HIGH SCHOOL EDITION
Journal of Student Research
Volume 13 Issue 4 (2024)

Hierarchical CNN ASL Recognition with Feature Comparison

Antony Ouyang

[1]Lexington High School, USA
[#]Advisor

ABSTRACT

Every day, many individuals live with conditions such as deafness, muteness, or blindness, and they struggle to communicate effectively with others. Effective communication between deaf and hearing individuals is also crucial for accessibility and inclusion in daily life, including education, healthcare, and public services. Sign Language Recognition technology addresses this challenge by providing real-time translation of sign language into text or speech, facilitating smoother and more natural interactions. The increasing importance of SLR lies in its ability to bridge the communication gap fast and accurately, making everyday activities more accessible for deaf people. However, most such approaches to addressing these challenges primarily relied on labeled output from a neural network, but these methods have not provided a comprehensive solution when it comes to immense sign variation.

A Short Overview and Introduction of Sign Language Recognition System

The Sign Language Recognition System, or SLR, is a common way to help deaf and hearing people communicate and understand each other better. It's designed to interpret and translate hand gestures into corresponding words or phrases. The system typically processes video or image data, where the hands and body movements are segmented and analyzed using computer vision techniques such as Convolutional Neural Networks (CNNs), which are often employed to extract features from these segmented images, enabling the model to recognize and classify the signs. To improve accuracy, various preprocessing steps like color filtering, optical flow, and background subtraction are utilized to isolate the hand shapes to create good images for learning [2]. Recent advancements in deep learning and image processing have significantly enhanced the system's ability to recognize signs in real time, making SLR systems increasingly effective in practical applications.

Despite the advances, these systems typically rely on convolutional neural networks (CNNs) that have been trained on large datasets of labeled sign images. The CNNs then extract features from the input data and directly classify them into one of the predefined output labels (which correspond to specific signs) based on the patterns learned during training.

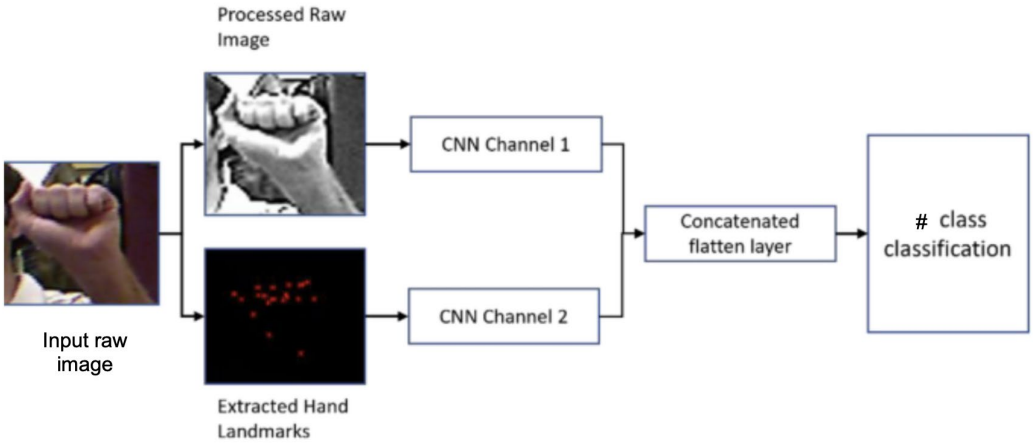

Figure 1. Traditional recognition system

This would work well on relatively small vocab set size, but when dealing with large vocabularies, it will take much longer time to run and output since the CNNs have such a huge number of label types to output. With increasing vocab size and sign variation, efficiently recognizing a large vocabulary of signs in a short time is crucial for real-time communication and ensuring seamless interaction between deaf and hearing individuals.

This paper delves into alternative methods for sign recognition, incorporating essential aspects of the SLR system, such as convolutional neural networks (CNNs) and image segmentation or preprocessing techniques. Furthermore, it uses different classification approaches, including comparing recognized images to a database of labeled images and utilizing hierarchical categories to reduce the number of comparisons required. By combining these methods, the algorithm seeks to reduce the complexity and time required for the recognition system to accurately identify and output the correct sign while still maintaining high accuracy and efficiency.

Basics of American Sign Language

American Sign Language (ASL) is the main language of Deaf and hard-of-hearing communities in the United States and parts of Canada. Unlike spoken languages, ASL is visual, relying on a combination of hand movements, facial expressions, and spatial placement to convey meaning. It has its own grammar and syntax, distinct from English, making it a unique and complex form of communication. Some signs in ASL include finger spelling using the alphabet, where each letter has a corresponding hand shape, as well as other signs that correspond to a word like “hello” with a salute gesture, “thank you” by moving the hand away from the mouth, and “love” with crossed arms over the chest. ASL is not universal; it varies regionally and even has different dialects. Its development has roots in early sign languages from France and indigenous sign systems in North America.

Today, ASL is recognized as a vital means of communication and cultural expression in the Deaf community. However, not every person would learn ASL since most people don’t encounter deaf people, and it’s completely different from most languages that are based on sound. So, having a recognition system will increase the convenience of communication between deaf people and hearing people and potentially make the hearing and Deaf community blend in better.

Overview of Computer Vision Techniques Like CNNs and Feature Comparison

One of the major deep learning and feature extraction (as well as sometimes label prediction) is the convolution neural networks or CNNs [5]. A convolution network uses convolution filters that extract features from an image and

optimize the filter parameters through training. The convolutions detect the feature from a smaller one in the beginning layers to a more complex hand feature in the later layers. For example, w × h × c, w=width, h=height, c=number of channels. The convolution multiplies the previous volume by *nc* filters of parameters, usually each with dimension f × f, where f is odd. It first puts the parameters of the f-by-f filter onto each of the channels in the input volume's first f by f square on the original image (the top left corner f by f square), and the corresponding position in the input volume is multiplied with its position in the filter [5]. Then, the f by f by c volume is summed up into a value, which is the top left corner of the output volume.

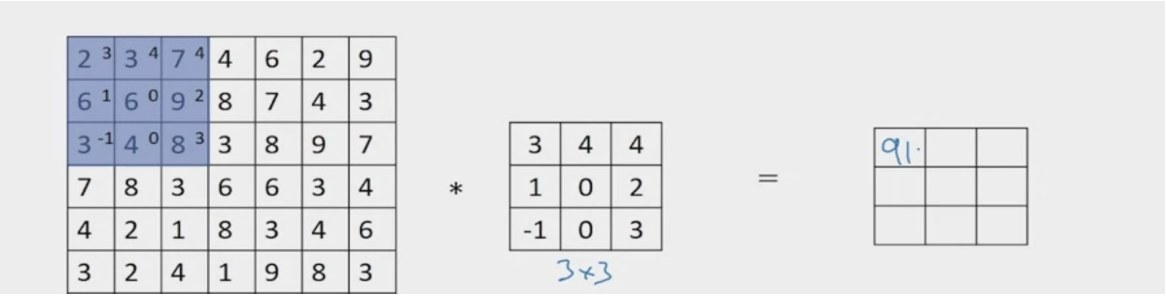

Figure 2. Convolution calculation

There are also input hyperparameters like the padding p that make the output volume's face (the first two dimensions) have the same dimension as the input volume's face; another important one is the stride s that controls how many windows the filtering on the input image moves in each convolution calculation [5]. With all those, the output of one convolution step of *nc* f × f filters, padding of same (same face dimension for output as input), and stride s, the output volume's size would be $\left\lfloor\frac{w-f+2p}{s}\right\rfloor$+1 by $\left\lfloor\frac{h-f+2p}{s}\right\rfloor$+ 1 by *nc*.

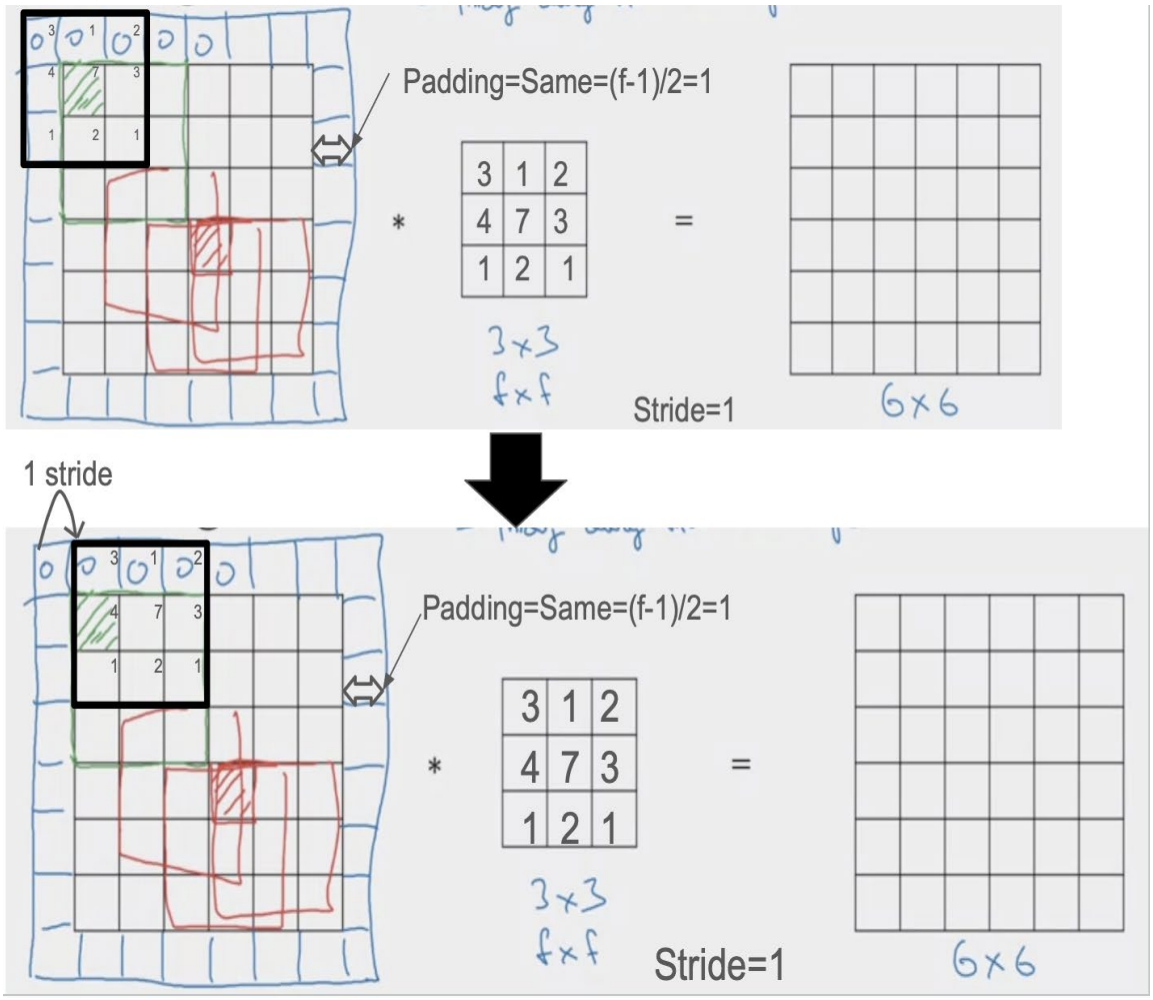

Figure 3. Convolution layer dimension with hyperparameters

Pooling layers are typically put between the convolution layers to reduce the dimension of the output of the features by the convolution layer and to make sure that the strongest feature is selected or the general feature is kept in a smaller window, depending on the case [6]. This not only helps the neural network to keep its strongest or general features but also makes it more sustainable to noise. There are two types of pooling: max pooling and average pooling. Max pooling reduces the size of the features generated by finding the strongest value/feature from a certain filtering window on the input image, while average pooling averages the values/features [6]. Since the operation types on a certain window in the input image are determined by whether it's Max or Average, unlike convolution layers, the pooling layer doesn't have parameters to train. Pooling layers do have hyperparameters like the filter size f and stride s [6]. So, if an input volume has dimension w × h × c, with filter size f × f, stride s, the output volume will have dimension $\left\lfloor\frac{w-f}{s}\right\rfloor$+1 by $\left\lfloor\frac{h-f}{s}\right\rfloor$+ 1 by c.

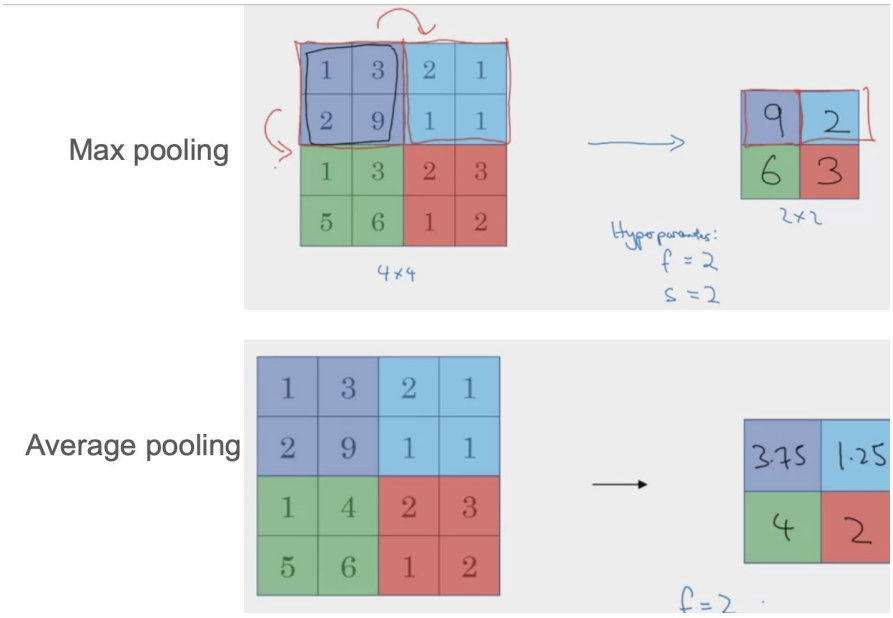

Figure 4. Pooling layer dimension with hyperparameters

Another important technique in the recognition system is feature comparison. When you only have one image of each label, it's very hard to train an accurate CNN as the training set is too small, and feature comparison solves the issue by just comparing the new features to the feature that is known in the database. 2 of the most common feature comparison is triplet loss and binary classification. Triplet loss uses an archer image, positive (same label/type) image, and negative image to determine the differences. Binary classification takes two images, one in the database and one in the new image, and calculates the difference between them [4]. A threshold is set for both algorithms to determine if the images are the same thing. So, suppose there's feature vector A for the anchor image, P for the positive image, N for the negative image, and $dP = (A - P)^2$ for each value in A and P, similarly $dP = (A - N)^2$ for each value in A and N. The total triplet loss is then

$$\sum_{i=1}^{n} dN_i - \sum_{i=1}^{n} dP_i$$

As was said before, a threshold of alpha α is used to determine if the positive image has the same type as the anchor image. So, it's the same if

$$\sum_{i=1}^{n} dN_i - \sum_{i=1}^{n} dP_i \geq \alpha$$

, or

$$\sum_{i=1}^{n} dN_i \geq \sum_{i=1}^{n} dP_i + \alpha$$

Alpha's a hyperparameter though, so it's a input to the triplet loss.

Binary classification, unlike triplet loss, uses the trained parameters of the difference of feature vectors generated at the end of CNNs to determine if the new image is the same label as the image in the dataset (or a known image using just two images [4]. Suppose a known image with the known label has its feature vectors xi generated by

CNNs compared with the feature vectors of a new image xi, then the binary classification will put the two feature vectors into another classification layer, which takes in the vectors and output

$$y = \sigma(wl \times (xi \cdot xj) + b)$$

where wl and b are the weight and bias parameters that could be previously trained on similar image sets and applied through transfer learning, and σ is the activation function of that layer [4]. The expression $xi \cdot xj$ is the dot product of the 2 feature vectors xi and xj, given by summing the product of each of their element. With output y being a probability between 0 and 1(closer to 1 is more similar), the binary classification is called a Dot-Product Based Model [4].

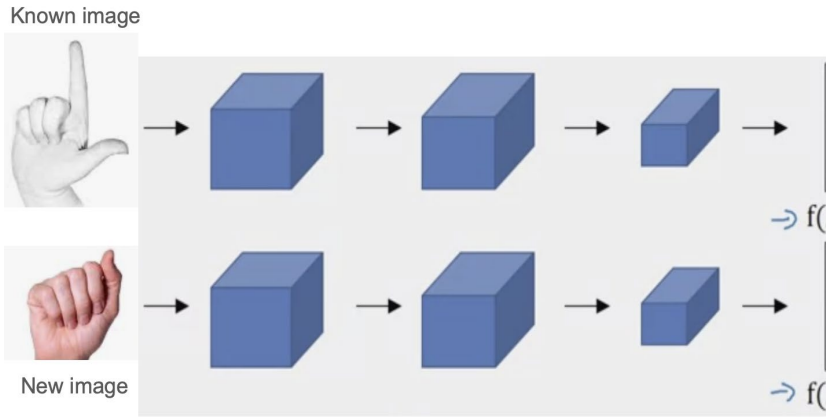

Figure 5. Binary classification with image pair

ASL Recognition with Hierarchical Comparison CNN Algorithm

Overview Through a Simple Example of Using Hierarchical Comparison Convolution Neural Network in ASL Recognition

For example, to recognize ASL signs in the Kaggle dataset [1], which contains 16054 signs for the training set, a structured approach is vital. The dataset consists of labeled data, so X represents the images, and Y corresponds to the labels or predicted outcomes. Initially, we can apply color filtering and optical flow techniques (since ASL signs at least move a bit) to isolate the hand shape and facial expression within each image in the training set to ensure the key features are highlighted. In cases where signs are rotated, image rotation techniques can be employed to correct and realign the signs for more accurate recognition. Using this pre-processed masked data of segmented sign features in the training set and training a CNN's filtering parameters to learn from the labeled images, its ability to recognize and predict ASL signs will be faster and more accurate.

Then, train a CNN to accurately recognize features within the images of the 16054-sign training set. Once the CNN's parameters are optimized, we'll generate the features, which are mainly hand shape features and facial expressions for each of the segmented images in the training data set using the trained CNN. We can then use the trained or the most optimal filter parameters to output the feature vectors for each of the images in the training dataset.

Next, train a classification layer using two sets of images from the Kaggle dataset [1], and ensuring that each set has corresponding labels since some images overlap. Then, when recognizing the new ASL sign image, we can

first use the trained CNN to output the features for the new images. Finally, input the feature vector xi of the new image into the trained binary classification layer alongside the feature vector xj of a known image for comparison [4].

For large datasets like the 16054, comparing the new image to each of the 16054 sign image's feature vectors could take a long time to run. So, we can choose one in the first several general categories to find the closest match using the trained binary classification layer and do the same within that general category until we go down to the sign. For instance, like the 16054 datasets, we could do the first category layer with four types, the second category layer (the category that is in the chosen type in the first layer), six types, the third layer with 23 types, and 4th layer with 29 signs. This is 16008 signs, and according to Kaggle, each sign is repeated on about ten images in the 16054-sign dataset [1], so we can definitely reduce 46 images by deleting off some of the overlapping images to get to the final sign without comparing every single image in the dataset.

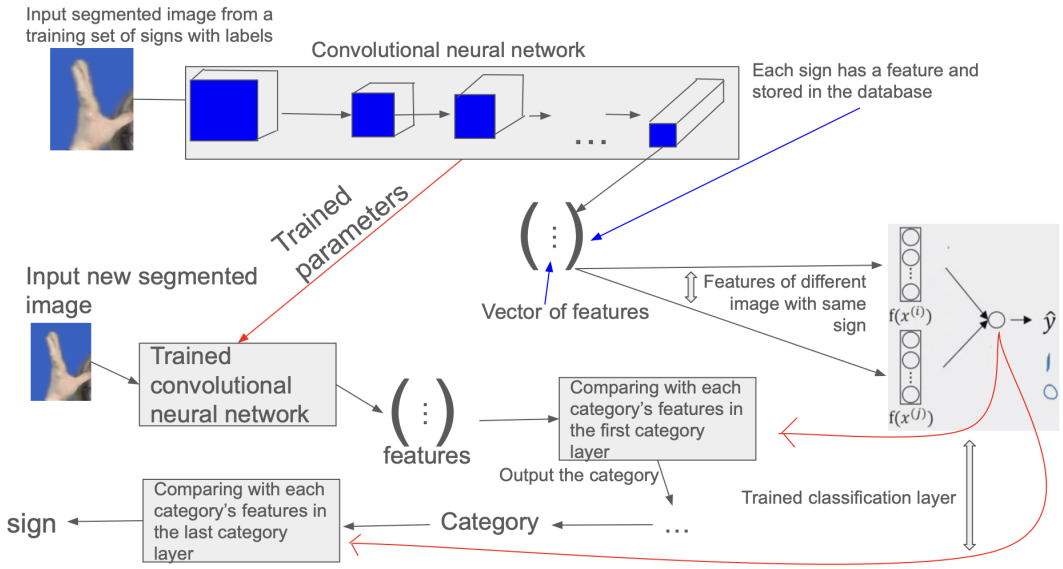

Figure 6. Alternative way for ASL recognition

Image Preprocessing for Hierarchical Comparison CNN ASL Recognition

When taking in a sign image or any sort of image that is going to be analyzed, it's always good to segment out the part you want (or ROI) to yield better accuracy and better conditions for feature extraction. Some of the most common image segmentation techniques include color filtering, optical flow, and background subtraction [2].

Color filtering, which filters a color image based on its color aspects (or called color spaces), can be done in 4 color spaces: RGB, HSV, YCbCr, and LAB to segment out the ROI (in here the hand sign and facial expression). RGB is just the red, green, and blue combination. HSV is HSV, which stands for hue, saturation, and value. These are the three components used to describe colors in the HSV color model. Hue refers to the color type and is represented as a degree on the color wheel, ranging from 0 to 360 degrees. Saturation indicates the intensity or purity of the color, ranging from 0 to 100 percent, with higher saturation meaning more vivid colors and lower saturation meaning more washed-out colors. Value, also known as brightness, refers to the lightness or darkness of the color, ranging from 0 to 100. YCbCr is a color model used primarily for video compression and digital image processing. It represents color as three components: Y (luminance), Cb (chrominance-blue), and Cr (chrominance-red). The Y component corresponds to the brightness or intensity of the color, similar to grayscale, while Cb and Cr represent the color information by describing the difference between the blue and red channels with the luminance. The L*a*b* color space is a model that describes colors based on human vision, designed to be more perceptually uniform than other color spaces. The

L* component represents the lightness of the color, ranging from 0 (black) to 100 (white), indicating how light or dark a color appears. The a* component represents the green-red axis, where positive values indicate a shift toward red and negative values indicate a shift toward green. The b* component represents the blue-yellow axis, where positive values indicate a shift toward yellow and negative values indicate a shift toward blue.

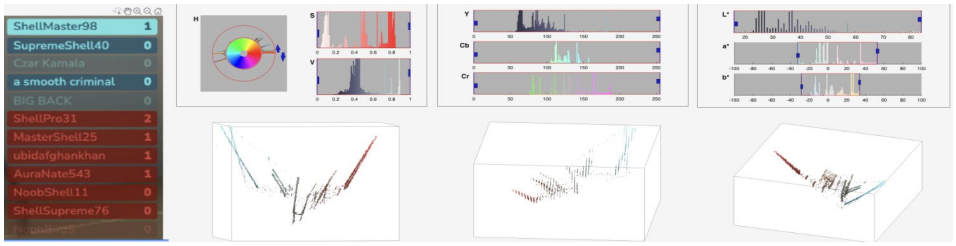

Figure 7. Color spaces for the image 3v6 team

You can also convert the RGB input sign image into grayscale by using the function rgb2gray() in MATLAB and use the image segmenter app to segment out the hand shape and facial expression.

Another useful segmentation is optical flow. When the color of the ROI is not working out well, you can use optical flow to detect the moving object in the image. And in ASL, both hand signs (especially moving signs) and facial ex-pressions move. For example, we could use two consecutive frames and find the movement difference be-tween the two frames using the estimated flow(opticalFlowFarneback, inputting) function, which essentially takes inputting and calculates the flow vector for each of the pixels. Then, a threshold is set to filter out the objects with a flow vector magnitude bigger than the threshold. We can have the hand sign and facial expression segmented out since they are going to move faster relative to the background.

And at last, background subtraction. Similar to optical flow, it detects ROI through its motion, but back-ground subtraction subtracts the background from the image in cases where you have an image with no ROI (in sign recognition, no signer but the same background). After the subtraction is done, the output will show an ROI, while most other background regions are black, just like the optical flow filtering.

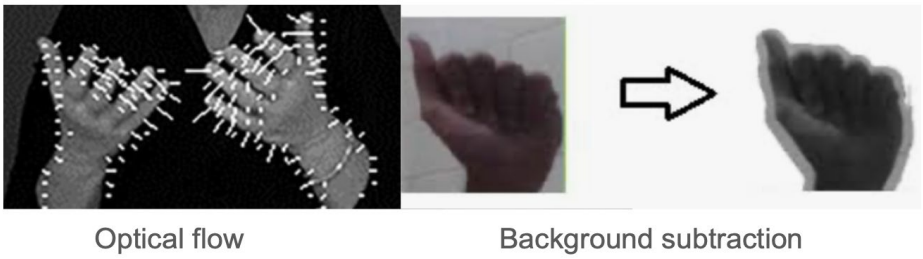

Figure 8. Optical flow vs background subtraction; image credit: [3]

For cases where the ROI is rotated after segmentation, image rotation can be used to correct the masked image to give better results in later feature extraction steps. In MATLAB, we can usually use the function imrotate(in-putimg, degree, 'crop' or 'loose') to correct the image with rotated sign.

CNN Training for Hierarchical Comparison CNN ASL Recognition

As what was said in section 3, a convolution network uses convolution filters that extract features from an image and optimize the filter parameters through training [5]; even when a segmented sign image is somewhat inaccurate, as

long as the general shape is discernible, the neural network can still make reliable predictions. This process not only facilitates accurate ASL recognition but also underscores the potential for machine learning to aid communication. After segmenting the hand signs and facial expressions in each image in the training set, we can feed them into a convolutional neural network. The goal is to train the convolutional layers' parameters to achieve the highest possible accuracy or attain a training error within 1 percent of the Bayes error, which represents the lowest achievable classification error. Additionally, we aim for the test set performance to closely match that of the training set.

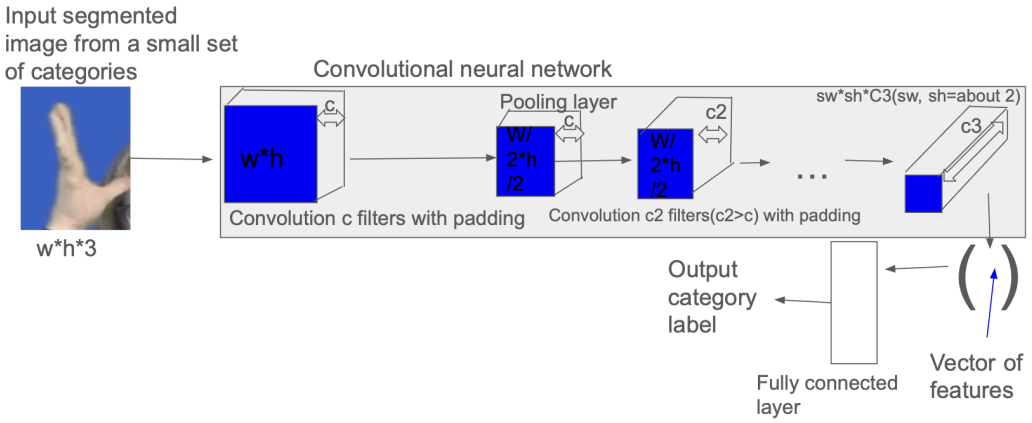

Figure 9. CNN network training

So, suppose each image input is RGB with width w and height h. Applying the first convolution layer with the padding option is the same, so the face dimension will be the same as the input segmented image's face dimension (face is the first two dimensions of a 3D volume) [5]. The number of filters will determine how many channels are in the output convolution layer. So, with all those being said, with padding as same, stride 1, and a f-by-f filter, then the output volume will be $\left\lfloor\frac{w-f+2[(f-1)/2}{s}\right\rfloor+1$ by $\left\lfloor\frac{h-f+2[(f-1)/2]}{s}\right\rfloor+1$ by c. Next, the pooling layer is primarily used to reduce the face dimension of the volume outputted by the convolution layer, so there's no need for padding, and the channels will stay the same [6]. Then with stride s equal to filter size f in most cases to ensure that each pooling window is not overlapping, the output volume size will then be

$$\left(\left\lfloor\frac{w-f}{s}\right\rfloor+1\right)\times\left(\left\lfloor\frac{w-f}{s}\right\rfloor+1\right)\times c=\left(\left\lfloor\frac{w-f}{f}\right\rfloor+1\right)\times\left(\left\lfloor\frac{w-f}{f}\right\rfloor+1\right)\times c=\left\lfloor\frac{w}{f}\right\rfloor\times\left\lfloor\frac{w}{f}\right\rfloor\times c$$

in the above diagram, $f = s = 2$ in the pooling layers, and w and h are both even. So, the convolution followed by pooling continues $L/2$ layers to the last sw by sh by $c3$ volume, where L is a hyperparameter previously set. As you can see in the diagram, sw and sh are very small, while $c3$ is quite big, so the last layer is essentially a long series of complex features. It is then turned into a feature vector and passed into a fully connected layer to get the output label [5].

Another important layer that needs to be trained is the binary classification layer. After training the CNN to its expected accuracy, we can output a feature vector for each of the sign images in the training set using the trained parameter of the CNN convolution layer filters [4].

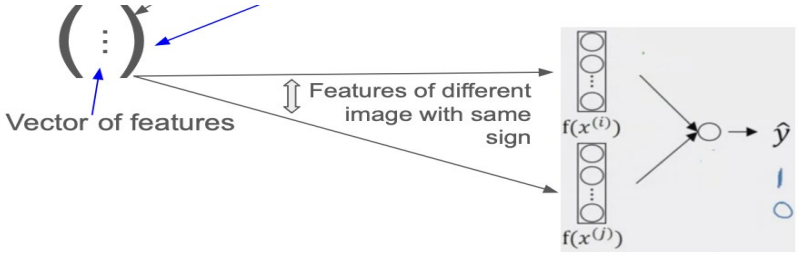

Figure 10. Binary classification layer training

Then, train a binary classification layer with some of the overlapping labels in the training set. The binary classification takes the two feature vectors xi and xj to calculate the output y, which outputs a probability between 0 and 1, through

$$y = \sigma(wl \times (xi \cdot xj) + b)$$

, where wl and b are the trained parameters, both being a scalar or having dimension 1×1. And the output y is a probability between 0 and 1, with 1 indicating completely the same [4].

Recognition for Hierarchical Comparison CNN ASL Recognition

With all the CNN and classification layer parameters trained and the feature vector for each of the images stored in the database, we can now get a new image and first do similar segmentation and preprocessing on them, just like what we did with the training set. Then, the trained CNN with its trained parameters will be used to get the feature vector output for the new images through transfer learning. Since with the preprocessing for both the new sign image and the training set sign images, the segmented hand sign and facial expression, or ROI, is going to be very similar in the new image and the training set image, transfer learning is viable. If the dataset size is too big to compare, so that the binary classification would take too long, we can select some big categories and put signs into each category. The category size and the many layers of the category are determined by the specific signs and the total number. For example, in the diagram below, I used the Kaggle training set [1], and it has 16054 sign images;
most signs have ten different images repeated, so we can reduce it to 16008.

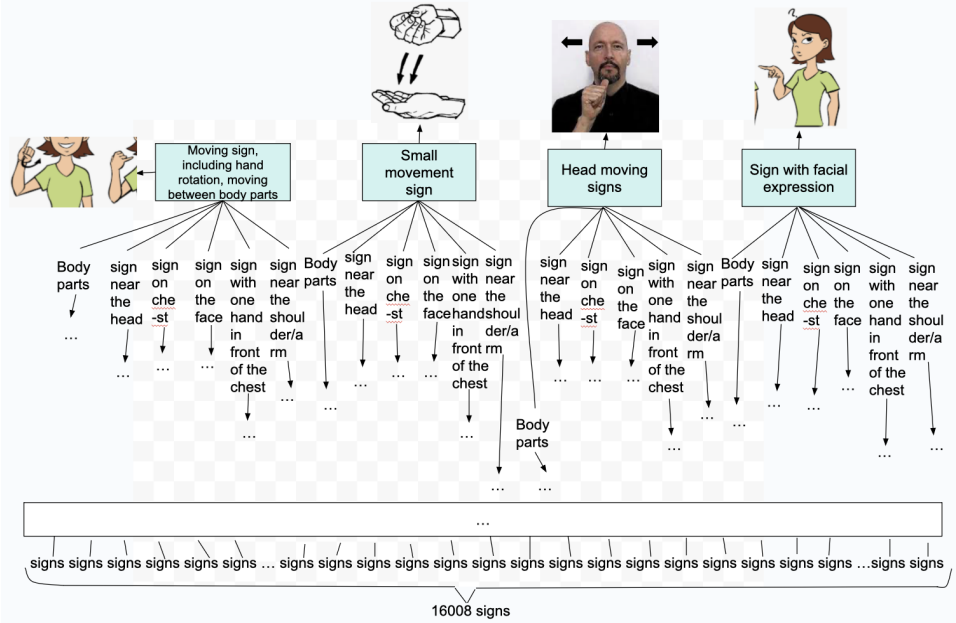

Figure 11. ASL sign category split example for Kaggle dataset 16008 train signs

Using a 4-layer category of distribution 4 × 6 × 23 × 29, we can first split the signs into moving signs (like letters J and Z fingerspelling in ASL, so signs that move between body parts), small movement (most ASL signs that don't move half a distance from the screen), head movement (mostly affirmative or negative), and facial expression. The second layer could be body parts, including signs near the head, signs on the chest, signs with two hands not on the chest but in front of it, signs across the chest, signs on the face, signs with one hand in front of the chest, and sign near the shoulder/arm. And so on, two more layers to the final sign, as shown in Figure 10. This is just an example; the exact numbers are based on signs in the training set.

In determining the new image category for each category layer, we can use the trained binary classification layer to compare the new image to each representative image in each category and update the highest output (most similar) y with the category pair number in each comparison [4]. This way, the classification layer compares a total of 62 times; the maximum times the classification layer needs to compare is the 29 signs in the last layer by the 4 × 6 × 23 × 29 category distribution. If we didn't use hierarchical categorization, the binary classification would need to be compared over 10k times without eliminating some signs previously.

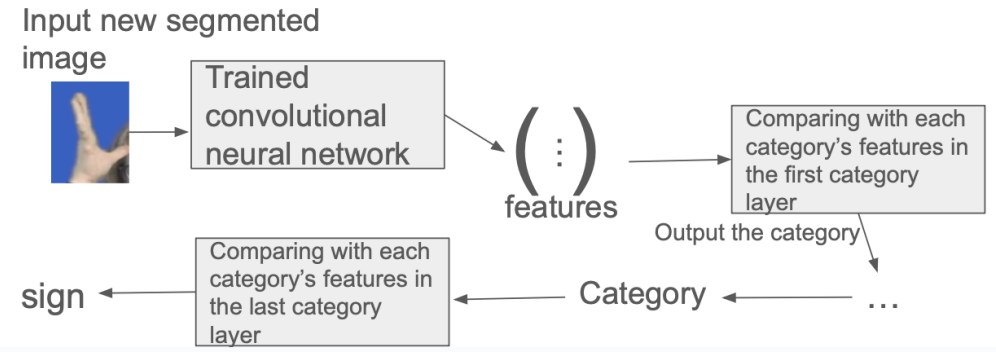

Figure 12. Hierarchical recognition comparison

Conclusion

In conclusion, using feature comparison through a binary classification layer and using hierarchical categorizing has provided us with an alternative way for sign recognition that might be faster on larger datasets than the traditional sign recognition system. Although the system does take some time to prepare and train the data through the convolution neural network and train the classification layer, especially when the training data size is big, it's better to have a long training time than a long runtime.

The system will evolve to handle larger and more diverse datasets while becoming more robust in different lighting and background conditions. Enhancing real-time processing capabilities will enable effective performance in live recognitions or translations. Exploiting the hierarchical classification approach will increase efficiency, particularly for recognizing thousands of unique signs. Incorporating additional modalities, such as depth sensing or motion tracking, can further boost accuracy and adaptability. The system might also be optimized and modified for deployment on edge devices, making it accessible for widespread use in various real-world applications.

Acknowledgments

I would like to thank my advisor for the valuable insight provided to me on this topic.

References

[1] *American Sign Language Dataset*. (2021, February 6). Kaggle. https://www.kaggle.com/datasets/saurabhshahane/american-sign-language-dataset
[2] Bhavsar, K., Ghatiya, R., Gohil, A., Thakkar, D., & Shah, B. (2021). Sign language recognition. *International Journal of Research Publication and Reviews*, *2*(9), 771–777. https://ijrpr.com/uploads/V2ISSUE9/IJRPR1329.pdf
[3] Joglekar, S., Sawant, H., Jain, A., Dhadda, P., & Sonawane, P. (2020). A Multi-Modular approach for gesture recognition and text formulation in American sign language. *International Journal of Computer Applications Technology and Research*, *9*(7), 217–224. https://doi.org/10.7753/ijcatr0907.1001
[4] Ng, A. (n.d.). *Face verification and binary classification* [Video]. Coursera. https://www.coursera.org/lecture/convolutional-neural-networks/face-%20verification-and-binary-classification-xTihv
[5] Ng, A. (n.d.-a). *Convolutions over volume* [Video]. Coursera. https://www.coursera.org/lecture/convolutional-neural-networks/convolutions-%20over-volume-ctQZz
[6] Ng, A. (n.d.-c). *Pooling layers* [Video]. Coursera. https://www.coursera.org/lecture/convolutional-neural-networks/pooling-layers-hELHk