SP 2017 B657 - Computer Vision Project Report on

Fingerspelling Recognition for American Sign Language

By Chitesh Tewani, Pralhad Sapre, Rutuja Kulkarni
Guided by David Crandall

Introduction

Sign language acts as a bridge in communication for people who are deaf or mute. In 2011, it was found that approximately 500,000 people used sign language as their primary language of communication. Sign languages are considered to be independent languages and most of the population deprived of hearing sense is unable to understand the spoken language which is used by broader population. In this project, we aim to create a interpreter system that recognizes fingerspelling to English language. This will aid the process of communication with Sign Language practitioner through hand gestures. The aim is to provide a convenient interpreter to sign language practitioners which will assist them in their day to day activities.

Previous work on sign language recognition for fingerspelling and gestures could be done using a glove-based method and vision-based method[7]. In a glove based method, the user is required to wear hand glove which has electrical setting to connect to the device. Whereas, computer vision based technique involves hand detection and then interpreting what the character or gestures is after hand is detected. Research has shown that the key features to recognize hand gesture are shape of the hand, location, motion and orientation. We plan to use various Computer Vision techniques and restrict to identify two key features- hand shape and orientation in our project and the culminate the findings from our experiments.

In order to detect the hand region from image or live video, we used template matching and then applied deep learning techniques like convolutional neural network. We have trained the neural network on both dark and light background conditions using two different datasets.

Background and Related work

Notable research related to hand tracking and hand shape recognition includes Jacobs *et al.* [1] to utilise deep learning techniques, specifically convolutional neural networks, to recognise a set of hand shapes in various orientations within a live video stream captured on an iPhone mobile device. Jacobs *et al.* [1] extracts hand segments through procedure of template matching, skin color extraction from face detection, CAMshift for hand tracking and background subtraction using Gaussian Mixture Model. The orientation of the extracted hand segments is detected using an single CNN classifier which is passed to CNN-based hand shape classifiers. The research showed an average accuracy of 92% on hand orientation classifier and average accuracy of 89% for hand shape classifier.

Li *et al.* [2] and Foster *et al.* [3] also conducted research in hand tracking and hand shape recognition. Li *et al.* [2] system extracted hand shape features from a stream of web camera images and recognised the hand shape by means of a support vector machine (SVM) classifier. It was shown to classify hand shape accurately even with variations in skin tone, body dimensions and gender of test subjects, and despite background clutter. The system was able to recognise hand shapes with an average accuracy of 83.3%.

Foster *et al.* [3] uses artificial neural networks (ANNs) and random forests (RFs) techniques for hand shape recognition. The research showed that the ANN outperformed both the SVM and RF at 85.9% average accuracy.

We also surveyed the techniques developed by Mittal et al. [5] in their paper on detecting hands in an image. In the paper they present three complementary detectors to propose hand bounding boxes. The three proposal methods are shape, context, and skin colour. Each of these delivers a number of hypotheses for a bounding box for the hand, specified as a rotated rectangle (i.e. it is not axis aligned).

1. The shape detector relies on HOG features for the hand.
2. In order to learn the context in the context detector, a part based deformable model is trained from the hand bounding box training annotations extended to include the surrounding region of hand viz. wrist, forearm.
3. The skin detection results are further enhanced by using detected face regions (using techniques like Viola Jones) to determine the colour of skin in the target image. This enables skin to be detected despite unusual lighting and colour balances which may be peculiar to the target image.

The bounding box is then scored independently by a confidence evaluation function for each technique. These confidence scores yield a vector of 3 values <x1, x2, x3> which is then given to a trained SVM (using a similar scheme) to detect true positives. They have also mentioned a big dataset on which they trained their models which might be useful for our work.

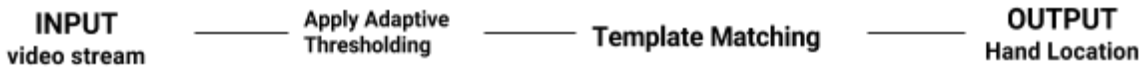
For gesture detection and real time tracking we found the work by Chen et. al [6] to be interesting. They introduce a hand gesture recognition system to recognize continuous gesture before stationary background. The system consists of four modules: a real time hand tracking and extraction, feature extraction, hidden Markov model (HMM) training, and gesture recognition. Specifically their approach to hand tracking and extraction works in near real-time using difference between successive video frames with thresholding. They achieved an average accuracy of 90%.

Hand Tracking and Hand Segmentation

In order to train and test with the Massey University's Sign language Dataset (dark background), we had to extract the hand image from every frame and convert it into a skin-pixel image. We have developed a robust hand tracking method which makes use of the initial frames to detect hand and uses the motion of the hand to track it.

Template Matching

Once the video starts inputting frames to the system, we apply template matching algorithm to detect initial set of coordinates of the region of the hand in the image. The number of templates use were restricted to two, because template matching is a computationally expensive process. Applying template matching only on the initial set of frames actually improved the time complexity drastically.

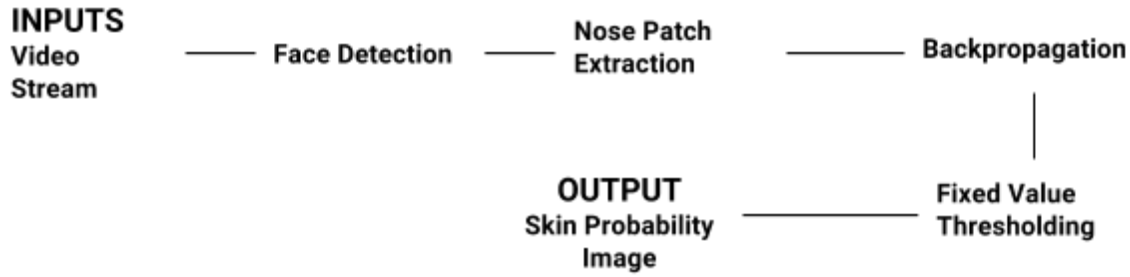


Extraction of Skin Image

The process of obtaining skin image involves detecting only skin pixels in the image. This is an important step as it was observed that extracting only skin pixels had an impact on the learning and error rate of the convolution neural network.

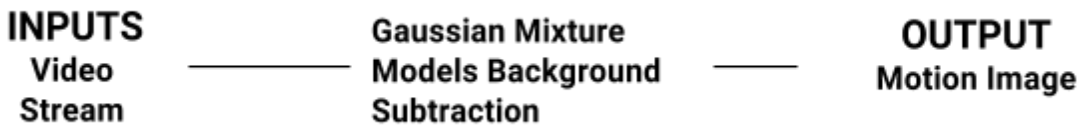
Face Detection

In order to obtain skin pixel image, it is necessary to determine the skin color texture of the person. We have used Viola-Jones face detection algorithm to determine the facial area of the person in the frame. We extract nose patch from this facial area and create a histogram as suggested by Jacobs *et al.* [1]. We assume that this histogram is a near representation of the skin pixels of the person. We use backpropagation technique to output a binary skin pixel image.



Hand Tracking
After the skin-pixel image is obtained, we use the initial coordinates of the hand obtained using template matching to trace the motion of the hand using CAMShift[8] algorithm. Unlike Mean-Shift algorithm, CAMShift algorithm not only tracks the hand but also adapts hand size as the person in the frame performs various fingerspelling.

Background Subtraction
In this step we obtain a background subtracted image using Gaussian Mixture Model thus helping in reducing noise.



Extraction of Hand Image
Finally, we combine the obtained skin-pixel image and background subtracted image by performing a logical AND operation over it and get the extracted hand image. This extracted hand image is then trained using neural net. In Case of Sign language dataset, we input the neural net with output of CAMShift step.

The process of hand tracking and hand segmentation for a real-time video can be seen as -

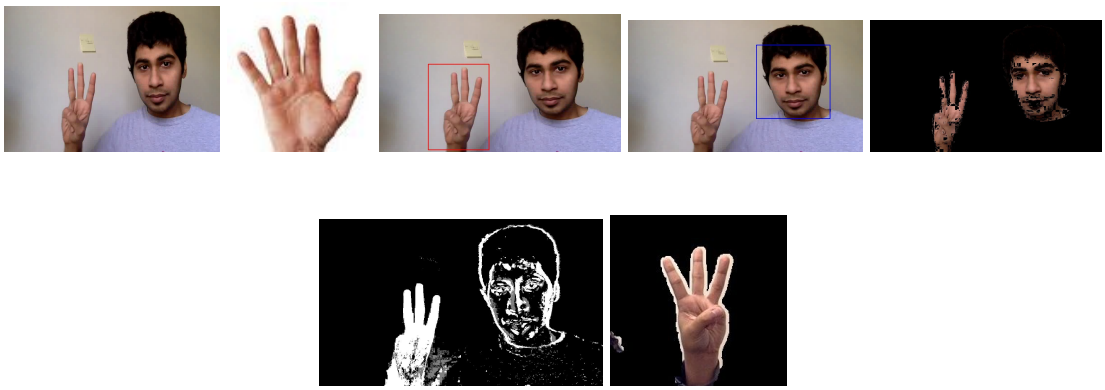


Fig 1. *(l-r)* Original Frame, Hand Template, Hand Location, Face Detection, Skin Probability Image, Back Subtraction, Extracted Hand Location

Hand classification

Once hand region has been segmented we focus on classifying the fingerspelling. Classification of the fingerspelling is done by us using two different approaches

1. Extract the hand skin pixel region by hand segmentation
2. Extract the hand region from the video frame using the bounding box given by CAMShift

For each of these two approaches we use a CNN. Before we dive into the details of how exactly we have used them, we present an intuitive overview to better understand the material to follow.

CNNs [9]
Convolutional Neural Networks (**ConvNets** or **CNNs**) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars.

LeNet was one of the very first convolutional neural networks which helped propel the field of Deep Learning. This pioneering work by Yann LeCun was named LeNet5 after many previous successful iterations since the year 1988. At that time the LeNet architecture was used mainly for character recognition tasks such as reading zip codes, digits, etc. It had 4 main operations

1. Convolution
   ConvNets derive their name from the “convolution” operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. In CNN terminology, the 3×3 matrix is called a ‘**filter**‘ or ‘kernel’ or ‘feature detector’ and the matrix formed by sliding the filter over the image and computing the dot product is called the ‘Convolved Feature’ or ‘Activation Map’ or the ‘**Feature Map**‘.

2. Non Linearity (ReLU)
   An additional operation called ReLU is used after every Convolution operation. ReLU stands for Rectified Linear Unit and is a non-linear operation. ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

3. Pooling or Sub Sampling
   Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

4. Classification (Fully Connected Layer)
   The Convolution + Pooling layers act as Feature Extractors from the input image while Fully Connected layer acts as a classifier. The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer (other classifiers like SVM can also be used). The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer.

CNN + SVM approach

From the above discussion it is clear that CNN’s extracts features in the initial layers and deals with actual classification in the final layers. We first started our classification by using publicly available trained CNNs for extracting features from the images. We used the following CNNs

- OverFeat from ILSVRC 2013
- GoogLeNet from ILSVRC 2014

These two CNNs were the winners and the state of the art in the ILSVRC competition in the years 2013 and 2014. OverFeat gave us features of length 4096 and GoogLeNet output features of length 1024. On these features we trained an SVM for the 26 signs of the English alphabet. This trained model was then validated on the dataset. Here are the results we attained.

Dataset	Feature Extractor	Classifier	Accuracy
Massey	GoogLeNet	SVM	49%
Massey	OverFeat	SVM	42%

Transfer Learning

In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet as an initialization for fine tuning to a specific task [10].

We adopted this approach since the two CNNs we used above as feature extractors were trained on a different problem than ours. The deep learning framework Caffe also has a model trained on the sample ImageNet challenge. Its called CaffeNet. It was this model that we used as a starting point for the transfer learning to our specific problem. Caffe CNNs have their detailed layer setup in the file `train_val.prototxt` and the control parameters in the file `solver.prototxt`. We modified the last layer in the `train_val.prototxt` to output just 26 values, corresponding to the English alphabet. Here is the section which was changed

```
inner_product_param {
    num_output: 26
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
```

```
      value: 0
    }
  }
```

We also modified the control parameters in `solver.prototxt` to the following values before commencing the transfer learning. Note that these setting were used when transfer learning on Sign Language Recognition Dataset [11].

```
net: "models/caffenet_slds/train_val.prototxt"
test_iter: 100
test_interval: 1000
# lr for fine-tuning should be lower than when starting from scratch
base_lr: 0.0001
lr_policy: "step"
gamma: 0.1
# stepsize should also be lower, as we're closer to being done
stepsize: 5000
display: 20
max_iter: 2000
momentum: 0.9
weight_decay: 0.005
snapshot: 500
snapshot_prefix: "models/caffenet_slds/caffenet_slds"
# uncomment the following to default to CPU mode solving
solver_mode: GPU
```

Also note that we did not prohibit the learning of any layer before the last one, that way minor changes did happen on the earlier layers as well. Big Red 2 and CUDA enabled laptop were used to train these models. In a time span of 30 minutes the transfer learning was completed on the CUDA enabled laptop running Ubuntu. This model gave us one of the best results on the live video stream data.

Results

Here are the validation results we obtained for the different approaches we tried

Dataset	Feature Extractor	Classifier	Accuracy
Massey	GoogLeNet	SVM	49%
Massey	OverFeat	SVM	42%
Massey	CaffeNet	CaffeNet (transfer learning)	77.3%

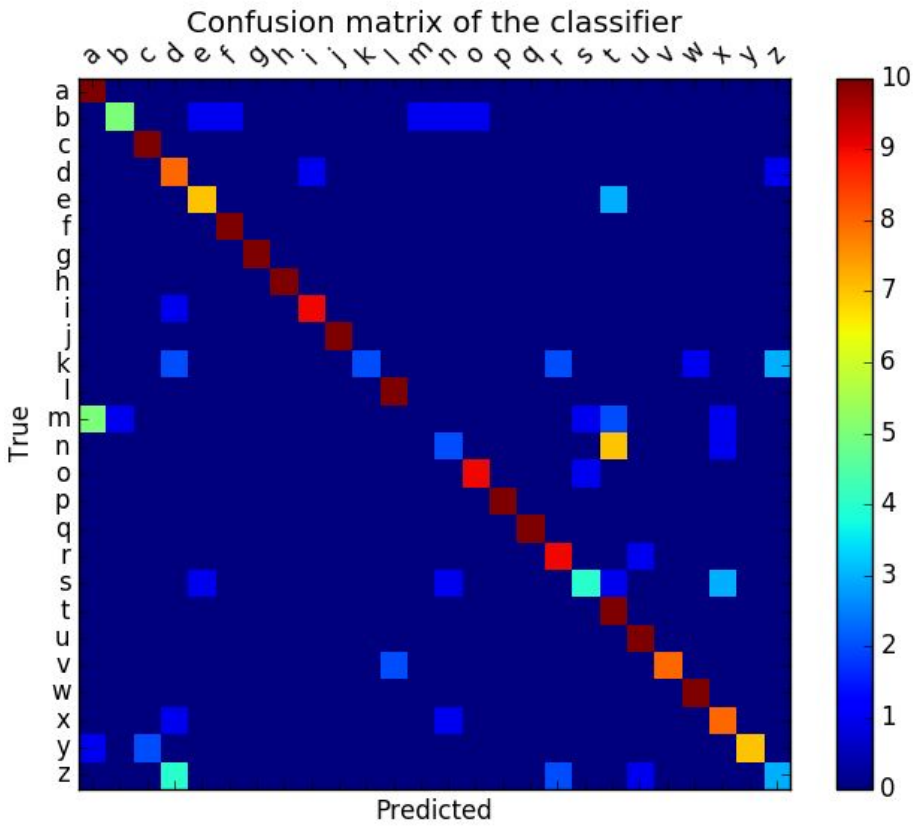


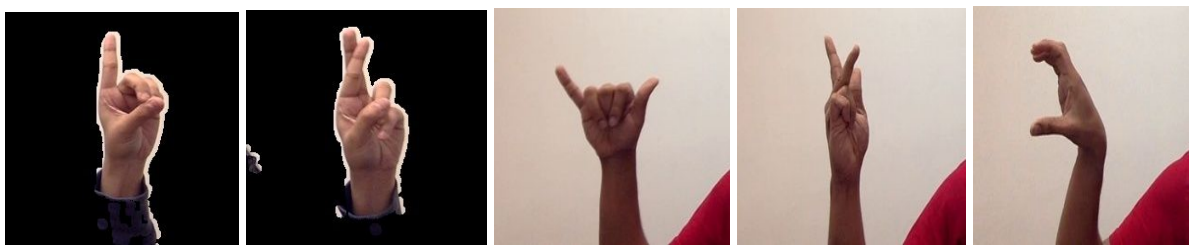
Fig 2. Confusion matrix when performing validation

Clearly transfer learning was giving us the best results in model validation. This model was thus used when testing on live video. As mentioned before we had models trained on the two datasets. One model expected a hand segmented image and the other the hand region of interest.

We observed that the hand segmentation approach gave worse results than the hand region of interest approach because of the inability to perfectly segment the hand region out of a video frame.

When counting the accuracy, the notion of accuracy was relaxed in the sense that if the ground truth was correctly predicted in the top 5 labels it was counted as a hit.

Here are some example images which were **correctly predicted** with the top 5 predictions



L T **D** V R L C X P **R** V **K** L Y D L **Y** V W U L **C** X P R

Fig 3. Letter in bold blue is the predicted output

And here are some example images where the labels were **not correctly predicted**

S M V X W G H T A X

Fig 4. Misclassified images

On a live video using transfer learning for the hand region of interest patch we obtained an **accuracy of 65.07%**. Here we have used to relaxed notion of accuracy. The rationale behind this relaxed metric and the inherent difficulty of the problem are presented in the next section. Our trained caffe model is available at [12].

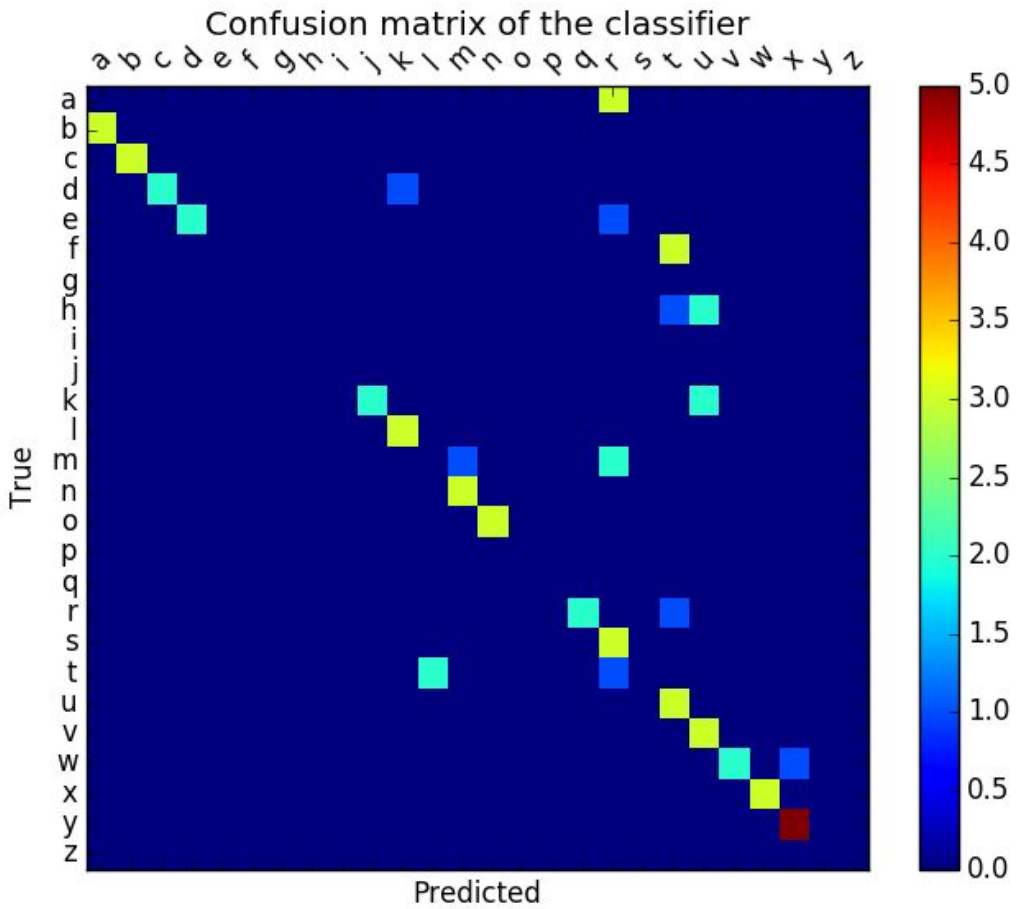


Fig 5. Confusion matrix on live video
(Video did not contain signs for all characters)

Discussion

In the fingerspelling recognition task an important point to note is the difficulty of clearly distinguishing between the different signs. Some signs are really similar in appearance to each other and even a human might have some difficulty sometimes to figure out the exact differences. Here are some examples

Fig 6. ASL Signs for M, N, T and S

Here the differences might seem obvious since this is a clean high resolution curated image from the Massey dataset. Now the problem gets harder when there is a varied background, maybe something which resembles the skin color or the distance from the camera is longer making these differences less obvious. To top it off there is the issue of illumination too.

To tackle this problem we had another dataset from [11] where images had the background present. The reason models trained on this dataset worked well was that our test locations in IU buildings had a similar setting too. What could also be done here is data augmentation - where these clean images from the Massey dataset could be infused with various kinds of noise to make the CNN robust and ready for real world scenarios.

Our application of hand segmentation could also be further refined as an alternative to this technique. But for now we were not able to get extremely clean cut outputs from this processing. Here is an example of the output of extraction versus the type of images present in the dataset.

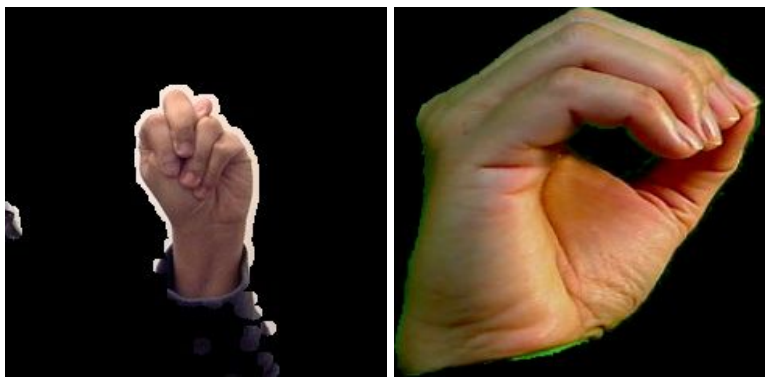


Fig 7. Extracted image and hand image in dataset

These are some of the directions where more work could be put in for extracting better results from the existing pipeline.

For now we also expect the user of the system to hold up his hand for the template matching and CAMShift to lock in on the hand before making any signs. The letters J and Z in ASL are actually gestures which are hard to deduce through just fingerspelling. That is another area where the problem of fingerspelling detection is not just restricted to sign detection but rather has a temporal aspect too.

Conclusion

We implemented and trained an American Sign Language translator by real-time video processing and a CNN classifier. We were able to produce a robust model to classify 16 out of 26 letters accurately. Due to lack of variation in the datasets, testing on real-time video did not reach near our validation accuracies. The project can have a potential impact on everyday users to better communicate with american sign language practitioners and bridge the gap between the two communities. This project could be ported as a real-time video chat application through handheld devices, portable or laptops.

Future Work

Additional Models: We focused our efforts on optimizing GoogLeNet, but it would be worth exploring different nets that have also been proven effective at image classification (e.g. a VGG or a ResNet architecture).

Data Augmentation: We believe that the classification model was trained only on standard hand images of a single orientation. However, while processing real-time video, the hand orientation might be of a different orientation. We can train the CNN model for different orientation of hand-images or adding some background noise.

Language Model Enhancement: Building a bigram and trigram language model would allow us to handle sentences instead of individual words. Along with this comes a need for better letter segmentation and a more seamless process for retrieving images from the user at a higher rate

References

1. Kurt Jacobs, Mehrdad Ghasiazgar, Isabella Venter, Reg Dodds “Hand Gesture Recognition of Hand Shapes in Varied Orientations using Deep Learning” ACM Digital Library, 2016

2. P. Li, M. Ghaziasgar, and J. Connan. “Hand shape recognition and estimation for South African sign language.” In South African Telecommunication Networks and Applications Conference, pages 344–349, 2011.

3. R. Foster, M. Ghaziasgar, and J. Connan. “A comparison of machine learning techniques for hand shape recognition”,. In Proc. South African Telecommunication Networks and Applications Conference ’13, pages 441–442, 2013

4. Barczak, A. L. C., Reyes, N. H., Abastillas, M., Piccio, A., Susnjak, T. “A new 2D static hand gesture colour image dataset for ASL gestures”, Research Letters in the information and Mathematical Sciences, 2011

5. A. Mittal, A. Zisserman, P. H. S. Torr “Hand detection using multiple proposals”, British Machine Vision Conference, 2011

6. Feng-Sheng Chen, Chih-Ming Fu and Chung-Lin Huang “Hand gesture recognition using a real-time tracking method and hidden Markov models” Image and Vision Computing, 2003

7. W. C. Stokoe. Sign language structure: An outline of the visual communication systems of the American deaf. studies in linguistics: Occasional papers. Buffalo: Dept. of Anthropology and Linguistics, University of Bualo, Tech. Rep. 8, 1960.

8. OpenCV documentation - Meanshift and CAMShift
http://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html

9. Convolution Neural Networks Intuitive explanation
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

10. http://cs231n.github.io/transfer-learning/

11. Sign Language Recognition Dataset by Anmol Singh Jaggi
https://github.com/Anmol-Singh-Jaggi/Sign-Language-Recognition

12. Trained Caffe Model
https://drive.google.com/open?id=0B3f0p3K1My8LQnY2M1B2aFlESzg