SIGNER-INDEPENDENT RECOGNITION OF STATIC ASL SIGNS

by

MELISSA ABRAMSON

Advisor
JOSHUA ECKROTH

A senior research paper submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science
in the Department of Mathematics and Computer Science
in the College of Arts and Science
at Stetson University
DeLand, Florida

Spring Term
2016
ACKNOWLEDGMENTS

I wanted to take a few moments and acknowledge some of the people who have made this paper and research project a success. First and foremost, thank you Dr. Eckroth for putting up with my crazy nonsense and questions this past year. On a project which I had no preexisting background for, you helped me learn and accomplish a tremendous amount in the machine learning world. Your active involvement in my project and encouragement when things did not go as planned helped push me to new bounds in my research, and really helped me get a grip on what it means to be resilient in the face of new challenges. I also want to thank my other computer science professors Dr. Plante, Dr. ElAarag, and Dr. Branton for helping get me to this place the last four years. If it hadn’t been for your wonderful leadership and direction I probably wouldn’t be a computer scientist, or have the awesome job I have! You challenged the boundaries of my thinking, helped me to recognize that I am much more intelligent than I previously believed, invested in my life, and helped shape me into the person I am today. There are boundless others who I also owe thanks to for emotional support and encouragement. Most of all I thank Jesus, because I can do ALL things through Christ who strengthens me! 😊
# TABLE OF CONTENTS

ACKNOWLEDGMENTS ........................................................................................................ ii

TABLE OF CONTENTS ...................................................................................................... iii

LIST OF FIGURES ............................................................................................................. v

ABSTRACT ........................................................................................................................ 1

1. INTRODUCTION .................................................................................................. 2

2. RELATED WORK ................................................................................................. 5

2.1 How Does ASL Compare with Spoken English Language? ................................. 5

2.2 Overview of Hand Gesture Recognition Methods ................................................... 7

2.3 Two Categories of Hand Gesture Recognition ........................................................ 8

2.4 Datasets ................................................................................................................. 9

2.5 Haar Cascades ....................................................................................................... 9

2.6 Neural Networks .................................................................................................... 11

2.7 Deep Learning ..................................................................................................... 12

2.8 Caffe .................................................................................................................... 14

2.9 Convolutional Neural Networks (CNNs) and Caffe’s ImageNet Model ............... 17

3. IMPLEMENTATION ........................................................................................... 19

3.1 Stepping into Caffe ............................................................................................. 27
LIST OF FIGURES

Figure 1: ASL Alphabet [17] ........................................................................................................ 7
Figure 2: Haar Cascade Features ..................................................................................... 11
Figure 3: GoogLeNet Object Classification Results [7] ..................................................... 15
Figure 4: An MNIST digit classification example of a Caffe network, where blue boxes represent layers and yellow octagons represent data blobs produced by or fed into the layers [11]. ........................................................................................................ 16
Figure 5: Example Sign Image Capture “I Love You” ...................................................... 19
Figure 6: Small Survey of the Dataset Images .................................................................... 21
Figure 7: Varied Orientations of the Signed Letter ‘P’ .................................................... 21
Figure 8: Data Classification Program Interface .......................................................... 22
Figure 9: False Positives Using Haar Cascades ............................................................. 23
Figure 10: Sample of Negative Binary Images ............................................................... 24
Figure 11: Example of Median Skin Tone Retrieval [29] ................................................ 25
Figure 12: Process of Determining Median Skin Tone [29] ............................................ 26
Figure 13: Final Hand Detection Results [29] .................................................................. 26
Figure 14: 256x256 Resized Image Example .................................................................. 27
ABSTRACT

As the Deaf community continues to grow [1], so does the market for tools that aid communication between the Deaf and hearing communities. One such tool is software for automatically recognizing ASL signs that could assist in learning and communicating in ASL. We have investigated the development of such a system that recognizes 28 static ASL signs. We have also developed several tools for data collection, modification, and classification. We found several common machine learning techniques that are not ideal to use for ASL recognition. We also explored the use of Caffe, a deep learning framework using convolutional neural networks. Due to hardware limitations and time constraints, we were not able to complete the training using Caffe and thus are unable to confirm if this is the ideal approach.
1. INTRODUCTION

American Sign Language (ASL) is a complex visual-spatial language that is used by the Deaf community in the United States [15]. To clarify terminology, deaf (with a lowercase “d”) refers to the partial or complete physical inability to hear, whereas Deaf (with a capital “D”) refers to embracing the cultural norms, beliefs, and values of the Deaf Community [17]. The World Health Organization (WHO) estimates that there are more than 360 million people worldwide with a disabling hearing loss, including 32 million children and one-third of individuals over 65 years of age [9]. Individuals can incur hearing loss as a result of genetic causes, complications during childbirth, certain infectious diseases, chronic ear infections, exposure to excessive noise, and aging. Since spoken language development is often delayed in children with deafness, hearing loss can be detrimental to an individual’s ability to communicate with others [27]. The use of technology has the ability to significantly increase opportunities for deaf individuals to communicate with those who have hearing ability as well as improve their overall quality of life.

According to a study done by the Pew Research Center, over 64% of American adults own a smartphone as of October 2014, and as of January 2014, 90% of American adults own a cell phone, 32% of American adults own an e-reader, and 42% of American adults own a tablet computer [22]. This indicates that for Americans, technology can be found at almost everyone’s fingertips. With smart phones and tablets readily available in
classrooms and in individual possession, the use of technology for learning and assisting
with communication between the Deaf and hearing world is both plausible and sought
after.

Currently, when a hearing person wants to talk with a Deaf person, there is a
language barrier that has to be overcome. This barrier can be resolved if the hearing
person learns ASL or, if the Deaf person knows English, through written means. The
only problem is that no matter what method of communication is used, at least one person
will not be able to speak in their native language. This process can be much more time
consuming and difficult than speaking to someone in your native language [3].

As technology continues to develop we find there are applications for smart
phones such as Skype and FaceTime that allow Deaf individuals to video chat over long
distances. These applications can help communication amongst Deaf individuals. It does
not, however, solve the Deaf to hearing communication issue. There exist online
dictionaries such as Signing Savvy and Lifeprint that allow one to search an English term
and receive a video of the corresponding sign. Reference sources also exist which allow
users to access ASL sign definitions without previously being aware of the English
definition. For example, the Hand Shape Dictionary allows users to look up signs based
on hand shape and other motion features [28]. It is a resource used in classrooms settings,
however, in practical real-life situations it is difficult to look up a word in this manner.
There are currently no available applications for real-time ASL gesture recognition. A
company called MotionSaavy is in the process of developing a system called the UNI that
would provide ASL to text translations [20]; however, the product as advertised is
expensive which may prevent it from being used by the majority of the population. This
research paper addresses the first steps needed to create what could ultimately be a product for doing real-time ASL sign recognition, and helps future researchers be able to continue the work towards a goal that would help solve this communication barrier present in the world today.

The remainder of this paper is organized as follows: Section 2 introduces background concepts of ASL and current methods of hand gesture recognition. Our implementation is described in Section 3. Section 4 describes the current results. Finally, the conclusion is described in Section 5, and future work is discussed in Section 6.
2. RELATED WORK

2.1 How Does ASL Compare with Spoken English Language?

In spoken languages, individuals utilize their mouth and voice to make specific sounds that are recognized as words. For deaf individuals, one does not have the ability to hear the sounds of speech, and very few speech sounds can be physically seen on the lips. Sign languages arose out of the need for deaf individuals to communicate through visual rather than audible means [21]. ASL is a separate and distinct language from English. All of the features that define a language including rules for pronunciation, complex grammar, and word order can all be found in ASL. Meaning and expression of ideas is also conveyed differently through sign language than through spoken language. Instead of using tone and pitch as a means of asking a question or expressing emotion, individuals using ASL will incorporate facial expression and body language to convey meaning. For example, to ask a question individuals might raise their eyebrows inquisitively as well as widen their eyes or tilt their bodies. Similarly to spoken languages, ASL also has variations in how individuals sign, much like regional accents and verbiage [21].

ASL has two distinct kinds of signs that are described as follows: Static signs (gestures) are signs without movement and are determined by a certain configuration of the hand. Examples of these are the letters in the ASL alphabet, with exception of J and Z. Dynamic (or non-static) signs (gestures) make up the majority of the signs used in
ASL. A dynamic sign is a moving gesture determined by a sequence of hand movements and configurations. Dynamic signs are also sometimes accompanied by body and facial expressions that can convey as much meaning as the hand posture [33].

Another aspect of ASL addresses how words are handled that do not have a predefined sign, this is accomplished through fingerspelling. Fingerspelling is an important component of ASL and a necessary skill for complete communication in sign language [24]. When fingerspelling, individuals use their dominant hand to create a series of manual symbols, each symbol corresponding to each letter of the word. Fingerspelling is used for spelling proper nouns, technical terms, acronyms, initialized signs, and words from foreign languages. It uses one hand and 26 gestures to communicate the 26 letters of the alphabet. The 26 alphabets of ASL are shown in Figure 1. An individual fluent in ASL can achieve a finger spelled word rate of four characters per second. In contrast, for people first learning sign language, the fingerspelling recognition rate is much lower [19]. Acquisition of fingerspelling recognition skills typically falls behind other sign language skills [19]. “Fingerspelling is the first skill learned and the last skill mastered” [16]. This tells us that the ability to learn to recognize fingerspelling as well as master fingerspelling is imperative for communication through ASL. The next sections describe different methods that have been tested to recognize these signs.
2.2 Overview of Hand Gesture Recognition Methods

There are several methods available for hand gesture recognition. Kulkarni and Lokhande provide a complete list of current methods for hand gesture recognition. There are two methods we will explore in this paper: Haar Cascades and Neural Networks. Haar Cascades were initially used for facial detection and are easily transferrable to hand gesture recognition [23]. Neural networks are used for solving more complex pattern recognition problems, such as computer vision problems where procedural algorithms are
difficult to generate [31]. The next few sections further explore these methods; describe the datasets these methods use, as well as mention the ways hand gesture recognition is accomplished in practice.

2.3 Two Categories of Hand Gesture Recognition

Attempts to automatically recognize sign languages first appeared in the 90s. Research on hand gestures can be classified into two categories:

- Glove-based systems, which use electromechanical devices to measure different gesture parameters such as hand position, location of the fingertips, angle, direction of movement, etc. [33]. This method can be problematic to use outside of a research laboratory setting due to the fact it requires extra hardware. Most people do not own electromagnetic gloves, nor are gloves generally worn in ASL conversation. Although they are more accurate, glove-based systems are not appropriate in practical applications.

- Visual-based systems use machine vision and image processing techniques to recognize hand gestures. These systems can further be divided into two more categories. The first one relies on using gloves specially designed with visual input markers called “visual-based gesture with glove-markers (VBGwGM)”. For example, this can include gloves with colored dots or sections on the fingers where joints are located as well as the front and back of the hands, which help in determining hand postures. Much like glove-based systems, this method can also be problematic because again it is not natural or realistic for hand gesture recognition. The second
category is a visual-based gesture recognition system called “pure visual-based gesture (PVBG)”, a system that does not use glove-markers [33]. This uses bare hands to recognize gestures the most natural approach.

2.4 Datasets

The methods of hand gesture recognition we explore all require large datasets for training. Datasets for hand gesture recognition consist of images or videos of the signs to be recognized. For signer dependent datasets the images or videos will consist of one individual signing a letter, word, or phrase. These images or videos tend to be created in controlled settings with solid and distinctly colored clothing and backgrounds, consistent lighting, and high quality camera resolution. For signer independent datasets, the images or videos tend to incorporate different individuals, a variety of skin tones, and alternative orientations of signing a letter, word, or phrase. They also are not restricted to controlled settings, meaning that many of the images and videos may contain diverse backgrounds, camera resolutions, camera quality, lighting conditions, etc. We expect that signer independent datasets produce models that are better able to recognize signers not represented in the dataset. However, signer independent datasets typically include a wider variety of features such as skin color and background clutter, which make the training of an accurate model more difficult. Signer independent datasets therefore need to be more extensive to represent a wide variety of situations. We have not found a public dataset that is sufficient for our purposes, thus as explained in Section 3, we have developed our own dataset.

2.5 Haar Cascades

Object detection using Haar feature-based cascade classifiers was found to be an
effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features". It is a machine learning approach where a cascade function is trained from a multitude of positive and negative images. It can then be used to detect objects in other images [10]. Initially, the algorithm was used for facial recognition. It needed a lot of positive images (e.g. images of faces) and negative images (e.g. images without faces) to train the classifier. Once the classifier was trained, features could then be extracted. Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from sum of pixels under the black rectangle [10] an example of what this looks like is in Figure 2. Many of the features calculated by this method are outside of the scope of the object being detected. The use of cascade boosts is what helps narrow the scope to just the desired features. Adaboost for example, finds the best threshold to classify an object as positive or negative. Since there is a good chance of errors, the user selects the features with the lowest error rate (meaning those are the features that best classify the object and non-object images). After each classification, the weight of misclassified images is increased. The process is then repeated, new error rates are calculated, and new weights are calculated, until a certain required accuracy rate is achieved, or the required number of features is found. Once that is completed, a user is now able to take an image, apply however many features he/she has created to that image, and check if the object he/she is looking for is in that image [10].
Cascade Classifiers were introduced to help the program disregard regions of an image that are unlikely to have the object being recognized. Instead of applying all the features on the entire image, the program groups the features into different stages of classifiers and applies each group one-by-one. If a region fails the first stage, the program moves on to a different region (and doesn’t apply anymore feature groups on that region). If the region passes, the program applies the second stage of features and continues the process until a region is found that passes all of the feature groups [10]. Employing this method saves time by only evaluating the regions of an image that are likely to possess the object being detected, in our case, a hand.

Chen, et al. proposed [23] a two level approach to recognize hand gestures in real-time using Haar-like features to effectively described the hand posture pattern, and the AdaBoost learning algorithm to efficiently speed up the performance speed and construct a strong classifier. A parallel cascade structure was implemented to classify different hand postures. The experiment results found this structure could achieve satisfactory real-time performance, as well as classification accuracy above 90% [23]. This project was just one example of hand gesture recognition.
2.6 Neural Networks

A neural network is a computer program that ‘‘learns by example’’. ‘‘Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques’’ [6]. Neural networks have proven efficient for classification and are able to reach a 95% successful recognition rate [31]. The difficulty that arises with Neural Networks is an extensive amount of processing and time is required to obtain that success rate.

In 1993, a hand gesture to speech system was developed where capture was done using a VPL Data Glove connected to a DECtalk speech synthesizer through five neural networks [30]. This system was able to recognize a hand shape, and then based on which direction (out of six) the hand could move, added an ending to the word. The five networks they used were strobe time, root word, ending, rate, and stress. Unfortunately this system only recognized gestures, not ASL vocabulary. By 1999, the CyberGlove was being used to capture signs and then process them using the Stuttgart Neural Network Simulator, which is a free software package [8]. The limitation with this was that only segmented words could be recognized [5]. A more recent study done in 2009 resulted in a gesture recognition system that was proved to be robust for the Mayanmar Alphabet Language (MAL) gestures. The system developed worked in real time and was fully automatic. A user did not need a glove or a uniform background to recognize the MAL gestures. The experiments carried out revealed up to 98% recognition accuracy [31].

2.7 Deep Learning

Deep learning is a more sophisticated application of neural networks that have
performed well in a variety of tasks. In order to understand how deep learning works, it is necessary to first understand how humans process information.

- Humans hierarchically organize their concepts and ideas.
- Humans learn simpler concepts before determining what represents more abstract concepts.

Similarly, an engineer’s goal is to separate solutions into multiple levels of processing and abstraction. However, it is much more difficult to express concepts by only analyzing certain inputs that may only cover a small fraction of all possible concepts [34].

As of yet, we do not have learning algorithms that can discover the many visual and semantic concepts that would seem to be necessary to interpret most images on the web [34]. However, in 2006 a breakthrough occurred in the world of machine learning and artificial intelligence. Hinton and collaborators at the University of Toronto introduced Deep Belief Networks (DBN) [13], with a learning algorithm that greedily trains one layer at a time, exploiting an unsupervised learning algorithm for each layer. More recently, other algorithms for deep architectures were proposed that use other methods with the same principal that have been found successful in various classification tasks [35]. Until 2006, deep architectures were minimally discussed in machine learning literature due to generalization errors and poor training results [34]. Convolutional Neural Networks (CNN) were discovered to be much easier to train and effective in image classification [18].

Compared to image classification, object detection is a more challenging task that requires more complex methods to solve. Due to this complexity, current approaches (e.g., [26], [14]) train models in multiple stages that are slow to converge [25].
Complexity arises because detection requires the accurate localization of objects. This creates two problems, the first being how numerous candidate object locations must be processed, and the second being how these candidates provide only rough localization that must be refined to achieve precise localization [25]. Further explanation on how neural networks work will be discussed later in this paper.

2.8 Caffe

Many researchers have looked at neural networks for object detection and recognition. Google has annually held a challenge that has shown promising results in the field of computer vision. “The ImageNet large-scale visual recognition challenge (ILSVRC) is the largest academic challenge in computer vision, held annually to test state-of-the-art technology in image understanding, both in the sense of recognizing objects in images and locating where they are.”[7]. The winners of the competition in 2014 were part of a team named GoogLeNet made up of 8 individuals, two of which were PhD students interning at Google while actively working on the project. They placed first in two of the three categories scored on, those being classification (the ability to accurately classify and label a detected object) and detection (the ability to accurately detect an object or multiple objects in an image). As can be seen in figure 3 they were able to identify multiple objects in complex images. This is important to note, because the improved neural network model they created is used in the R-CNN detector by Ross Girshick et al. used in the Caffe framework we chose to use.
Caffe is intended to provide developers and scientists with a modifiable framework for deep learning algorithms and reference models. Within the framework, Caffe uses Convolutional Neural Networks, or CNNs that are discriminatively trained via back-propagation through layers of convolutional filters and other operations such as pooling[11]. “Following the early success of digit classification in the 90’s, these models have recently surpassed all known methods for large-scale visual recognition, and have been adopted by industry heavyweights such as Google, Facebook, and Baidu for image understanding and search” [11]. How CNNs work and are incorporated in Caffe will be discussed further in section 2.9.

Caffe stores and communicates data through 4-dimensional arrays called blobs. These blobs serve as a unified memory interface that conceal both mental and computational overhead created by a mixed CPU/GPU environment by synchronizing between CPU and GPU hosts. Blobs not only hold image data, but also parameters and derivatives. The use of blobs allow low-level details to be ignored while still maintaining
a high level of performance [11]. Memory on the host device is allocated on demand for efficient memory usage [11]. Caffe layers are another essential part of the neural network layer. Each layer accepts one or more blobs as an input yielding one or more blobs as an output. Layers have two main responsibilities that impact the network as a whole. They serve as a forward pass taking inputs and producing outputs, and as a backward pass, back-propagating gradients with respect to the input parameters by the output gradients.

“Caffe provides a complete set of layer types including: convolution, pooling, inner products, nonlinearities like rectified linear and logistic, local response normalization, element wise operations, and losses like softmax and hinge,” all of which are types needed in these kinds of visual tasks [11]. Caffe models themselves are end-to-end machine learning systems that typically begin with a data layer loaded from disk, and ends with a loss layer that computes the objective for a task (classification for example). Figure 4 visualizes what a typical Caffe network would look like during training.

![Figure 4: An MNIST digit classification example of a Caffe network, where blue boxes represent layers and yellow octagons represent data blobs produced by or fed into the layers [11].](image)

Caffe processes data in mini-batches that pass through the network sequentially. Learning rate decay schedules, momentum, and snapshots for stopping and resuming are
vital for tasks such as object detection, because from a snapshot of an existing network and a model definition for a new network, Caffe can fine-tune old model weights for the new task and initialize new weights as needed [11].

2.9 Convolutional Neural Networks (CNN’s) and Caffe’s ImageNet Model

The ImageNet model Caffe primarily utilizes is comprised of a deep convolutional neural network that was found to be capable of producing “record-breaking” results on highly challenging datasets using purely supervised training [2]. “Compared to typical feed-forward neural networks with similarly sized layers, CNNs have fewer connections and parameters which make them easier to train” [2]. The main building block for CNN’s is the convolutional layer. The convolutional layer has parameters that consist of a set of learnable filters. Although these filters have a small receptive field, they extend through the full depth of the input volume [4]. With each forward pass, every filter is convolved across the width and height of the input volume, and the dot product between the entries of the filter and the input volume is computed, thus creating a 2-dimensional activation map of that filter [4]. This results in the network being able to learn filters that activate when they see a specific input feature. The full output of the convolution layer is achieved by stacking the activation maps for all filters along the depth dimension [4]. Every output volume entry is thus interpreted as a neuron output, which can look at a small region in the input, and can share parameters with other neurons in the same activation map [4].

The ImageNet model does not use the standard way to model a neurons output. Instead, it follows the research of Nair and Hinton, who used neurons with nonlinearity called Rectified Linear Units (ReLUs)[32]. It was found that deep CNNs with ReLUs
trained several times faster than the standard saturated neuron models. ReLUs are desirable because they do not require input normalization to prevent them from saturating [2]. Thus as long as some training examples produce positive input in the ReLU, learning will happen in that neuron.

Another variation the ImageNet model improves upon the classic CNN revolves around pooling layers. Pooling layers summarize the outputs of neighboring groups of neurons in the same activation map. Typically a CNN would make separate pooling layers, however the ImageNet model utilizes overlapping pooling layers [2]. The overlap was found to reduce error rates by making it more difficult for the outputs to overfit [2].
3. IMPLEMENTATION

For the machine learning techniques we previously discussed, one of the first obstacles we had to overcome was the acquisition of a large data training set. While there are a few online dictionaries for ASL whose data sets we can request to use, many of these datasets only have one example of a sign, and normally utilize one individual as the case example. This leads to signer dependence within the program. We wanted to tackle the issue of signer independence by utilizing a dataset that encompasses the static ASL signs being performed by multiple individuals with different skin tones, hands, backgrounds, attire, lighting conditions, and camera quality. As a result, it was necessary to create our own unique dataset for training the machine learning algorithms. We developed a program to collect both positive input photos of the signs, as well as negative input photos (for training fail cases). Figure 5 shows an example capture with the program.

![Figure 5: Example Sign Image Capture “I Love You”](image)

Our dataset contained images with variable conditions to help obtain signer independence. Figure 6 displays a survey of images from our dataset. Currently we

These conditions can further be broken into four categories: exposure, clutter, skin tone, and orientation. The exposure conditions of the images vary from dim and grainy to bright and clear. This accounts for the different lighting conditions as well as resulting camera quality that will be experienced in real-life situations. Several of the images also contain background clutter which can make gesture recognition more difficult since the program would have to differentiate the hand from say a bookcase or a chair that can blend in with the hand, or otherwise impact the accuracy of gesture recognition. Multiple individuals with different skin tones are also utilized. This accounts for variations amongst signers, as well as amongst skin tones. Figure 7 gives an example of the letter ‘P’ being signed from three different orientations, another feature of gesture recognition that needs to be considered since signs can look different from varying angles. There are also several images that incorporate multiple individuals (besides the signer), or have movement interference in the image, cases which can occur in practical application of hand gesture recognition. In situations where a user is in a public place instead of a controlled setting, hand gesture recognition programs need to be able to differentiate the important information (sign of the user), from the other distractions within the image.
We obtained a dataset with 1924 images and began the process of data classification. Having never done previous work in the area of gesture recognition, we decided to first try an approach proven to work by other researchers. Since OpenCV [12] already has several functions in place for using Haar Cascades on images, we decided to try implementing our own specialized Haar Classifiers for the recognition of hands. Figure 8 displays the program interface and how the images were marked up for classification.
Following the creation of a dataset of marked up images; we were able to fully train a Haar Cascade. However, even using various threshold sizes for what would be considered a hand in the image, there were still an overwhelming amount of false positives. The cascade appeared to confuse both the hands and the face as can be seen in figure 9. We attempted to remove the face from the output display by using the prebuilt OpenCV face detection and subtracting those results from the display. While this did reduce the number of objects the program recognized as “hands,” it still left us with results that contained too many false positives. We attribute that to the fact the classifier was trained on signer independent data. By having marked up images that contained multiple skin tones, background attributes, and lighting conditions, the classifier would have trained on all of those simultaneously allowing it to be able to accept a book case or door (for example) as a “hand” object although it is not. The confusion seen in the output led us to believe that this method alone would be unreliable in hand detection.
Using Andresen’s code for gathering the mean color of an individual’s hand for hand detection, we were able to modify it to obtain the mean hand color, convert the images we already had into negative binary images showing just the hands and face, and then save those images to a new file for use elsewhere [29]. This can be seen in figure 10.
Each of the dataset modifiers we created can also be applied to things other than hands. They are generic programs that can be used for the recognition of anything from faces, hands, flowers, birds, and whatever else you desire to focus on in an image. This makes it a helpful tool that individuals can utilize for creating their own datasets without having to start from step one of learning how it is done. These programs and datasets are a resource we plan to make available to the public so they can not only recreate our results, but can build off what we have established. The goal is to help make opportunities arise for the continuance of research done in this field instead of having to continuously build from the bottom up because open source code is unavailable, out of date, or too convoluted to use.

The discovery of Andresen’s program solved the initial signer independence issue by calibrating on skin color. The user places their palm inside certain extraction boxes where it finds the mean color of the hand in order to differentiate the hand (and consequently face) from the rest of the body and background as can be seen in figure 11.
and 12 [29]. We used this code to modify our image dataset to accommodate for new methods of hand sign recognition. We visualized the results and found them to be accurate in both hand and finger detection (figure 13). The system still confused the hands and face at times, however, that will be beneficial in the future when adding more dynamic and static signs that are differentiated by their relative location to the face. This allowed us to move on to specific sign recognition seeing as this was an adequate way to recognize a users hand. Although the code did not assist the deep learning portion of our research, this code will be useful when developing an end user application for sign recognition.

Figure 11: Example of Median Skin Tone Retrieval [29]
3.1 Stepping into Caffe

It was necessary to find a framework that would be robust and easy enough to
adapt for ASL recognition. Caffe provided a stable, well-documented, complete toolkit for training, testing, and fine-tuning recognition models. The deep learning algorithms and process itself was abstracted enough that a model could be trained without having any extensive neural networking background. That made it ideal for our research.

When the decision was made to use Caffe as the deep learning framework, it was our original intention to make use of the preexisting models in Caffe with their datasets; however, the datasets contained more images than we could store. Caffe requires that every training image have the same dimensions, so we updated our dataset likewise. An example of such an image can be seen in figure 14.

Figure 14: 256x256 Resized Image Example
Caffe needs each image to be classified numerically, so we chose the following encodings: ‘A’ = 0, ‘B’ = 1, ‘C’ = 2, ‘D’ = 3, ‘E’ = 4, ‘F’ = 5, ‘G’ = 6, ‘H’ = 7, ‘I’ = 8, ‘K’ = 10, ‘L’ = 11, ‘M’ = 12, ‘N’ = 13, ‘O’ = 14, ‘P’ = 15, ‘Q’ = 16, ‘R’ = 17, ‘S’ = 18, ‘T’ = 19, ‘U’ = 20, ‘V’ = 21, ‘W’ = 22, ‘X’ = 23, ‘Y’ = 24, “I Love You” = 26, “Water” = 9, Wine” = 25, and all negative images were classified as 99. It is imperative to note that Caffe can only train with classification values that are sequentially ordered starting from 0, it assumes that once values stop iterating after 0, the following images are not the ones you desire to train on. The network training is “supervised” meaning it learns through associating what it sees “visually” with what the correct classification should be. This means that throughout the training and in testing of a trained model, we are able to “see as we go” what the accuracy of the model is relative to what the anticipated outcome is.

The ImageNet model is a neural net that has 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax [2]. In order to train the Imagenet model on our dataset, we had to make several modifications to the layers and how the neural network would iterate. We left most of the pre-trained weights for the neural network alone, however we made several adaptations for how the layers would progress.

We changed the batch sizes for training and testing to suit the number of classification outputs we expected. The batch size needed to be a multiple of two and we had a total of 27 different image classifications so we rounded up to a batch size of 28. This meant that for each iteration through the neural network, 28 images would be
processed from the dataset at a time (one of each sign on average). We also modified (for different tests) the learning rate, number of iterations, step size, test interval, and test iterations values. The test iterations value and test interval values were kept the same for each model we attempted to train. The test iterations value was determined by the batch size, if you were to divide the number of images being trained by the batch size, you get an estimate of what the iterations value should be to the nearest power of ten. In our case that value was 100. The test interval value is based off of how many iterations you desire to wait before each accuracy test is performed on the model. We chose that value to be 1000. The learning rates we tested between 0.001 and 0.0001 since our dataset is smaller. By starting with smaller values (than the default 0.01) we were able to speed up the output classification process. Finally, we modified the step size and max number of iterations. These two values go hand in hand because the max number of iterations is the total number of iterations that will be run on the dataset, and the step size determines how quickly the learning rate will be lowered to notice more minute details. We originally chose to do 100,000 iterations at a step size of 10,000, however, due to hardware limitations we found it more reasonable to lower the number of max iterations to 30,000 with step sizes of 1,000 and 10,000 respectively for different tests.
4. RESULTS

Through the process of attempting to fully train the neural network to recognize the 28 ASL signs and phrases, we found that it was not possible in the time and with the hardware available. With reliable hardware, the training would have taken roughly ten days to complete the original 100,000 iterations we had originally attempted. We discovered that due to the hardware limitations of the server we were training on, it was not possible to complete. Although we reduced the number of iterations, as well as the number of neural networks that were training, the server continued to crash before any of the training was fully completed. The progress logs were lost, thus we are unable to accurately chart the accuracy of the training process. At this time we can not give an account as to whether using Caffe as a framework for training a neural network to recognize static ASL signs is ideal or not. However, we did conclusively find that using Haar Cascades alone is not ideal for ASL sign recognition.
5. CONCLUSION

As the Deaf community continues to grow [1], there is still a need for technology that can recognize American Sign Language. While we did not have enough time to fully implement such technology, we were able to produce programs and datasets that can contribute to the further pursuit of enabling easier communication between deaf and hearing people. We produced several programs for modifying and classifying data that are useful to topics and images not strictly related to hand signs. We have built a signer independent dataset that can be released for other researchers to use, along with marked up datasets for specific methodologies. We have modified code for identifying the hand in a frame to utilize in an end user program for recognizing ASL signs. We have also provided further support for training a Caffe model. Overall, despite the lack of a publically available signer independent dataset, a lack of open source code, time constraints, and unreliable hardware, we were able to complete significant components of this project and provide support for future research.
6. FUTURE WORK

In the future, it would be ideal to not only fine-tune a model to have a high level of classification accuracy, but also do so for significantly larger dataset. Hardware with over 500 GB of storage would be necessary along with having a reliable power source, and incorporating the use of GPU instead of CPU for faster training. Once an accurately trained model is obtained, it would then be possible to begin development on an end user system for ASL sign recognition for static signs. A system that could recognize bare hand dynamic signs would be the goal, however recognition of static signs would need to be accomplished first.
REFERENCES


Nov- 2015].

Sparse Representations with an Energy-Based Model,” Advances in Neural

Recognition', Bowling Green State University, 2006.


[22] Pew Research Center: Internet, Science & Tech, 'Mobile Technology Fact Sheet',


