Real-Time Sign Language Recognition

Jordan Bierbrier
bierbrj@mcmaster.ca

Abstract

Sign language is a ubiquitous communication standard for individuals who are deaf and/or mute. It can be difficult to effectively communicate with an individual who only knows sign language. This paper presents various image processing methods to bridge the communication gap between individuals who know and use sign language and those who do not, by creating a program that translates sign language into human readable text in real time. First the hand must be extracted from the image, which is done by background subtraction. Then three different means of classification are done. In one method, Histogram of Oriented Gradients (HOG) features are extracted then classified using a Support Vector Machine (SVM) classifier. Another method extracts features using Scale Invaraint Feautre Transform (SIFT) and classifies the features using an SVM classifier. Lastly, a Convolutional Neural Network (CNN) is partially trained by using transfer learning of the ResNet-50 architecture. The experimental results show us that the HOG method classifies 67% of the points correctly. The SIFT method classifies 77% of the images correctly and the CNN classifies 28% of the images correctly. The HOG method works the best in practice as it is computationally inexpensive and has good generalization to unseen test points.

1. Introduction

Communication is a fundamental to share ideas and information. Everyone around the world communicates with others for these reasons. Additionally, living in a data-driven world, there is a lot of information that needs to be shared between individuals and groups. These two examples demonstrate the necessity of communication. To demonstrate the ubiquity of sign language, there are over 1 million people in Canada and the United States who rely on sign language as their means of communication [3]. Sign language is used by individuals with impaired hearing or speech, namely the deaf and mute community. Sign language uses hand gestures, facial expressions and lip patterns instead of vocals to convey information. These gestures and movements are combined to form different meanings and sentences. Similar to spoken languages, there are a plethora of sign language dialects around the world. For example, there is the American Sign Language (ASL), which is considered the International Sign Language, Indian Sign Language (ISL) and Chinese Sign Language (CSL). Individuals who know sign language (in their respective dialect), can communicate with each other efficiently. However, individuals who do not know sign language cannot easily communicate with the deaf and mute community. In this situation an interpreter is required to aid the communication. Although this brings up the issue of finding an interpreter for a specific place and time. Additionally, simple communication would then become costly. Implementing a digital program or system that translates sign language into human readable text or speech can create a bridge of communication between deaf and mute individuals, and those who do not know sign language. Additionally, it can save you from the inconvenience mentioned above. This paper discusses and compares various implementations of systems which translate American Sign Language into readable text in real time.

2. Related Work

Various studies and methods have been proposed to translate sign language into speech or text. The proposed methods fall into one of two categories, which are glove based approach and vision based approach. Research has been conducted for sign language detection using sensor gloves to predict ASL [4]. Another research paper used flex sensors, an Arduino, and accelerometer to create a hand talking system which transmits the information to a smartphone [2]. The glove based approach obviates the step of hand segmentation during preprocessing. Although, it requires additional hardware, which can often be expensive or missing. Additionally, it can be inconvenient or awkward for users to wear. As a result, there has been a larger focus on the vision based approach. This process is more natural and convenient in everyday situations. The vision based approach requires preprocessing to detect and track a user's hand, then extract features from their hand, and classify the features. There have been various publications with

different approaches, but in general use the same methodology as mentioned above. Google has released real time sign language recognition model that estimates the individual's hand position and extracts features using optical flow then classifies the features using a linear classifier [6]. Another publication suggested extracting Histogram of Oriented Gradients features (HOG) then passing those features into a neural network for sign language recognition [8]. Additionally, a researcher proposed [9] a system wherein the Red, Green, Blue (RGB) image is converted into Hue Saturation Value (HSV) colour space to extract then hand. Following, Gabor filters are applied to extract features and Kernel Principal Component Analysis (PCA) is applied to reduce the dimensionality. Finally, it employs a Support Vector Machine (SVM) for classification. Another publication extracts the hand using the YCbCr colour space, then extracts HOG features and classifies the features using an SVM model [1]. Lastly, a publication uses Canny Edge Detection algorithm, followed by HOG feature extraction and an SVM classifier [5].

3. Proposed Method

There are three proposed methods to achieve sign language recognition. The operations in each method vary slightly and will be discussed in detail below. At a high level, Figure 1 demonstrates the stages for this sign language recognition. For each method, there includes a preprocessing stage. The feature extraction and classifier differs between the methods.

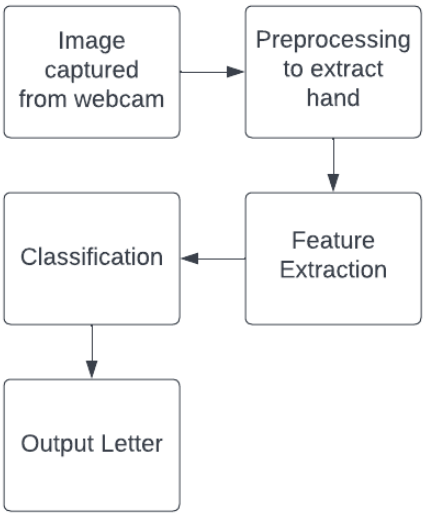

Figure 1. High level block diagram of sign language recognition

3.1. HOG Method

In this method, preprocessing is completed, then HOG features are extracted and finally the SVM classifier is used to predict ASL letters. This is shown in Figure 2.

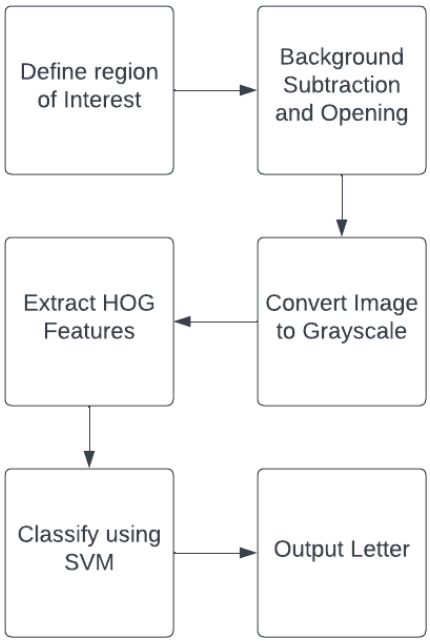

Figure 2. Block diagram for HOG method.

3.1.1 Preprocessing

In the preprocessing stage, there are three main components. First, a region of interest is defined within the frame. As opposed to searching the whole image for the hand, the user can place their hand in the bounding box displayed on the video screen. This decision was made as skin segmentation and hand detection are complex tasks in themselves, which require a lot of processing to locate [7]. Additionally, since lighting conditions change the appearance of your skin, it is an additional roadblock to searching for an individual's hand.

Following the region of interest, background subtraction is computed to segment the hand from the background. Equation 1 shows the equation used to threshold the region of interest. $P(x_0, y_0)$ corresponds to the initial background pixel value before the hand was placed in the frame.

$$P(x,y) = \begin{cases} 1, & \text{if } P(x,y) - P(x_0,y_0) \geq \text{thresh} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Therefore, if the current pixel value is greater than some predefined threshold, then set the pixel value to one (white), otherwise set to zero (black). This process creates a binary mask.

To eliminate any noise in the binary mask, the binary image processing technique of opening is applied. This operation removes small noises in the binary mask and only keeps the large objects. Finally, this binary mask is applied to the original region of interest. The hand (foreground) will be segmented and everything else will be black.

Lastly, the image is converted to a grayscale image. This is done because it is computational more efficient to work with one channel, and it is required for the feature extraction in the following step.

3.1.2 Feature Extraction

In this stage, HOG features are extracted from the segmented hand image. This feature extractor counts occurrences of gradient orientation in different areas of an image and combines them into a histogram. The combinations of the histograms then create features. Figure 3 demonstrates how the HOG features are created.

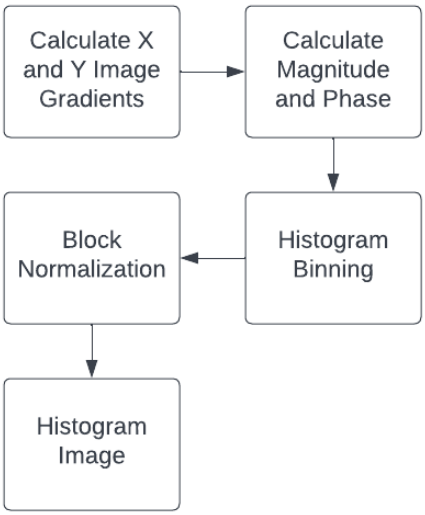

Figure 3. Block diagram for HOG feature descriptor.

First the image gradients are calculated in the X and Y direction. This can be achieved with Sobel filters which are shown below.

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Next, the magnitude and phase are calculated in equations 2 and 3.

$$|I| = \sqrt{I_x^2 + I_y^2} \tag{2}$$

$$\angle I = tan^{-1}\left(\frac{I_y}{I_x}\right) \tag{3}$$

The next step is to combine the phases within a spatial region. This is done by histogram binning. Each pixel within a cell adds a weight/vote for it's specific direction/phase. This creates many regions of histograms within the whole image.

Following, block normalization is done by concatenating neighboring histograms and dividing this vector by its magnitude. This is done to standardize the features. You will then have features descriptors for the segmented hand.

3.1.3 Classification

After the feature vector is obtained - which consists of HOG features - they can be classified using the SVM classifer. The output of the SVM classifier will be one of the classes (letters) of the ASL. By using SVM, the accuracy of the model can be maximized while reducing the chance of over fitting. To train the SVM classifier, the objective is shown in equation 4.

$$\begin{aligned} &\max \tfrac{2}{||w||} \\ &\text{s.t.}(w \cdot x + b) \geq 1, \forall x \text{ of class 1} \\ &\text{s.t.}(w \cdot x + b) \leq -1, \forall x \text{ of class 2} \end{aligned} \tag{4}$$

Although this is a multi class problem, equation 4 can be applied to all of the classes by pairing the classes up each once.

3.2. SIFT Method

In this method the Scale Invariant Feature Transform (SIFT) feature detector is used to extract features. The benefits of SIFT compared to HOG is that it is invariant to rotation and scale. Being rotation invariant is not that important for sign language recognition, since each orientation of an individual's hand has a meaning. Although, scale invariance is important. This is because the individual can hold their hand close to the camera or far away and they both have the same meaning. Therefore, the features should be the same regardless of their scale. This is the benefit that SIFT has over the HOG descriptor.

The procedure for using the SIFT feature descriptor is slightly different than the HOG method. Figure 4 outlines the steps for this method.

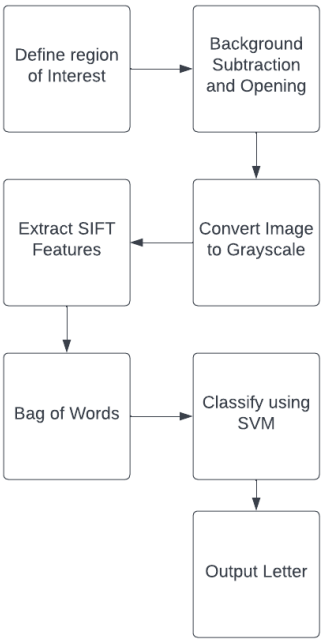

Figure 4. Block diagram for SIFT method.

3.2.1 Preprocessing

The preprocessing step is the exact same as described above in Preprocessing subsection for the HOG method. The end result is to have a grayscale segmented hand that can be passed to the feature descriptor stage.

3.2.2 Feature Extraction

The SIFT feature descriptor is another way to describe the detect and describe keypoints in the image. As mentioned previously, SIFT features are scale and rotation invariant. Figure 5 shows the four steps taken in the SIFT algorithm.

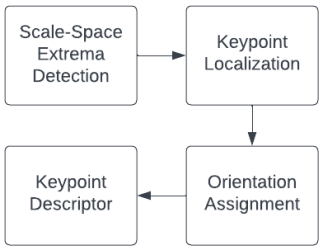

Figure 5. Block diagram for SIFT feature descriptor.

The first step is Scale-Space Extrema Detection. In this step, points of interest are found by taking the maxima and/or minima of the Difference of Gaussians (DoG) at different scales. The DoG (shown in equation 5) is an approximation of the Laplacian operator, which detects edges and sharp changes between pixels.

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma) \tag{5}$$

Keypoint Localization follows after keypoints are found from the previous step. In this step, keypoint locations are determined through interpolation of nearby data. Additionally, keypoints are discarded if they are below a certain threshold or they are edges. To determine if the keypoint is an edge, equation 6 can be used, where I_x and I_y image partial derivatives.

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{6}$$

Next, keypoints are assigned orientations based on gradient direction. As a result, this achieves rotation invariance.

Lastly, keypoint descriptors are created by taking a 4x4 neighbourhood each with a histogram of gradient directions (with 8 bins). Therefore, there are 16 different histograms with each 8 bins. This creates a a keypoint vector of dimension 128. In all, the feature descriptor will be of size (# of keypoints, 128).

3.2.3 Classification

An issue that may have been noticed while reading through the SIFT method section is that the number of keypoints can change, which changes the input size to the classifier. However, the classifier input has to remain constant. To solve this issue, Bag of Words can be employed.

Bag of Words is used to create a constant feature vector size. It works by finding all of the keypoints within the training data and clustering them using K-Means clustering. As a result, similar keypoints will have the same label. Following, a histogram for the frequency of each label is constructed for each image, by iterating over all of the keypoints for a specific image. As a result, the feature vector is now the histogram with the frequency of occurrences of each label (which was a cluster of keypoints). It should be noted that the number of clusters is a hyperparameter and should be adjusted during training.

The new feature vector can now be used as input to the classifier. The SVM classifier is also used for this method. The same objective (equation 4) is used to train the SVM classifier.

3.3. Convolutional Neural Network Using Transfer Learning

The third method to recognize ASL is using a convolutional neural network (CNN). CNNs have gained a lot of

popularity over the past decade since they are quite successful at image processing such as object localization and image classification. CNNs are able to learn the features by updating filter weights which extract these regions of interest. Since the problem is classification, the last layer of the CNN has to include a softmax function. The softmax function forces inputs all the inputs to be between 0 and 1. The output class with the highest value will have a value closest to 1, and that will be the corresponding class. The softmax function is shown in equation 7, where K is the number of classes and $\mathbf{z}$ is the input vector.

$$\sigma(\mathbf{z})_\mathbf{i} = \frac{\mathbf{e}^{\mathbf{z_i}}}{\sum_{\mathbf{j=1}}^{\mathbf{K}} \mathbf{e}^{\mathbf{z_j}}} \tag{7}$$

The parameters (filter weights) can be updated by defining a loss function, which determines how much the model's output differs from the true output. For the CNN trained, the categorical cross entropy loss function is used, which is shown in equation 8. Where N is the number of training points, $\hat{y}$ is the network output and y is the true label/output.

$$\text{Loss } = -\sum_{i=1}^{N} y_i \; log \; \hat{y}_i \tag{8}$$

3.3.1 Transfer Learning

Transfer learning is a machine learning technique wherein a pretrained model is used on a different dataset. This model is usually trained with thousands of data points and has a good generalization, which is crucial. Transfer learning since the model has good generalization and can extract common features from the images.

In this paper, ResNet-50 is used. It is an architecture that is 50 layers deep and can classify over 1000 object. ResNet-50 is a residual neural network which uses skip connections to jump over layers. An example of this skip connection is shown in Figure 6 They are employed to mitigate the problem of vanishing gradients and to reduce degradation. ResNet-50 is used as it has fewer parameters (25.6 million) compared to most other architectures, for example VGG16, which has 138.4 million parameters.

A fully connected layer with 64 neurons is added to the output to add some flexibility/training to the model. Following, there is the output layer with uses the softmax activation function to classify the input image.

4. Experimental Results

4.1. Dataset for Classification

For this task, the ASL alphabet is used. Since some of the letters use dynamic gestures, they are discarded from the classification. As a result, there are 24 static letters in the ASL alphabet. Additionally, 40 images of each letter were collected and processed to segment the hand. In total there are 960 images in the dataset. The image sizes are all 400 by 300 pixels. Figure 7 shows each letter along with hand segmentation.

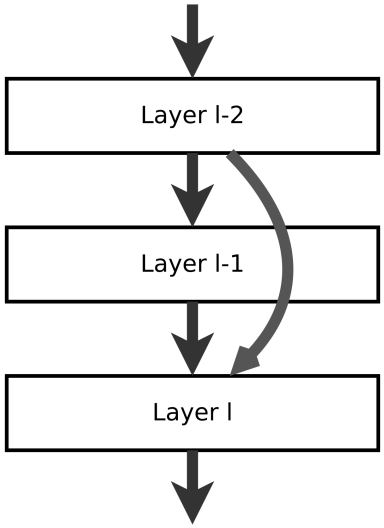

Figure 6. Skip connection used in residual neural network.

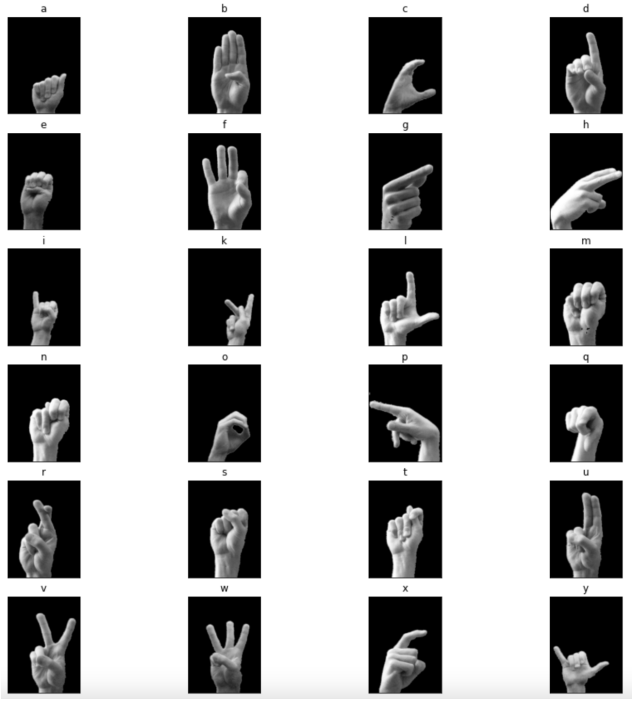

Figure 7. Dataset with 24 static ASL letters.

4.2. Hand Segmentation

As discussed in the Proposed Method section, hand segmentation is achieved by using background subtraction. Although binary image processing techniques are used to mitigate noise, that is, using opening, noise can persist in the segmented hand. Factors that control the quality of the segmented hand include the lighting conditions and the colour of the background compared to the individual's skin colour. Parameters can be tuned to counter these issues. Figure 8 demonstrates the issues presented above. Figure 9 shows how the hand can be properly segmented when lighting conditions are focused on the hand.

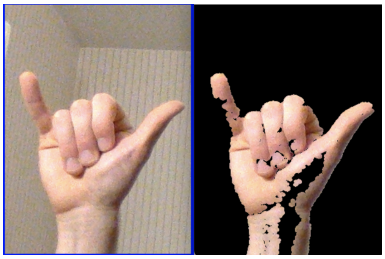

Figure 8. Noisy hand segmentation due to background colour similarity.

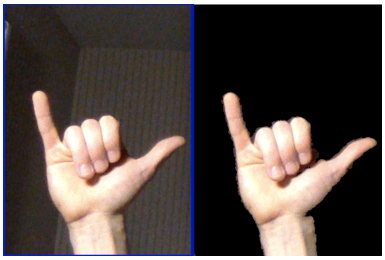

Figure 9. Adjusted lighting to achieve clean segmentation.

Figure 10 demonstrates another hand segmentation that has minimal amount of noise.

Figure 10. Hand segmentation for the letter Y.

4.3. HOG Method

After the hand is segmented from the background, features must be extracted. In this method, HOG features are extracted, which was explained above. Figure 11 demonstrates what these features look like from an grayscale image.

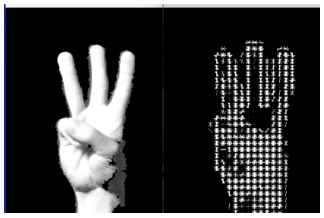

Figure 11. Grayscale segmented hand and HOG features for the letter W.

Training the SVM classifier, each HOG feature descriptor had a dimension of 1x17100. The training data was split up into a training and validation set, 768 training images and 192 validation images. Using a linear SVM classifier with a l2 penalty and loss function of squared hinge, the model achieved an accuracy of 67% the validation data. The confusion matrix for the validation data is shown in Figure 12. Most of the misclassfied letters were a, m, n as they are all very similar static gestures, which brings down the overall accuracy significantly.

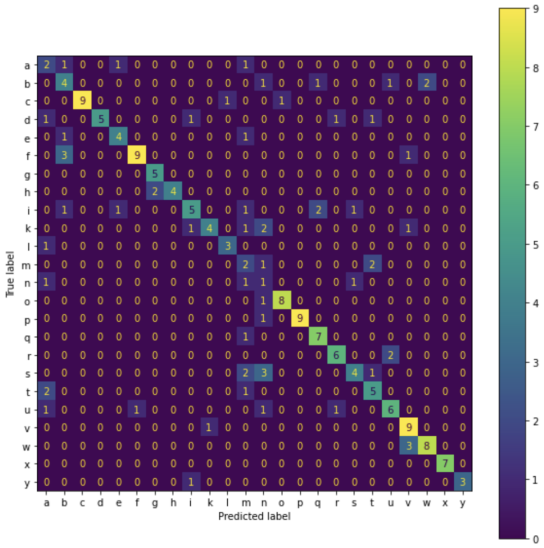

Figure 12. Confusion matrix for validation data.

Although the validation accuracy has room for improvement, testing the SVM classifier using HOG features in real-time performs quite well. Most of the ASL letters can be recognized almost immediately. The most difficult one to classify is n, which is quite similar to a and m. Otherwise,

most of the letters classify well. Additionally, it is not computationally expensive to extract the HOG features and predict the letter. The video camera can still show 13-16 frames per second (fps), which is enough to keep a clear video. Figure 13 shows 'happy' being spelt using the trained classifier. All of the letters were classified on their first attempt.

Figure 13. Real-time test with HOG method.

As mentioned above, the gesture 'n' sometimes does not correctly classify as n. For example, in Figure 14 the model falsely predicted the letter 'a'.

Figure 14. Real-time wrong classification for letter n.

4.4. SIFT Method

In this section, features are extracted using the SIFT method. These SIFT features can be visualized to see which points are being detected. This is shown in Figure 15.

For K-means clustering, 50 clusters were used. This was chosen as the training accuracy increased as the number of clusters increased, although too many clusters can lead to overfitting. Therefore, only 50 clusters were used. After training the SVM model (with 768 training images and 192 validation images), the validation accuracy was 77%. The confusion matrix for the validation model is shown in Figure 17. It can be seen that this implementation was able to better classify the letters a, m, than the HOG method.

It should be noted that during real time testing the classifier kept changing its guessed letter. This can be because the SIFT keypoints are very sensitive and keypoints are picked up at multiple scales. As a result, the classifier keeps getting different input feature vectors which causes it to constantly change its prediction/output.

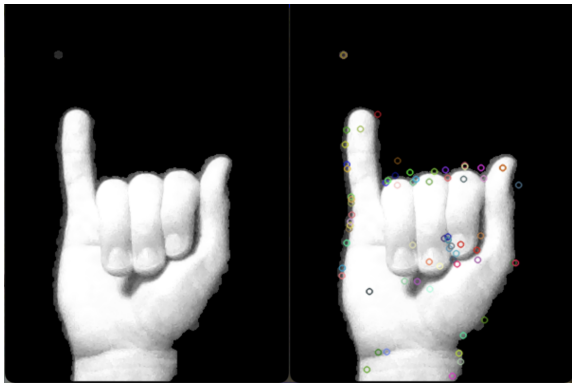

Figure 15. Keypoints detected using SIFT algorithm.

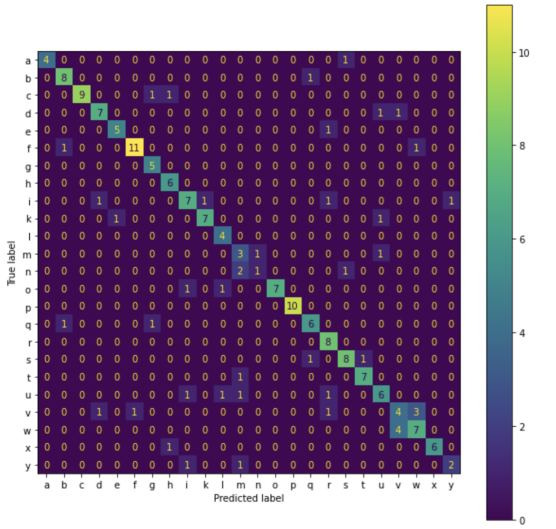

Figure 16. Confusion matrix for validation data.

Figure 17. Correctly classified letters using SIFT method.

Figure 18. Incorrectly classified letter e.

4.5. CNN Method

The CNN method uses transfer learning to aid in the classification. The architecture ResNet-50 was used along with two Dense layers that were trainable. The output layer has 24 outputs, one for each class, and uses a softmax activation function. The added layers and total number of parameters (trainable and non-trainable) in the network are shown in Figure 19. There are 132,696 parameters in the network, which are from the layers that were added at the end. However, the 23,587,712 parameters are pretrained on the ResNet-50 model from the imagenet database.

Layer (type)	Output Shape	Param #
resnet50 (Model)	(None, 2048)	23587712
dense_9 (Dense)	(None, 64)	131136
dense_10 (Dense)	(None, 24)	1560

Total params: 23,720,408
Trainable params: 132,696
Non-trainable params: 23,587,712

Figure 19. Layers and parameters for ResNet-50 architecture.

The model was trained for 10 epochs using the a loss function of categorical cross entropy. Figure 20 shows the training and validation accuracy. As it can be seen, the CNN model does not fit the data well, since the largest validation accuracy is 28% in epoch 2 (epoch starts at 0). Additionally, the model overfits the data since the training accuracy reaches 100% but the validation accuracy remains low. Figure 21 shows the confusion matrix for the validation data. Most of the data is incorrectly classified as a, k and l.

There are two main reasons why the model performed poorly. First, there are too many parameters and not enough training points. This can be seen as the validation accuracy remains low as the training accuracy reaches 100%. Increasing the number of training points will reduce overfitting. Secondly, and most importantly, the ResNet-50 model with the pretrained weights is not a good transfer learning model for this specific application. This is because the model was trained with different samples that vary from the images used in this task.

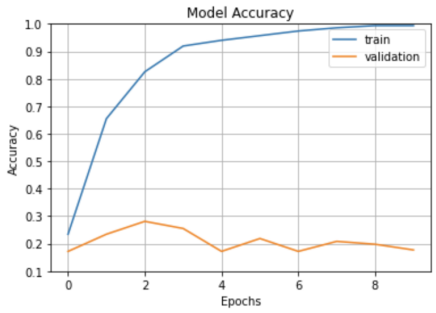

Figure 20. CNN training and validation accuracy.

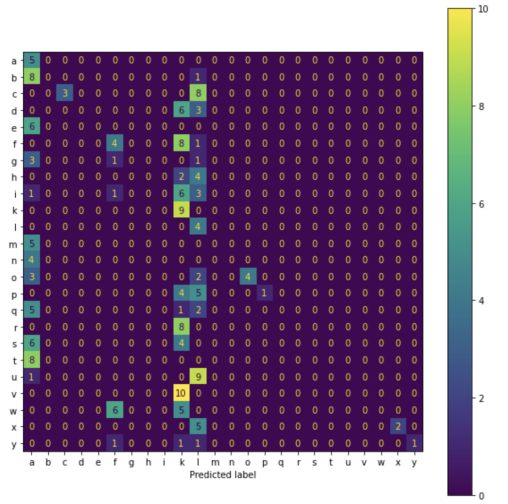

Figure 21. Confusion matrix for validation data.

Real time testing was done with the model to see its performance. Although it was able to classify some points letters correctly, as seen in Figure 22, its speed was very slow. It takes approximately $\frac{1}{0.669 \text{ fps}} = 1.5s$ to classify an image (fps taken from Figure 22). As a result of the high computation, the real time video is very choppy and slow. Additionally, Figure 23 shows an incorrectly classified letter.

Figure 22. Correct classification of letter.

Figure 23. Incorrect classification of letter w.

5. Summary

In this paper three methods for sign language recognition were implemented and compared. All of the classifiers were trained with 960 test images, split up into 768 training images and 192 validation images. The first method used HOG features and an SVM classifier. This method achieved a validation accuracy of 67% and was computationally efficient in real time videos. The second method used SIFT features and an SVM classifier. This method achieved a validation accuracy of 77%. In real time videos, the SIFT features are very sensitive so the classifier keeps outputting different letters, although a majority are the correct letters. The third model used transfer learning of a pretrained CNN, ResNet-50 with imagenet. This achieved a highest validation accuracy of 28%. This is because imagenet does not classify sign language letters, but rather dogs, cats, humans, etc. Additionally, due to the large number of parameters, it takes a considerable amount of time to classify an image.

In summary, the best performing method is the HOG method. This is determined through live testing as opposed to strictly looking at validation accuracy. This method can successfully recognize ASL letters and convert them into human readable text. This implementation can help mute or deaf individuals communicate with those who do not know sign language.

References

[1] Shubhendu Apoorv, Sudharshan Bhowmick, and R Prabha. Indian sign language interpreter using image processing and machine learning. *IOP Conference Series: Materials Science and Engineering*, 872:012026, 06 2020. 2

[2] Abdullah Baktash, Saleem Mohammed, and Huda Farooq. Multi-sign language glove based hand talking system. *IOP Conference Series: Materials Science and Engineering*, 1105:012078, 06 2021. 1

[3] Sevgi Z. Gurbuz, Ali Cafer Gurbuz, Evie A. Malaia, Darrin J. Griffin, Chris S. Crawford, Mohammad Mahbubur Rahman, Emre Kurtoglu, Ridvan Aksu, Trevor Macks, and Robiulhossain Mdrafi. American sign language recognition using rf sensing. *IEEE Sensors Journal*, 21(3):3763–3775, 2021. 1

[4] Syed Atif Mehdi and Yasir Niaz Khan. Sign language recognition using sensor gloves. pages 2204 – 2206 vol.5, 12 2002. 1

[5] Stafford Michahial. Hand gesture recognition using support vector machine. *The International Journal Of Engineering And Science*, 4:42, 06 2015. 2

[6] Amit Moryossef, Ioannis Tsochantaridis, Roee Yosef Aharoni, Sarah Ebling, and Srini Narayanan. Real-time sign language detection using human pose estimation. 2020. https://www.slrtp.com/. 2

[7] Khamar Basha Shaik, P. Ganesan, V. Kalist, B.S. Sathish, and J. Merlin Mary Jenitha. Comparative study of skin color detection and segmentation in hsv and ycbcr color space. *Procedia Computer Science*, 57:41–48, 2015. 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015). 2

[8] Neha V. Tavari and Prof. A. V. Deorankar. Indian sign language recognition based on histograms of oriented gradient. 2014. 2

[9] Md Azher Uddin and Shayhan Ameen Chowdhury. Hand sign language recognition for bangla alphabet using support vector machine. In *2016 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, pages 1–4, 2016. 2