AMERICAN SIGN LANGUAGE VIDEO CAPTIONING WITH OBJECT DETECTION AND TRANSFER LEARNING

IE 590: DEEP LEARNING IN MACHINE VISION

Michael Wang
School of Industrial Engineering
Purdue University
wang3246@purdue.edu

Shuzhan Sun
School of Electrical & Computer Engineering
Purdue University
sun630@purdue.edu

December 15, 2019

ABSTRACT

American Sign Language (ASL), although widely used by the deaf community, has a limited number of signers in the hearing community. To bridge the communication barrier between the deaf community and the general public, in this work, we developed a real-time video captioning system for ASL. From our testing, we found many of existing CNN-based recognition networks suffer from reduced accuracy for new backgrounds and different light conditions, which limited the reliability of a real-time video captioning system. Here, we proposed a framework that included 1) transfer learning trained CNN model on very large dateset to achieve an optimized CNN architecture, 2) utilizing object detection network to reduce the effect from background, and 3) optimizing various regularizations for ASL dataset. Ultimately, the video captioning system achieved $95\%+$ accuracy on the self-recorded test dataset and is able to caption live stream video.

Keywords American Sign Language · Image Captioning · Object Detection · Transfer Learning

1 Introduction and Related Work

There is a disconnect in communication between the deaf community and the general public. This is especially problematic in a live conversation where the hearing individual cannot sign or interpret sign language. An automatic captioning system would help bridge this gap and facilitate communication. This captioning system will convert gestures (American Sign Language Fingerspelling) to captions (alphabet letters and numbers), which can be applied to still images and videos. Other projects have attempted to A real-time captioning system would help bridge the communication barrier between the deaf community and the general public. American Sign Language (ASL) is the predominant sign language used by deaf communities in the United States. In ASL, there are gestures for words and letters along with other nuances such as regional differences and accents. To further complicate things, body and facial movements are critical in the realization of ASL signs [1]. One current deep learning approach examine fingerspelling, or spelling out words by signing one letter at a time [2]. Rastgoo et al. used CNNs and Restricted Boltzmann Machines (RBM) to extract and recognize still images of ASL signs from a combination RGB and depth modalities. Another study uses accelerometer data to classify rightward vs. leftward gestures [3]. Little progress has been made in applying of deep learning to recognize full gestures (words) in ASL due to the complexity of depth and motion.

2 Methodology

The video captioning system receives a video of a user fingerspelling as input, separates each frame of the video as a separate image, finds the region of interest, and classifies each frame to a class (letter) in ASL. Then, the output of this workflow is a captioned video with the class with highest likelihood on each frame. Ideally, these labels will be concatenated together as the video progresses to form words. The whole process is illustrated in Fig. 1.

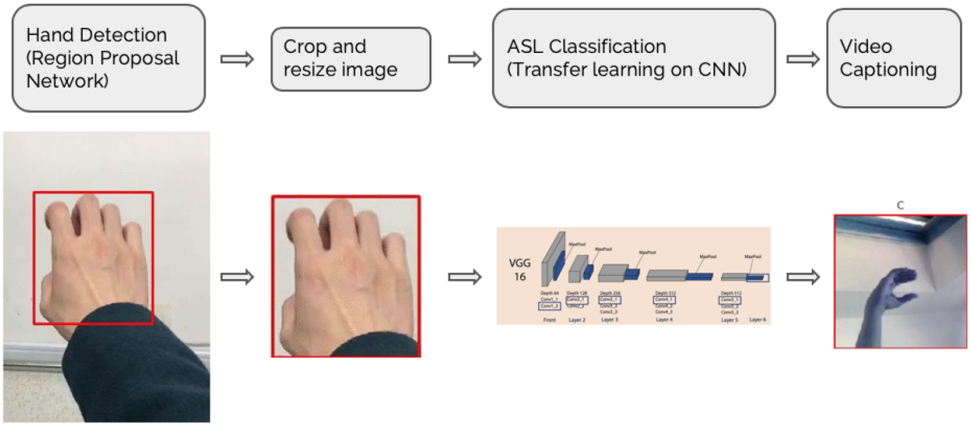
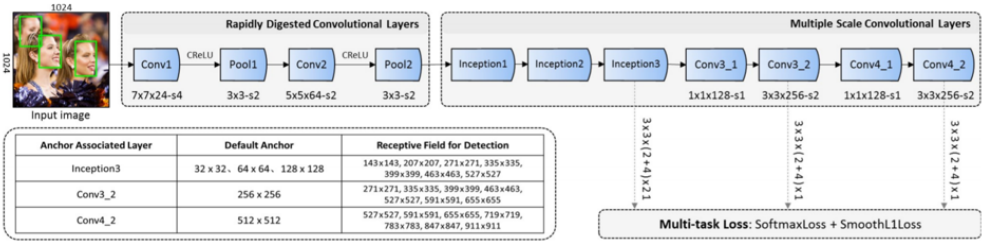


Figure 1: Framework of the proposed video captioning system.



Anchor Associated Layer	Default Anchor	Receptive Field for Detection
Inception3	32 x 32、64 x 64、128 x 128	143×143, 207×207, 271×271, 335×335, 399×399, 463×463, 527×527
Conv3_2	256 x 256	271×271, 335×335, 399×399, 463×463, 527×527, 591×591, 655×655
Conv4_2	512 x 512	527×527, 591×591, 655×655, 719×719, 783×783, 847×847, 911×911

Figure 2: Architecture of the FaceBoxes and the detailed information table about the anchor designs.

2.1 Data Generation

Apart from open-source ASL Alphabet Dataset, a second dataset was created specifically for this study. Using a computer webcam, all 26 letters (A-Z) of the ASL fingerspelling alphabet were recorded in different angles and distances from the camera. Then, each of the frames were split, labeled, and used for training data. This dataset supplemented the ASL Alphabet Dataset. Both will be detailed in the next section.

2.2 Object Detection Network

To detect the hand from background, we use a so called "FaceBox" network [4], which is first proposed to detect faces. The architecture of the FaceBox network is shown in Fig. 2, consisting of Rapidly Digested Convolutional Layers (RDCL) and the Multiple Scale Convolutional Layers (MSCL). The RDCL is designed to enable FaceBoxes to achieve real-time speed on the CPU. The MSCL aims at enriching the receptive fields and discretizing anchors over different layers to handle faces of various scales. The FaceBox network has a 96% accuracy on finding the region of interest according to [4]. In our system, for detecting hands, we use a FaceBox network pretrained on EgoHands dataset.

2.3 Classification Network

After the object detection network detects the hand and crops the image at the predicted bounding box, the image is then sent to the convolutional neural network (CNN) for ASL classification. The base network was pretrained on ImageNet and downloaded from PyTorch for further training. The process of only retraining a section of a large pretrained network on a custom dataset is called transfer learning. Since the first few layers of the pretrained CNN does a good job detecting geometries and patterns, the last fully connected layers can be trained to classify based on a custom dataset. In this case, the 1000 output classes of the ImageNet classifier was fully connected to an intermediate layer of 256 classes, and then that intermediate layer was fully connected to the final output layer of 26 ASL fingerspelling classes (letters A-Z). These were trained with ReLU nonlinearity, dropout probability of 0.2, Adam optimizer, and log softmax loss. Both ResNet50 and VGG16, trained on ImageNet, were tested with this transfer learning framework. This code was modified for our own application from a tutorial on transfer learning [5] and PyTorch's website on transfer learning [6].

To further vary our datasets, data augmentation and processing was performed before training. This included rotating, flipping, and randomly center cropping the input images so that the training data could be expanded and overfitting could be reduced. The mean ([0.485, 0.456, 0.406]) and standard deviation ([0.229, 0.224, 0.225]) of ImageNet were subtracted from all training data (normalized) to be consistent with the pretrained network. During training, we used early stopping to halt the training if validation loss did not improve for five consecutive epochs. Then, the network was frozen and the checkpoint was made so that the model could be referenced without retraining the entire network again.

	ResNet50	**VGG16**
Non-trainable Parameters	23,508,032	95.26%
Trainable Parameters	531,226	1,055,514
Training time/epoch	636s / 78s	784s / 101s

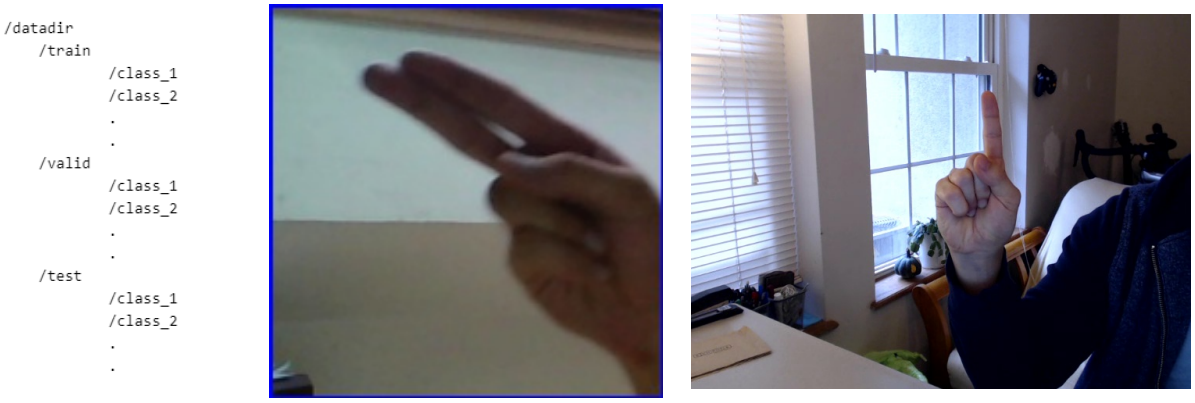
Table 1: Details of ResNet50 and VGG16 networks for training. Training times include time per epoch for both datasets.

2.4 Video Processing

To process the videos, two options are available. The first is live video captioning. Using OpenCV in Python 3, the webcam serves as input to both networks. Each frame is processed as a separate image for both hand detection, cropping, and classification. The predicted class is annotated on the frame and then displayed to the user. Second, the user can upload existing footage to the root directory. From there, a script included in the Jupyter notebook splits up the video into separate frames, processes and labels those frames, and the output is saved as a `.gif` file.

2.5 Dataset

Two datasets were used: a publicly available ASL Alphabet Dataset and a self-recorded and labeled dataset. The organization hierarchy of both datasets were consistent with the standard organization for convolutional neural networks. This also makes it easier for PyTorch's data loader to load in the data.



(a) Hierarchy (b) ASL Alphabet Dataset (c) Self-recorded Dataset

Figure 3: The organization hierarchy and example images of (b) ASL Alphabet Dataset and (c) Self-recorded Dataset.

The ASL Alphabet dataset is a collection of images of alphabet characters from the American Sign Language, separated into 26 folders. The training data contains 87,000 images with size 200x200 pixels each. There are 26 classes, one for each of the letters A-Z. In Figure 6 below the details of the dataset are shown. Note that there is only one image in the column `n_test`. This is because the dataset allows for the user to load their own test data into the designated folders.

In the self-recorded dataset, each sign was recorded in a video in different angles and distances from the camera. Then, each frame was spliced, labeled, and saved as a separate image for training, testing and validation. First, 20% of the images were designated as test, then another 20% of the remaining images were designated as validation. The remaining images were designated for training.

category	n_train	n_valid	n_test
W	3416	30	1
R	3406	30	1
C	3255	30	1
K	3128	30	1
L	3056	30	1
J	3031	30	1
S	3021	30	1
B	3016	30	1
V	3008	30	1
Z	3001	30	1
Y	3000	30	1
P	2989	30	1
D	2983	30	1
A	2976	30	1
H	2939	30	1
G	2932	30	1
I	2926	30	1
E	2904	30	1
X	2889	30	1
U	2854	30	1
Q	2848	30	1
O	2846	30	1
M	2818	30	1
N	2777	30	1
F	2776	30	1
T	2695	30	1

(a) ASL Alphabet dataset

category	n_train	n_valid	n_test
A	466	98	123
S	448	74	93
T	390	96	121
R	344	74	92
W	338	68	86
P	319	79	98
C	312	58	73
N	292	72	90
O	286	70	88
D	285	60	75
B	281	69	87
L	279	68	86
I	273	60	75
V	271	67	84
H	250	58	72
M	240	59	74
Y	237	58	73
U	227	55	70
X	223	54	68
F	222	54	68
K	218	53	67
J	211	51	65
G	204	50	63
Q	201	49	62
Z	194	47	59
E	158	38	48

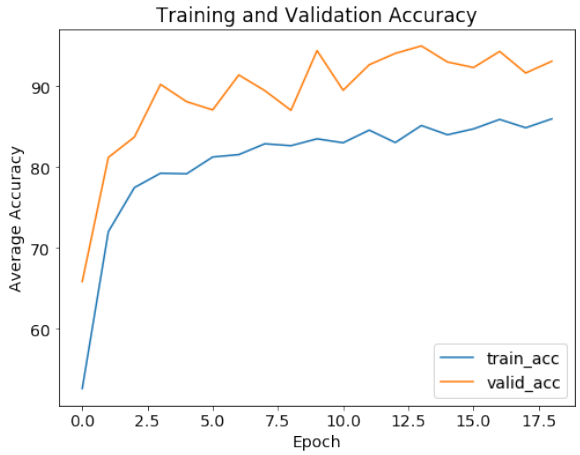
(b) Self-recorded dataset

Figure 4: Quantity of images in of each class for training, validation and test for (a) ASL Alphabet dataset and (b) Self-recorded dataset.

3 Experiments and Results

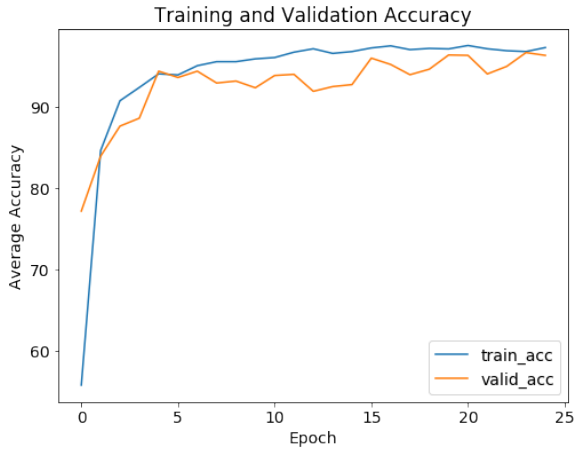
In ASL classification, the transfer learning model used was ResNet50 pretrained on ImageNet. ResNet50 was chosen because it has fewer untrained parameters and took less time to train the model. As detailed above, an additional two fully connected (FC) layers were trained to map from the 1,000 ImageNet classes to the 26 alphabet characters of ASL. The model trained on ASL Alphabet dataset is named Model 1 and the model trained on the self-recorded dataset is Model 2.

The training and validation accuracy and loss for both models are shown below in Figures 7 through 10. These values are for images during training on the same respective dataset. Model 2 achieved upwards of 97.07% training and 96.94% validation accuracies while Model 1 achieved upwards of 85.23% training and 78.39% validation accuracies.

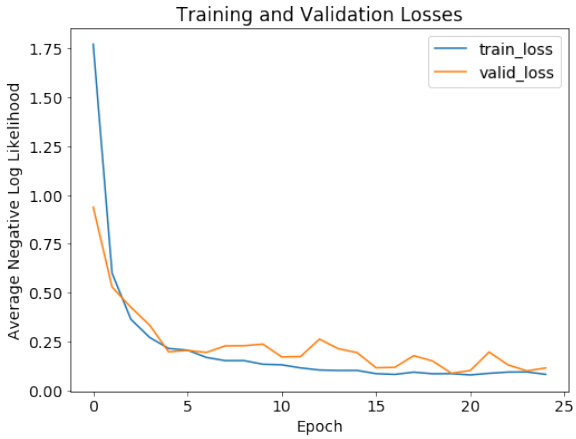


(a) Model 1: Training accuracy

(b) Model 1: Training loss



(c) Model 2: Training accuracy



(d) Model 2: Training loss

Figure 5: Training accuracy and loss vs. number of epoch for Model 1 and Model 2.

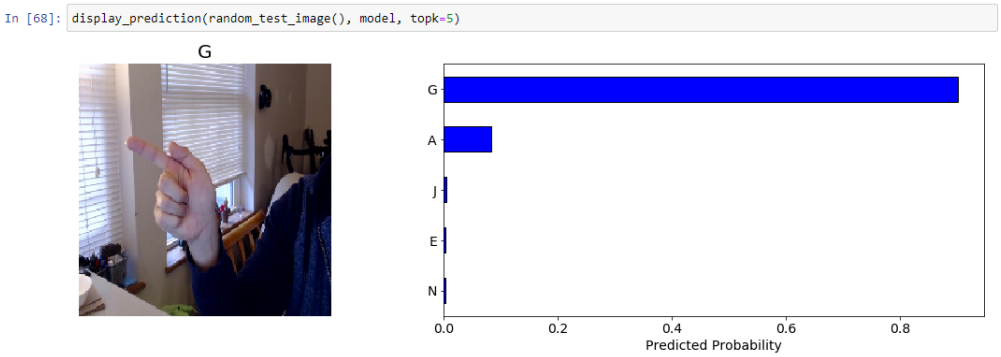
During test time, each model predicted a set of test images that were self-recorded. The test accuracies for each model are shown below in Table 1.

	Model 1	**Model 2**
Test accuracy	35.42%	95.26%

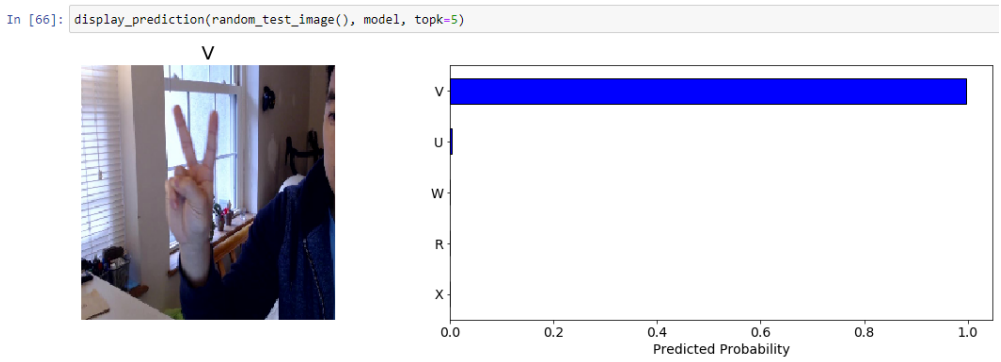
Table 2: Accuracies of Model 1 and Model 2 tested on self-recorded images

Examples of the test images are shown in Fig. 6 along with their ground truth labels (above the image) and the softmax probabilities in the bar chart immediately to the right of the images. During actual video captioning, the class with the highest p value is selected to be displayed on the frame.

The live video captioning was run on a Mid 2015 MacBook Pro with a 2.8 GHz Intel Core i7 processor and 16GB of 1600 MHz DDR3 RAM. The main bottleneck was the image cropping based on hand detection and as a result, the live video could only run at around 4-5 frames per second. Fig. 7 shows a few examples of the live video captioning with hand detection and classification. The predicted class (alphabet character) is shown in red in the lower left hand corner of each frame. The test accuracy on the live video was not quantified, but when Model 1 was used, the results similar to when we used the self-recorded data as test. Further, if we switch the model to Model 2, it performed very well in the same background conditions but performance dropped significantly in new environments.

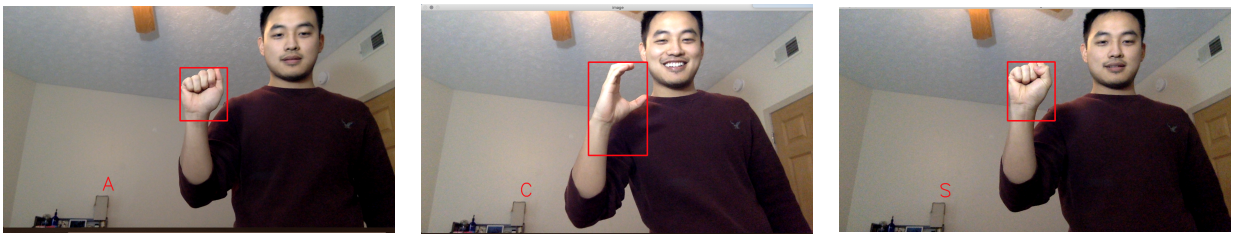


(a)



(b)

Figure 6: Test images with ground truth label (left) and softmax probabilities (right) for letter (a) G and (b) V.



(a) A (b) C (c) S

Figure 7: Demonstration of live video captioning system on letter (a) A, (b) C, and (c) S.

An alternative to live captioning is also offered. The user can upload a video file to the root directory and process each frame with the provided Jupyter notebook. Then, each frame is classified, labeled, and put together into a .gif file. An example output is given in the source code under `output.gif`.

4 Discussion and Conclusions

4.1 Significance of Results

A test accuracy of 95% was achieved with Model 2 on self-recorded test data. However, the accuracy of Model 1 on the same test data was only 35%. This is because the training set for Model 2 was much more similar to the test data, whereas the training set for Model 1 did not have similar backgrounds or lighting.

This is a problem of overfitting. However, the overfit model allows for the model to classify ASL fingerspelling signs with high accuracy given the test data is similar to the training data. Without it, a more generalized model does not perform nearly as well (35% accuracy). Techniques such as dropout and data augmentation were used during training to combat overfitting but the first model still struggled to reliably predict self-recorded signs. Many ASL signs are

similar to each other such as **G** and **H**, and **M** and **N**. This imposes a high level of difficulty in classifying ASL signs, compared to ImageNet where the classes have more obvious unique features.

4.2 Possible Improvements

Ideally, a Faster R-CNN network should be used for hand detection and classification in a singular model. However, we did not find any publicly available datasets that included ground truth bounding boxes along with ASL fingerspelling labels. Therefore, in this study, the model was split into two. One was only for hand detection and cropping and the other for ASL classification. If they were combined, then the model could learn how to crop and classify more reliably and with higher accuracy. Also, the inclusion of `SPACE`, `NOTHING`, and `DEL` could be useful for live image captioning.

To overcome the overfitting issue, the training dataset could have more diverse background and lighting conditions. Also, a model trained on a different dataset could be compared with the pretrained models on ImageNet. Transfer learning benefits when the pretrained dataset is similar to the training dataset. Therefore, if the pretrained dataset had hands as its primary classes, this model would likely experience higher performance.

4.3 Conclusion

A hybrid model of hand detection network and ASL classification with a transfer learning network was able to achieve over 95% accuracy on a self-recorded dataset. This is enough for intelligible video captioning from first-hand experience from the results. Additional work should be done to reduce overfitting the training data so that the results can be generalized to more backgrounds and lighting conditions.

There are many current avenues of research in this area of ASL translation including using 3-D depth data, wearable gloves, and use of Microsoft Kinect. The ability to use simple 3 channel RGB images for captioning is very valuable, however. Existing videos on YouTube made by the deaf community can be captioned retroactively for the hearing community. However, ASL fingerspelling is only a small fraction of the American Sign Language itself. It is comprised of many gestures for full words and depends on small body language shifts and facial expressions. This work is a small step towards the lofty goal of useful ASL translation forthe deaf community.

5 Contribution of Team Members

1. Team Member 1 - Michael Wang
   Contribution: Completed ASL classification through the use of transfer learning in Pytorch. Recorded and labeled custom dataset, and wrote Jupyter notebook to caption video from file.
2. Team Member 2 - Shuzhan Sun
   Contribution: Object detection network, part of live stream code, part of dataset augmentation.

References

[1] Martha E Tyrone and Claude E Mauk. The phonetics of head and body movement in the realization of american sign language signs. *Phonetica*, 73(2):120–140, 2016.

[2] Razieh Rastgoo, Kourosh Kiani, and Sergio Escalera. Multi-modal deep hand sign language recognition in still images using restricted boltzmann machine. *Entropy*, 20(11):809, 2018.

[3] Angela Caliwag, Stephen Ryan Angsanto, and Wansu Lim. Korean sign language translation using machine learning. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 826–828. IEEE, 2018.

[4] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. Faceboxes: A cpu real-time face detector with high accuracy. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9. IEEE, 2017.

[5] A comprehensive hands on guide to transfer learning. Accessed: 2019-11-23.

[6] Transfer learning tutorial - pytorch.