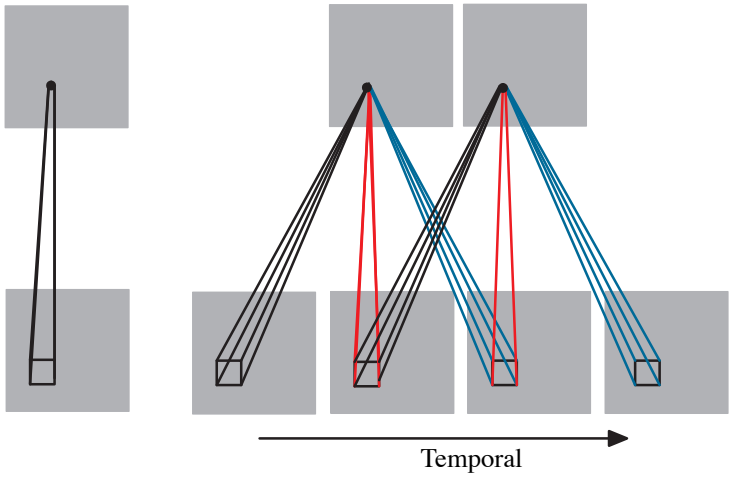


Figure 3: 2D-CNN Model Architecture

4.2.2 Spatio-Temporal Model: 3D Convolutional Neural Network (3D-CNN)

Similar to work done in [5] who use 3D convolutions, we propose a 3D CNN model in order to account for both spatial and temporal nature of our data. In a 3D-CNN model, the model generates multiple channels of information from adjacent video frames and performs convolution and pooling separately in each channel. Combining all channel codes results in a coding representation for the input sequence of video frames.

The 3D convolution operator is applied by convolving a cubic kernel to the cube formed by stacking multiple contiguous frames together [5]. Basically, every aspect of 2D convolutions will be generalized across time for 3D convolutions. In other words, applying each 3D convolution to a sub-sequence of images results in a shorter image sub-sequence and this reduction can be controlled by how long the coverage of 3D convolution filter is and how much overlapping in terms of time we consider for applying 3D convolution. Furthermore, we usually have both spatial and temporal weight-sharing for 3D convolution operators. This type of architectures are known as "Slow Fusion"architectures [7]. Fig. 4 (from [5]) explains more clearly the difference of 2D models and 3D models. In our model, we have 4 input image frames. For temporal convolution, we have two



(a) 2D-CNN (b) 3D-CNN

Figure 4: [5]: "Comparison of 2D (a) and 3D (b) convolutions. In (b) the size of the convolution kernel in the temporal dimension is 3, and the sets of connections are color-coded so that the shared weights are in the same color. In 3D convolution, the same 3D kernel is applied to overlapping 3D cubes in the input video to extract motion features. "

three-layer parallel 3D convolution column (time extent of two), exactly with the same structure as in 2D-CNN in Fig. 3 in first three layers. In the fourth layer, we apply a simple 2D convolution and aggregate the feature maps across time, before we flatten and feed the codings to fully connected layers. Fig. 5 depicts our proposed architectur.

4.3 Experimental Evaluations

4.3.1 Implementation Details

Both the 2D-CNN and 3D-CNN model are trained using standard backpropagation algorithm. We use RMSProp [**?**] optimization within a mini-batch stochastic gradient descent framework. We choose to take 4 frames of the preprocessed data (after skipping by 4 frames jumps and adding noise, rotation, etc.) as a single input to the model, as a sign usually takes multiple frames to convey. Extending the initial input to 10-15 frames involves a considerable computational cost.

In 2D-model, we simply classify each frame that is fed into the model. For 3D-model, we classify a sequence of images as one sign.

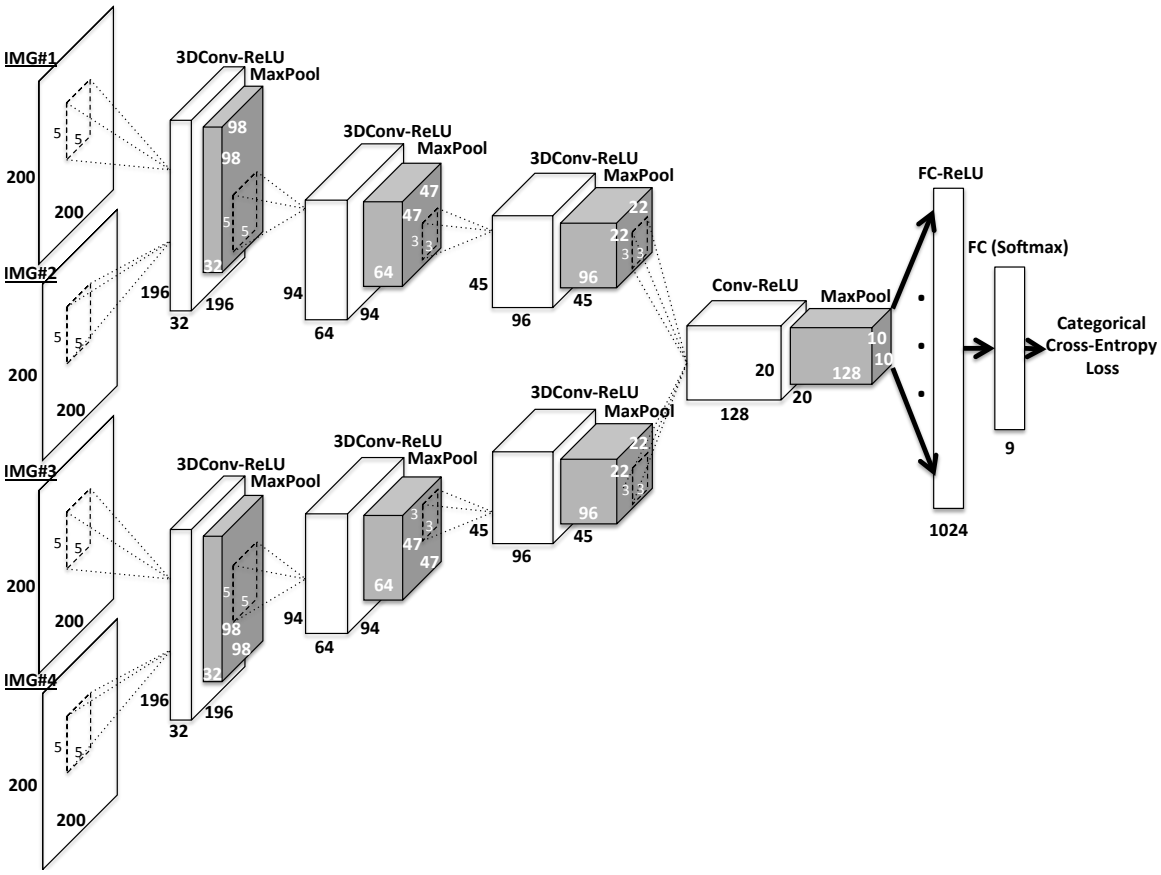


Figure 5: 3D-CNN Model Architecture

We implemented both models in Theano [12] and ran each of them on two servers with 2×6 cores AMD Opteron 4238, 64GB RAM and running Linux Red Hat Enterprise (Santiago).

4.3.2 Results

We measure Top-1, Top-3 and Top-5 accuracy of the classification in each epoch to present our results.

Comparison of 2D and 3D performance

Table 1 represents the results for 2D-CNN and 3D-CNN classification accuracy. Although 3D-CNN has inferior performance than 2D-CNN model for Top-1 accuracy, 3D-CNN produces significantly more accurate Top-3 and Top-5 results and hence a more generalizable model for sign recognition.

Table 1: Comparison of classification accuracy of 2D and 3D-CNN models (Best results for each measure)

Measure/Model	2D CNN	3D CNN
Top-1	0.276	0.211
Top-3	0.4225	0.556
Top-5	0.439	0.556

Visualization

Fig. 6 and 7 show the visualized filters for first and second layers of the 3D model. Intuitionally, such visualizations are desirable since they show the ability of the model for edge detection. We are able see the areas where the signer resides, esp. the hand and face regions are commonly more active than other areas.

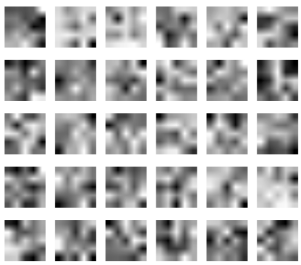


Figure 6: Visualization of 30 features in the first layer of 3D model. Here the weights are initialized randomly

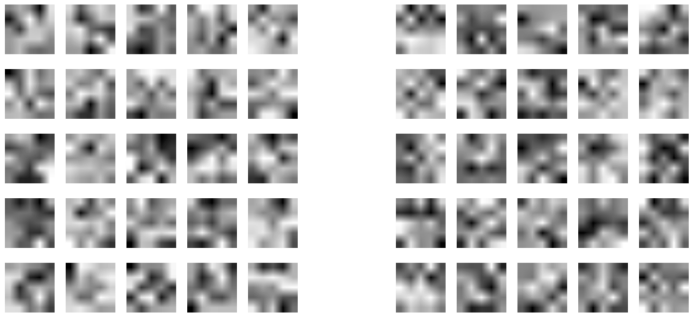


Figure 7: Visualization of 30 features in the first layer of 3D model. Here the weights are initialized randomly

5 Conclusions and Future Research

We showed 3D convolutions are able to learn spatio-temporal features but need better initializations. Current initialization schemes involve 2D CNNs learnt from the same data.

Several paths for future research may be considered:

- Current models do not involve max-pooling across time, this leads to time variant features. Adding max-pooling across time to investigate what time-invariant features the model may learn is an interesting problem for research.
- For this particular task, weights can be made biased to regions where the hands exist. It is interesting to see how this will affect the model accuracy, in both 2D-CNN and 3D-CNN models.
- Comparison with alternate spatio-temporal learning methods such as CNN-RNN is also a benchmark of interest for sign language recognition.

Acknowledgments

We thank Networking and Telecommunication Group at College of Computing, GeorgiaTech, for providing us the server computers.

References

[1] http://en.wikipedia.org/wiki/Sign_language.

[2] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(12):1371–1375, 1998.

[3] Xia Liu and Kikuo Fujimura. Hand gesture recognition using depth data. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 529–534. IEEE, 2004.

[4] Ao Tang, Ke Lu, Yufei Wang, Jie Huang, and Houqiang Li. A real-time hand posture recognition system using deep neural networks.

[5] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):221–231, 2013.

[6] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*, 2014.

[7] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1725–1732. IEEE, 2014.

[8] Carol Neidle and Christian Vogler. A new web interface to facilitate access to corpora: Development of the asllrp data access interface (dai). In *Proc. 5th Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon, LREC*. Citeseer, 2012.

[9] Carol Neidle, Ashwin Thangali, and Stan Sclaroff. Challenges in development of the american sign language lexicon video dataset (asllvd) corpus. In *Proc. 5th Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon*, 2012.

[10] http://secrets.rutgers.edu/dai/queryPages/search/search.php.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[12] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.