A comparison of common classifier models for hand pose estimation for the New Zealand Sign Language alphabet

Ryan B right1\*

1 Department of Mechanical Engineering, University of Canterbury, Christchurch, New Zealand

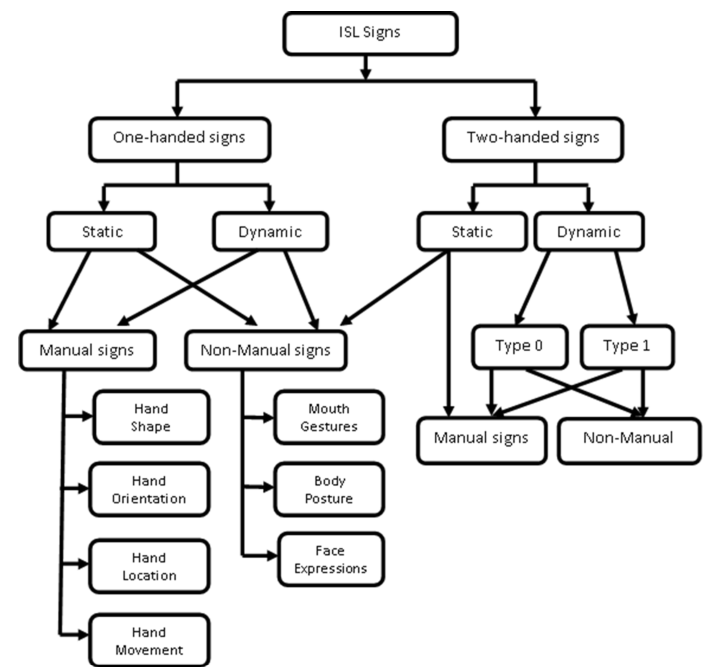
\* Corresponding Author Email: rbr115@uclive.ac.nz

|  |
| --- |
| *ABSTRACT:*  *A concise summary covering:*   * *The problem you address (e.g., NZSL gesture recognition).* * *Your approach (e.g., using MediaPipe and a neural network classifier).* * *Key results (accuracy, inference behavior).* * *Your main contribution (e.g., a live system for NZSL letter classification).* |

KEYWORDS: New Zealand Sign Language; MediaPipe Hands; Hand Pose Estimation

# Introduction

New Zealand Sign Language (NZSL) is one of New Zealand’s official languages, yet digital tools that provide accessibility and ease of social integration remain limited. There is a significant language barrier between people who rely on NZSL and those who don’t. Unlike spoken language, sign languages pose a unique set of challenges for translation tools. As shown in Figure 1, NZSL relies on precise hand positioning, movement, facial shapes, and body poses to communicate [1]. Additionally, dictionaries are challenging to use and develop due to NZSL not being a written language.



NZSL

Figure . Classification of sign language word forms. Adapted from [1]

This makes consistent, real-time classification a non-trivial task.

Previous efforts for other sign languages like American Sign Language (ASL) and Chinese Sign Language (CSL) show a wide range of approaches and methodologies. For example, the use of Hough transformations and CNNs [2], the use of 3D hand tracking gloves [3], and the use of various neural network models [4]. Notably, camera-based methods in papers reviewed in [1], all focus on single hand signs. This poses a challenge for NZSL which has a high amount of two handed signs.

Presented is a complete pipeline including data gathering, training and evaluation with additional scripts for evaluating training data balance and correcting camera lens distortion. The current system detects one or two handed static manual signs using MediaPipe to generate hand landmarks, and then a multilayer perceptron (MLP) network to classify the hand pose. The system is evaluated on validation data and real time webcam input, showing high accuracy in controlled environments and highlighting some key limitations.

# Methodology

## Environment

Table 1 lists the development environment used for this project.

Table . Development environment for project

|  |  |
| --- | --- |
| Part | Detail |
| Operating System | Windows 11 |
| Web Camera model | Built in 1080p webcam |
| Computer model | Toshiba Portege X30 E |
| CPU | Intel i5-8250U |
| Python Ver | 3.12.9 |
| TensorFlow Ver | 2.19 |
| OpenCV-Python Ver | 4.11 |
| MediaPipe Ver | 0.10.21 |
|  |  |

## Inference pipeline and architecture

The inference pipeline combines MediaPipe and a trained MLP classifier to process webcam input and detect hand poses in real time. Key steps in the pipeline are illustrated in Figure 2.

Figure . Flow of presented inference pipeline

The webcam video feed is loaded using the python library OpenCV-python. Then the lens distortion is corrected for using OpenCVs undistort() function and a precalibrated distortion matrix. The distortion matrix is calculated from a reference chess board pattern and measuring deviations from straight lines.

Google's MediaPipe Hands [5] library was selected for extracting landmark data. It provided a lightweight, real-time hand-tracking solution capable of detecting 21 landmarks per hand, Figure 3, with relatively high accuracy. Compared to training a custom model from scratch, MediaPipe reduced the problem from the domain of camera pixels to that of hand landmark coordinates.

A hand with dots and lines

AI-generated content may be incorrect.

Figure . Indexed Landmarks from MediaPipe Hand’s model.

To ensure the system is flexible and able to deal with different hand locations and sizes, a normalization algorithm is employed. Normalization begins by translating the coordinate system to the midpoint between the wrists, removing positional variance in the data set. Next, the maximum Euclidean norm from the origin to any landmark is used to scale the landmarks, achieving scale invariance while preserving relative geometry,   
Figure 4.

A comparison of a graph

AI-generated content may be incorrect.

Figure . Side by side comparison of x-y projection of original and normalized landmarks. Right hand drawn with skeleton.

One important edge case to cover in normalization is single hand signs. When there is only one hand detected, its landmarks get stored in place for both hands. This allows the normalization algorithm to function using the same rules for both single and double hands.

Lastly a neural network is employed to take the landmarks and classify the hand pose. Due to the structured, low-dimensional input and lack of temporal sequence data, an MLP neural network was selected

The implemented network has an input layer of 126 nodes, storing x,y,z coordinate data for all 21 landmarks for both hands. The hidden layers are outlined below.

* 256 node dense layer with ReLU activation
* 30% dropout layer
* 128 node dense layer with ReLU activation
* 30% dropout layer
* Output layer with Softmax activation

The output layer applies the Softmax activation to produce a class probability distribution. The class with the highest probability is selected and displayed in the interface.

## Data collection and processing

Due to the absence of publicly available datasets specific to New Zealand Sign Language (NZSL), a custom data collection tool was developed. This tool captures hand pose data using the same inference pipeline and saves the normalized landmarks from each detected frame as individual .json files.

A total of 4,400 labeled samples were collected, each containing normalized coordinates of 21 landmarks per hand, along with a label and timestamp. The data format for each entry is shown below:

{  
 "label": "A",   
 "left": [[x0,y0,z0], ..., [x20,y20,z20]],   
 "right": [[x0,y0,z0], ..., [x20,y20,z20]],   
 "timestamp": 123456789  
}

To mitigate class imbalance—which can bias the classifier—an approximately uniform distribution across all classes was targeted during data gathering. The final class distribution is shown in   
Figure 5.

A graph of blue bars

AI-generated content may be incorrect.

Figure . Frequency histogram of labeled landmark hand poses.

To ensure low false positive rates from the model, a large amount of null pose reference samples were needed. These samples were critical for distinguishing between intentional sign poses and idle hand positions. The null pose training data causes denser feature maps giving rise to better class boundary separation, as illustrated by Figure 6.

A diagram of different colored dots

AI-generated content may be incorrect.

Figure . Illustration of two feature maps showing three classes’ labeled data (Orange, Blue, and Grey) and the possible class boundaries between them (Black). (Left - loose fit , Right - Tight fit)

## Training

Of the full dataset, 80% of the samples were randomly allocated for training and 20% for validation. To increase the effective size and diversity of the training data, data augmentation techniques were applied. Each training sample was duplicated multiple times with slight variations, increasing the size of the training set by 300%. Augmentations included random translations to the left- and right-hand landmarks, followed by the addition of Gaussian noise to the (x,y,z) coordinates of each landmark. These transformations preserved the underlying hand pose while simulating natural variation and sensor noise, thereby improving model robustness.

The model was trained using the Adam optimizer, a sparse categorical cross-entropy loss function, a batch size of 32, and the model was trained over 10 epochs. Training was conducted using TensorFlow/Keras, and key metrics such as training loss, validation loss, and accuracy were tracked across all epochs.

Early stopping and learning rate scheduling were considered during model development but ultimately not applied, as the model converged reliably within a small number of epochs without signs of instability.

# Results

Present:

* Training/validation accuracy and loss plots.
* Confusion matrices and per-class performance.
* Comparison between different models (if evaluated).
* Live inference performance and failure modes.

# Discussion

Interpret your findings:

* Accuracy vs. generalization issues.
* Effectiveness of augmentation and normalization.
* Model overfitting or underfitting signs.
* Observations about specific signs (e.g., M vs. N).

# Conclusion

Summarize:

* What your system achieves.
* Limitations (e.g., low variability in training data).
* Possible extensions (e.g., full word recognition, deployment on mobile).

# References

[1] Wadhawan A, Kumar P (2021) Sign Language Recognition Systems: A Decade Systematic Literature Review. *Arch Computat Methods Eng* **28**, 785–813.

[2] Munib Q, Habeeb M, Takruri B, Al-Malik HA (2007) American sign language (ASL) recognition based on Hough transform and neural networks. Expert Syst Appl 32(1):24–37

[3] Oz C, Leu MC (2007) Linguistic properties based on American Sign Language isolated word recognition with artificial neural networks using a sensory glove and motion tracker. Neurocomputing 70(16):2891–2901

[4] Wu J, Tian Z, Sun L, Estevez L, Jafari R (2015) Real-time American Sign Language recognition using wrist-worn motion and surface EMG sensors. In: IEEE 12th international conference on wearable and implantable body sensor networks (BSN), pp 1–6

[5] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., & Grundmann, M. (2020). MediaPipe Hands: On-device real-time hand tracking. In *arXiv [cs.CV]*. http://arxiv.org/abs/2006.10214

Limitations and next steps:

* Overlapping hands and strong occlusion (media pipe issues)
* The rest of HOLM, movement or body position/ inclusion of pose, and facial expression
* I think that the rapid drop in val loss, and increase in training accuracy is due to there being redundant data
* A screenshot of a computer

  AI-generated content may be incorrect.

Context to cover