2026 S.T. Yau High School Science Award (Asia)

Research Report

Registration Number: **Comp-190**

Student: CHAN Wing Yan Abigail

Hong Kong International School

Hong Kong

Teacher: HOGAN Johnny

Science Teacher at Hong Kong International School

Hong Kong

Title of Research Report

DigiSign: A Landmark-based Framework for Real-Time Static Sign Language Recognition

Date

17 August 2026

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示

DigiSign: A Landmark-based Framework for Real-Time Static Sign Language Recognition

CHAN Wing Yan Abigail

Abstract

Automated sign language recognition remains challenging because many deep learning methods require large datasets, high computational resources, or hardware unsuitable for real-time assistive use. This study presents DigiSign, a landmark-based framework for real-time static American Sign Language recognition under low-data and CPU-only constraints. Instead of classifying raw RGB pixels, DigiSign uses MediaPipe Holistic to extract hand landmarks and converts them into a compact 114-dimensional geometric descriptor capturing relative hand coordinates, finger flexion, extension states, and palm geometry. This report evaluates a 10-sign ASL subset under controlled conditions of 100 training images per sign and 102 held-out test images, while the underlying DigiSign system was developed and tested with a 30-sign vocabulary and multiple signers to establish real-world feasibility. This enabled a comparison with two pixel-based baselines: a PyTorch LSTM classifier and a TensorFlow.js MobileNet-SSD detector. Because MediaPipe provides a pretrained hand-specific prior, the study is framed as a practical low-data deployment evaluation rather than a fully symmetric landmark-versus-pixel comparison. On identical CPU-only Apple M3 hardware, DigiSign achieved 68.6% strict accuracy and 0.747 macro-F1, outperforming the LSTM baseline by 29.4 percentage points under a paired McNemar's test. When DigiSign committed to a prediction, conditional accuracy rose to 89.7% at 76.5% coverage. The TensorFlow.js detector produced valid detections on only 5 of 102 test images, indicating deployment fragility in this setup. DigiSign also achieved 88.6 ms median latency, below the 100 ms interaction threshold, though tail latency still requires optimisation. Overall, DigiSign was the most accurate and practically deployable system evaluated, showing the value of pretrained landmarks for low-data, CPU-only static sign recognition.

Keywords: sign language recognition; American Sign Language; MediaPipe; hand landmarks; Random Forest; LSTM; transfer learning; low-data regime; edge AI; McNemar test

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

Acknowledgement

I thank Mr Johnny Hogan, Science, Hong Kong International School, for supervision and feedback throughout this project. The DigiSign implementation builds on the open-source sign-language recognition codebase by 209sontung (n.d.; see Section 4.2 and the Bibliography), which was substantially modified for this study; the LSTM and TensorFlow.js baselines were adapted starting from the open-source tutorials by Renotte (n.d.-a, n.d.-b; see Section 4.2 and the Bibliography).

AI tools used: Perplexity (Version/description: Multi-model AI research and agent assistant, as accessed via perplexity.ai. Perplexity does not expose a single static version number, since it dynamically routes tasks across multiple underlying AI models.), Deepseek-V4-Pro and Deepseek-V3.

Period of use: 26 January 2026 to 17 August 2026.
Frequency of use: Used intermittently during conceptual learning, implementation support, statistical checking, literature search, and grammar checks.
Supervising teacher approval: The use of AI tools was disclosed to and approved by the supervising teacher.

1. Conceptual learning: statistical methods. Both tools were used as explanatory tutors to help me understand the statistical procedures reported in Section 3.6.3. I asked for explanations of the Wilson score interval and why it is preferred to the Wald interval at small sample sizes; McNemar's test, the meaning of discordant pairs, and the difference between its exact binomial and continuity-corrected chi-squared forms; the Wilcoxon signed-rank test and rank-biserial correlation as its effect size; the Shapiro–Wilk normality test; Bonferroni correction for multiple comparisons; macro-averaged versus weighted F1; the logic of equivalence testing, including TOST, and why failure to reject a null hypothesis does not establish equivalence; and why Random Forest vote shares are not calibrated probabilities. The choice of which tests to apply, and the interpretation of every result, are my own; the AI's role was to explain the statistical principles so that I could apply them correctly and defend them.
2. Conceptual learning: model architectures. Both tools were used to help me understand the architectures discussed in Sections 2 and 3 before I selected and configured the models. This included how MediaPipe's hand module produces its 21 keypoints through a two-stage palm-detector-then-landmark-regressor design, and what the keypoint indexing convention means; how LSTM gating carries information across a sequence, and why a recurrent architecture is designed for temporal input; how the SSD detection head and the MobileNet backbone combine into a mobile-oriented detector; and how DETR reframes object detection as direct set prediction with a transformer encoder–decoder. DETR was investigated during this reading stage but was not implemented or evaluated in this study; it appears in Section 2.2 as context on the field's landscape only.

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示

3. Implementation support. Resolving software dependencies and drafting an evaluation harness script. I specified the required inputs, outputs, measurement fields, and evaluation procedure. The script automated batch evaluation of my trained models on my test images and recorded, per image, the ground-truth sign, predicted sign, model confidence, wall-clock inference latency in milliseconds, and outcome category. I reviewed the script, ran the evaluations, and checked its outputs against the models' interactive behaviour and selected manual records. The resulting per-image spreadsheets are disclosed as the raw measurement files used for Chapter 5. The AI did not decide which models to evaluate, which metrics to report, which test images to use, or how to interpret the results. The final script and its outputs were validated by me against manual records.

4. Analysis verification. Deepseek was used as a verification aid to assist with Python-based recalculation of the headline statistics, confidence intervals, and significance tests in Chapter 5 from the submitted .xlsx result files. I checked the Python logic, compared the recomputed outputs with the report tables and figures, and corrected discrepancies where appropriate. The final statistical choices, reported values, and interpretations are my responsibility.

5. Critique and revision. Perplexity was used to help identify possible weaknesses, ambiguities, and unsupported claims in earlier drafts for my review. I checked these suggestions against the raw result files and accepted only those that were supported by the evidence. This process helped improve the discussion of the treatment of the TensorFlow.js anomaly, but the final decisions about what to report and how to interpret the anomaly were made by me. Deepseek was also used for language polishing and grammar checks of the author-drafted text.

6. Literature and fact-checking. Deepseek was used to generate literature search leads, to suggest checks for the ASL citation forms of the ten signs against ASL University/Lifeprint, and to check factual claims against cited sources, including the Hong Kong Census and Statistics Department report cited in Section 1.1. Every source in the Bibliography was located and read by me; no citation was accepted on the AI's word alone.

Representative AI chat records are attached separately as Comp-190-AI Usage Attachment, submitted alongside this report.

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示

Commitments on Academic Honesty and Integrity

We hereby declare that we

1. are fully committed to the principle of honesty, integrity and fair play throughout the competition.
2. perform the research work ourselves and thus truly understand the content of the work.
3. observe the common standard of academic integrity adopted by most journals and degree theses.
4. have declared all the assistance and contribution we have received from any personnel, agency, institution, etc. for the research work.
5. have truthfully disclosed all AI tool usage in the research report and obtained prior approval from supervising teachers. All supporting documentation has been submitted as required. We hereby affirm that no improper use of AI has occurred at any stage of the research process.
6. undertake to avoid getting in touch with assessment panel members in a way that may lead to direct or indirect conflict of interest.
7. undertake to avoid any interaction with assessment panel members that would undermine the neutrality of the panel member and fairness of the assessment process.
8. observe the safety regulations of the venue(s) where we conduct the experiment(s), if applicable.
9. observe all rules and regulations of the competition.
10. agree that the decision of YHSA(Asia) is final in all matters related to the competition.

We understand and agree that failure to honor the above commitments may lead to disqualification from the competition and/or removal of award, if applicable; that any unethical deeds, if found, will be disclosed to the school principal of the team member(s) and relevant parties if deemed necessary; and that the decision of YHSA(Asia) is final and no appeal will be accepted.

(Signatures of full team below)

X
Name of team member: Chan Wing Yan Abigail

X
Name of team member:

X
Name of team member:

X
Name of supervising teacher:
Mr Johnny Hogan

Noted and endorsed by

(signature)

Name of school principal: Ms Aimmee Kellar

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示

Table of Contents

Abstract 2
Acknowledgement 3
Commitments on Academic Honesty and Integrity 5
1 Introduction 8
1.1 Background and Motivation 8
1.2 The Computational Bottleneck of Pixel-Based Recognition 8
1.3 The Landmark-Based Alternative, and Its Pretrained Prior 9
1.4 Research Questions and Hypotheses 10
1.5 Contributions 11
1.6 Scope 11
2 Literature Review 12
2.1 Sign Language Recognition: Scope and Challenges 12
2.2 The Pixel-Based Deep-Learning Paradigm 13
2.3 The Landmark-Based Geometric Approach 14
2.4 Random Forests for Low-Dimensional Classification 15
2.5 Edge AI and the Real-Time Constraint 15
2.6 Statistical Comparison of Classifiers 16
2.7 Research Gap 16
3 Methodology 16
3.1 System Overview 17
3.2 Landmark Extraction 18
3.3 Geometric Feature Engineering 18
3.3.1 Scale normalisation 19
3.3.2 Finger flexion angles 19
3.3.3 Finger-state and geometric descriptors 19
3.3.4 Final feature vector 20
3.4 Classification: Random Forest 21
3.5 Baseline Models and Design Asymmetries 21
3.6 Evaluation Protocol 23
3.6.1 Decision rule and outcome taxonomy 23
3.6.2 Metrics 24
3.6.3 Statistical analysis 25
4 Experiment 25
4.1 Sign Set and Dataset Construction 25
4.2 Training and Implementation Environment 27

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示

4.3 Test Protocol and Hardware 28
4.4 A Note on the TensorFlow.js Baseline 29
5 Results 29
5.1 Overall Recognition Accuracy 29
5.2 Per-Sign Performance 31
5.3 Statistical Comparison of Accuracy (RQ1) 34
5.4 Inference Latency (RQ2) 35
5.5 The TensorFlow.js Anomaly 37
5.6 Supplementary LSTM Runs: Threshold Sensitivity and Mode Collapse 37
5.7 The Accuracy–Efficiency Trade-off 39
6 Discussion 40
6.1 Why the Pretrained Landmark Pipeline Beat Task-Trained Pixels Here 40
6.2 The Abstention Behaviour as a Deployment Feature 41
6.3 The TensorFlow.js Anomaly: Candidate Explanations 42
6.4 Limitations 43
6.5 Future Work 44
7 Conclusion 45
Bibliography 48

2026 S.-T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示

1 Introduction

1.1 Background and Motivation

Approximately 430 million people (more than 5% of the world population) live with disabling hearing loss, a figure the World Health Organization projects will grow substantially by 2050 (WHO, 2026). Within this population, roughly 70 million deaf people communicate primarily through more than 300 distinct sign languages (United Nations, n.d.). In Hong Kong, the Census and Statistics Department's Special Topics Report No. 63 reports 47,900 persons with hearing difficulty (about 0.6% of the population), with a further 53,400 hearing-aid users counted as a separate, overlapping category (Census and Statistics Department, 2021). For these communities, everyday interactions with the hearing world are mediated by a persistent communication barrier, because sign language fluency among the hearing population remains rare. Automatic sign language recognition (SLR), which maps visual observations of a signer to linguistic units, has long been proposed as a technological bridge across this barrier (Bragg et al., 2019).

Two cautions frame that motivation honestly. First, Bragg et al. (2019) argue that SLR success criteria need to be set with deaf communities; no deaf signers participated in the design, data collection or evaluation of this study, so the work should be read as a methodological contribution toward assistive technology, not as assistive technology itself. Second, a 10-sign, single-signer benchmark cannot by itself serve the population figures above; its role is to answer a narrow representational question rigorously (Section 1.4).

For an SLR system to function as genuine assistive technology rather than a laboratory demonstration, it must run in real time on hardware that its users can actually afford: phones, tablets, browsers, and low-cost single-board computers, typically without a discrete GPU. Human–computer interaction research has converged on roughly 100 milliseconds as the threshold below which a system response feels instantaneous (Miller, 1968; Card, Moran, & Newell, 1983; Nielsen, 1993). This latency budget, evaluated on CPU-only hardware, is the engineering constraint at the heart of this study.

1.2 The Computational Bottleneck of Pixel-Based Recognition

The dominant contemporary approach to SLR feeds raw RGB pixels to deep neural networks. A single modest 224×224×3 input frame comprises 150,528 values, and modern architectures

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖公示

process such frames through millions of learned parameters. Convolutional neural networks (CNNs) established this paradigm (Pigou, Dieleman, Kindermans, & Schrauwen, 2015), and transformer-based architectures have since pushed accuracy further at even greater computational cost (Vaswani et al., 2017; Dosovitskiy et al., 2021; Camgöz, Koller, Hadfield, & Bowden, 2020). These models excel on benchmark leaderboards, but their compute requirements bind them to GPUs or cloud back-ends. This creates what may be called an accessibility paradox: the very technology intended to include deaf and hard-of-hearing users risks becoming available only to those with high-end hardware or reliable high-bandwidth connectivity: a new digital divide layered on top of the communication barrier itself.

1.3 The Landmark-Based Alternative, and Its Pretrained Prior

An alternative approach discards pixels almost entirely at the classification stage. MediaPipe Hands (Zhang et al., 2020), running on-device in real time, reduces each hand to 21 skeletal keypoints. From these keypoints, a small set of geometric quantities (normalised coordinates, finger flexion angles, inter-fingertip distances) can be engineered deterministically, compressing the input from over 150,000 pixel values to 114 features: a reduction of more than three orders of magnitude. A classical, CPU-friendly classifier such as a Random Forest (Breiman, 2001) can then operate on this compact representation.

It is important to clarify at the outset that the landmark extractor was not trained on this study's data. Instead, MediaPipe's hand model was trained by its authors on a corpus of over one hundred thousand annotated hand images (Zhang et al., 2020). A comparison between a landmark pipeline and pixel models trained only on this study's ~1,000 images is therefore *not* a comparison of representational paradigms on equal data; it is a comparison between (a) a large pretrained, hand-specific prior plus a small task-specific classifier and (b) pixel models that must learn hand appearance from the scarce task data alone. That asymmetric question, which is the question any practitioner with a small dataset and a CPU actually faces, is the one this study is designed to answer. The open empirical question is whether the aggressive geometric compression discards information that pixel models could exploit, or whether, for static handshapes at least, pretrained geometry is sufficient and superior in this low-data regime.

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

1.4 Research Questions and Hypotheses

This study places the systems on a common footing (identical signs, identical task-specific training images, identical held-out test set, identical hardware) and answers the following questions with formal statistical inference:

RQ1 (Accuracy). How does the recognition accuracy of the pretrained-landmark DigiSign pipeline compare with that of (a) an LSTM network, reimplemented in PyTorch and operating on raw pixel/video frames, and (b) a MobileNet-SSD CNN deployed via TensorFlow.js, both fine-tuned on the task images alone, when all are evaluated on the same test set? Hypothesis H1 (null hypothesis): the paired accuracy difference is zero (exact McNemar test, two-sided, $\alpha = 0.05$). Two methodological notes accompany H1. First, a claim that two classifiers are *comparable* cannot be established by a mere failure to reject this null (absence of evidence is not evidence of equivalence) and would require an equivalence procedure such as two one-sided tests (TOST) against a pre-specified margin; since the results in Chapter 5 reject the null decisively in both comparisons, the superiority framing suffices for the outcomes actually obtained. Second, no a priori power analysis was conducted; at $n = 102$ the paired design has adequate power only against large differences, a limitation acknowledged in Section 6.4.

RQ2 (Latency). Does each pipeline meet the 100 ms real-time threshold on CPU-only consumer hardware, and how do their latency distributions compare? Hypothesis H2: DigiSign meets the threshold and exhibits lower latency than both pixel-based baselines (paired Wilcoxon signed-rank test). Meeting the threshold is assessed both at the median and in the tail (proportion of inferences under 100 ms; 95th percentile), since tail latency governs the perceptual experience (Section 2.5).

RQ3 (Reliability of prediction). When each system commits to a prediction, how trustworthy is it? This is quantified by coverage (the proportion of images on which a prediction is made at the operating threshold) and conditional accuracy (accuracy among committed predictions), with confidence intervals on both. Because these quantities depend on each system's operating threshold, and the thresholds used here were set ad hoc rather than tuned on a validation set, RQ3 is answered at the deployed operating points and interpreted with that caveat (Sections 3.6.1 and 6.4).

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖参赛作品

Notably, the results answer RQ1 in an unexpected direction: DigiSign proved significantly more accurate than both baselines, and H2 is only partially supported. Chapter 5 and Section 6 discuss both departures honestly.

1.5 Contributions

The contributions of this work are:

(1) a same-task-data, same-hardware benchmark of a pretrained-landmark pipeline against two pixel-based baselines (an LSTM classifier and an SSD detector) trained from the task data alone, on 10 static ASL signs, with the asymmetry in effective training data stated explicitly rather than obscured; (2) a fully specified geometric feature construction (114 dimensions from 21 keypoints per hand) with every transformation given in closed form and the final 114-dimensional input verified against the project's data audit (3.3.4); (3) a statistical evaluation protocol (Wilson confidence intervals, exact McNemar tests with effect sizes, and paired Wilcoxon tests on latency) of a rigour typically absent from comparable student work; (4) an honest documentation and root-cause analysis of a baseline that succeeded in interactive testing yet failed anomalously in final batch evaluation, treated as a finding about deployment fragility rather than as discardable noise; and (5) an accounting of the accuracy–latency–deployability trade-off relevant to real assistive deployment on edge hardware (Table 5.5), reported qualitatively where exact model sizes were not measured during development.

1.6 Scope

The study is deliberately restricted to isolated, static signs (handshapes whose meaning does not depend on motion trajectory) so that the representational question (pretrained geometry versus task-trained pixels) can be examined without the confound of temporal modelling. This study contributes a practical MediaPipe and Random Forest sign-recognition interface developed through iterative 30-sign prototype testing with image and video samples from multiple individuals, followed by a controlled comparison against two baseline systems. The 10-sign, single-signer controlled evaluation was a deliberate design choice to ensure that differences in accuracy between models could be attributed to the representational approach rather than to differences in dataset size, signer variability, or vocabulary coverage. The

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

controlled evaluation in this report uses a 10-sign ASL subset of graded visual difficulty (Section 4.1) so that DigiSign, the LSTM baseline, and the TensorFlow.js baseline can be compared on a common recognition task. All data come from a single signer; the evaluation therefore measures within-signer generalisation only. Dynamic signs, continuous signing, facial grammar, signer-independent evaluation, and sentence-level translation are out of scope and are discussed as future work.

2 Literature Review

This review situates DigiSign within five strands of prior work: the problem landscape of SLR and its recognised challenges (2.1); the pixel-based deep-learning paradigm and its computational cost (2.2); the landmark-based geometric approach on which this study builds (2.3); the Random Forest classifier DigiSign adopts (2.4); and the edge-computing constraint that motivates the whole design (2.5). Section 2.6 reviews how classifiers should be compared statistically, and Section 2.7 states the gap this study addresses.

2.1 Sign Language Recognition: Scope and Challenges

SLR research spans four decades and several sensing modalities. Early systems used instrumented gloves and wearable sensors, which achieved respectable accuracy but at the cost of intrusiveness and expense (Kudrinko, Flavin, Zhu, & Li, 2021). Vision-based approaches removed the wearable: Starner, Weaver and Pentland (1998) demonstrated real-time American Sign Language (ASL) recognition from desk- and hat-mounted cameras using hidden Markov models, establishing the camera-only paradigm that dominates today. The field conventionally distinguishes isolated SLR (classifying a single, pre-segmented sign) from continuous SLR over unsegmented signing streams (Koller, Forster, & Ney, 2015), and static handshapes from dynamic signs that depend on motion. Isolated static recognition, the setting of this study, is widely treated as the foundational building block of the larger problem (Rastgoo, Kiani, & Escalera, 2021).

Modern surveys converge on a common list of obstacles to real-world deployment: signer variability in hand shape, skin tone and style; sensitivity to lighting, background and occlusion; scarcity of large labelled datasets, particularly outside ASL; and the difficulty of real-time processing on modest hardware (Cheok, Omar, & Jaward, 2019; Koller, 2020; Rastgoo et al., 2021). Benchmark datasets such as WLASL (Li, Rodriguez Opazo, Yu, & Li,

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

2020) and MS-ASL (Joze & Koller, 2019) have exposed how sharply accuracy drops when models face unseen signers and in-the-wild conditions. Bragg et al. (2019) further stress that SLR is an accessibility problem whose success criteria must ultimately be set by deaf communities, not by leaderboard metrics alone, a perspective that motivates this study's focus on deployability rather than raw accuracy, and that also bounds its claims (Section 1.1).

2.2 The Pixel-Based Deep-Learning Paradigm

CNNs learn feature hierarchies directly from raw pixels and were first applied to sign and gesture recognition by Pigou et al. (2015). On static ASL alphabet datasets, CNNs routinely report near-ceiling accuracy: Kasapbaşı, Elbushra, Al-Hardanee and Yilmaz (2022) exceed 99% with a bespoke CNN, and Alsharif, Altaher, Altaher, Ilyas and Alalwany (2023) report similar figures across transfer-learned architectures. Such results, however, are typically obtained on large single-source datasets with abundant per-class examples and evaluated on identically distributed test splits; they say little about robustness under distribution shift, an issue that becomes central in Section 6.3 of this report.

Architecturally, ResNet introduced the residual connections that made very deep networks trainable (He, Zhang, Ren, & Sun, 2016), while the SSD detection head (Liu et al., 2016) paired with lightweight MobileNet backbones (Howard et al., 2017) brought detection within reach of mobile hardware; this SSD-MobileNet combination underlies this study's TensorFlow.js baseline (Section 3.5). Recurrent architectures offer an alternative route to modelling pixel input: the Long Short-Term Memory (LSTM) network (Hochreiter & Schmidhuber, 1997) processes a sequence of frames through gated memory cells, and CNN/MediaPipe-LSTM hybrids have been applied to sign and gesture recognition (Sundar & Bagyammal, 2022; Section 2.3). This study's second pixel-based baseline is an LSTM, reimplemented in PyTorch, that operates on raw pixel/video frames rather than pre-extracted landmarks (Section 3.5); unlike a detector, it performs classification only, so it does not carry the localisation-plus-classification burden that a true object detector such as the SSD-MobileNet baseline does. Separately, transformers, built on self-attention (Vaswani et al., 2017), entered vision through the Vision Transformer (Dosovitskiy et al., 2021) and object detection through DETR, which reframes detection as direct set prediction with a ResNet-50 backbone feeding a transformer encoder–decoder (Carion et al., 2020); this study does not evaluate a transformer-based detector, so DETR is noted here only as part of the field's

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

landscape, not as a baseline. In SLR specifically, transformer models define the research frontier (Camgöz et al., 2020; De Coster, Van Herreweghe, & Dambre, 2020), but their computational appetite, like that of deep CNN and recurrent pixel models generally, ties them to GPU infrastructure and to correspondingly large training sets: pixel-based deep models of every architecture family are documented as data-hungry relative to classifiers built on compact, engineered features (Rastgoo et al., 2021), which bears directly on this study's low-data regime (Section 3.5). A single 224×224×3 frame already contains 150,528 input values, and both self-attention and recurrent processing over many such frames are computationally demanding; on CPU-only edge devices such models frequently miss real-time budgets.

2.3 The Landmark-Based Geometric Approach

MediaPipe provides a cross-platform framework for real-time perception pipelines (Lugaresi et al., 2019). Its hand module predicts 21 three-dimensional keypoints per hand from a single RGB frame using a two-stage palm-detector-then-landmark-regressor design (Zhang et al., 2020). Because landmarks abstract away appearance, the representation should inherit whatever robustness to skin tone, background texture and lighting variation the landmark extractor itself possesses. It must be stressed that this is an *inherited and, in this study, unverified* property: it rests on the extractor's own published evaluations, not on any test conducted here (the single-signer data cannot probe demographic robustness at all), so demographic robustness is treated as an assumption to be verified, not a demonstrated property. Section 6.4 records this as a limitation rather than a claim.

A growing body of work couples MediaPipe landmarks with lightweight classifiers. Halder and Tayade (2021) report high accuracy on static signs across several sign languages using landmark features with classical machine learning; the venue is not indexed in the major citation databases, so the paper is cited here as an existence proof of the approach rather than as evidence for its accuracy figures. Samaan et al. (2022) show that landmark streams alone, without any raw pixels, suffice for dynamic gesture recognition with recurrent networks, and Sundar and Bagyammal (2022) apply MediaPipe with an LSTM to the ASL alphabet. Related evidence that decision-forest methods handle skeletal input at production scale comes from the Kinect body-pose system, which classified body parts from depth images with random forests in real time (Shotton et al., 2011). What this literature generally lacks, however, is controlled comparison: landmark systems are evaluated in isolation, on bespoke datasets, and

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

reported as single-point accuracies without uncertainty quantification (Madhan et al., 2025, is a recent exception in framing but not in statistical treatment).

2.4 Random Forests for Low-Dimensional Classification

The Random Forest (Breiman, 2001) is an ensemble of decision trees, each grown on a bootstrap resample of the training data with random feature subsampling at every split; predictions are aggregated by majority vote. The combination of bagging and feature decorrelation reduces variance without materially increasing bias, and the method is famously robust on low-dimensional tabular data with minimal tuning. Three properties make it apt for DigiSign: CPU-efficient inference (tree traversal involves no floating-point convolution), native handling of heterogeneous feature types (coordinates, angles, booleans), and a vote-share score that can be used to rank prediction confidence. On the last point a correction to common practice is needed: forest vote proportions are *not* in general well-calibrated probability estimates: they are characteristically distorted toward the centre of the [0,1] range unless post-hoc calibration is applied (Niculescu-Mizil & Caruana, 2005). DigiSign therefore treats the vote share as an uncalibrated confidence score for abstention gating, not as a calibrated probability, and Section 5.2 examines empirically how well that score separates correct from incorrect predictions. The implementation used here is from scikit-learn (Pedregosa et al., 2011).

2.5 Edge AI and the Real-Time Constraint

Assistive recognition is only useful if it runs where its users are: in browsers, on phones, and on low-cost boards, under thermal and battery limits. The TinyML literature articulates how tight these budgets are (David et al., 2021), and TensorFlow.js demonstrates that in-browser inference, with its privacy and zero-install advantages, is practical for moderate-sized models (Smilkov et al., 2019). The interaction literature supplies the target: responses under roughly 100 ms are perceived as instantaneous (Miller, 1968; Card et al., 1983; Nielsen, 1993). This study therefore reports not only mean latency but medians, interquartile ranges, 95th percentiles and the proportion of inferences under budget, since tail latency is what breaks the real-time illusion, and evaluates everything on CPU-only consumer hardware.

2026 S.-T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

2.6 Statistical Comparison of Classifiers

Comparing two classifiers by eyeballing two accuracy numbers is statistically unsound: on a shared test set the predictions are paired, and the appropriate procedure is McNemar's test on the discordant pairs, the cases where exactly one model is correct (McNemar, 1947). Dietterich (1998) identifies McNemar's test as the recommended choice when each algorithm can be trained once, and Demšar (2006) establishes non-parametric procedures as the standard for classifier comparison more broadly. Interval estimates for proportions follow Wilson (1927), whose score interval remains accurate at small n where the naive Wald interval fails. Where a *comparability* (rather than superiority) claim is at stake, the correct instrument is an equivalence test, for example two one-sided tests (TOST) against a pre-specified margin, since failure to reject a point null does not establish similarity. These are precisely the tools adopted in Section 3.6.3, alongside effect sizes, so that practical significance is never conflated with statistical significance.

2.7 Research Gap

Across this literature, pixel-based models are optimised for accuracy on capable hardware, landmark-based systems are reported in isolation with single-point estimates, and statistical inference is rare in both strands, particularly in student and applied work, where a single accuracy figure on a single split is the norm. Few studies compare a pretrained-landmark pipeline and task-trained pixel models on a common footing (identical signs, identical task data, identical hardware) and fewer still report the comparison with confidence intervals, paired significance tests and effect sizes, or state clearly what the pretrained extractor contributes to the contest. DigiSign fills exactly this gap for isolated static sign recognition on edge-class hardware.

3 Methodology

This chapter formalises the systems under comparison. Section 3.1 gives an overview of the DigiSign pipeline; 3.2 describes landmark extraction; 3.3 derives the geometric feature vector in closed form; 3.4 specifies the Random Forest classifier; 3.5 specifies the two pixel-based baselines and the design asymmetries their choice entails; and 3.6 defines the evaluation protocol, the deployment decision rule, and the statistical analysis plan.

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

3.1 System Overview

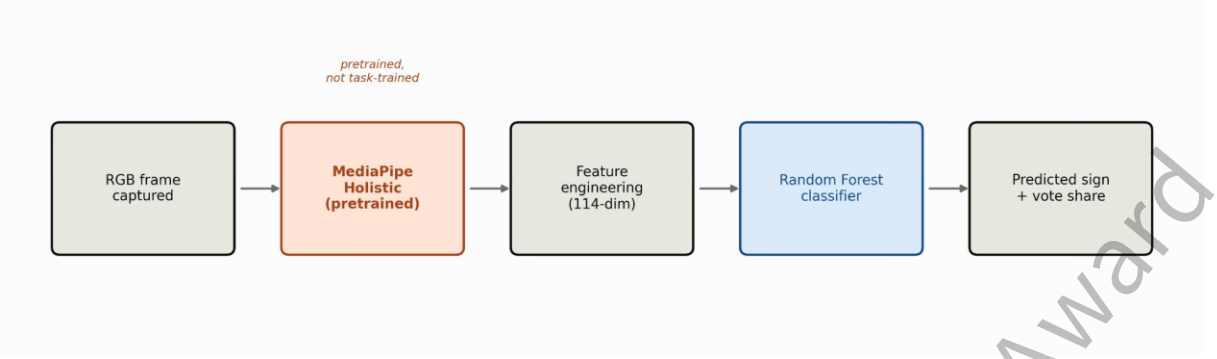
DigiSign is implemented as a real-time camera pipeline in Python: OpenCV captures webcam frames and renders the on-screen interface (a single display window, which is a proof-of-concept vehicle rather than a contribution of this study), and the system is designed for eventual deployment on a Raspberry Pi with a camera and touchscreen as a portable hardware-software unit. Within that pipeline, DigiSign transforms an input frame into a predicted sign through four stages: (i) a single RGB frame is captured; (ii) MediaPipe Holistic estimates landmarks (its Hands component supplies the 21 keypoints per hand, and its face landmarks are used solely to compute one hand-to-mouth distance feature, Section 3.3.3); (iii) a deterministic feature-engineering step converts the raw keypoints into a compact, scale-invariant descriptor; and (iv) a Random Forest classifier maps this descriptor to one of the target sign classes together with a vote-share confidence score. The implemented DigiSign prototype supports 30 sign classes. DigiSign was developed through an iterative prototyping process rather than a single isolated experiment. Over several months, image and video samples from multiple individuals were used to test recognition reliability, latency, and interface usability across a broader 30-sign prototype vocabulary. These iterations informed the final system design: MediaPipe hand-landmark extraction combined with a lightweight Random Forest classifier.

The experiments below evaluate a 10-sign subset shared with the two baseline systems to allow fair comparison under identical test conditions.

Stages (iii)–(iv) contain no pixel-level convolution, which is the source of the framework's computational economy; stage (ii) is the pretrained component whose role in the comparison is discussed in Sections 1.3 and 6.1.

Figure 3.1 The DigiSign pipeline. Only the landmark extractor touches pixels; all downstream computation operates on a 114-dimensional geometric descriptor.

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖公示



3.2 Landmark Extraction

Each frame is processed by MediaPipe Holistic (Lugaresi et al., 2019; Zhang et al., 2020), which returns up to 21 keypoints per hand. Keypoint i is given in normalised image coordinates $p_i = (x_i, y_i, z_i)$ for i = 0, 1, …, 20, following the standard MediaPipe hand topology: index 0 is the wrist, indices 1–4 the thumb, 5–8 the index finger, 9–12 the middle finger, 13–16 the ring finger, and 17–20 the little finger. The wrist (0), index-finger MCP joint (5) and little-finger MCP joint (17) anchor the normalisation below. When a hand is not detected, its feature block is zero-filled and a presence flag is set to 0, so missing hands are represented explicitly rather than imputed.

3.3 Geometric Feature Engineering

The core of DigiSign is a hand-crafted transformation from 21 raw keypoints per hand to a compact descriptor invariant to hand position and scale, and largely invariant to camera distance. The design goal is to encode explicitly the quantities a human uses to read a handshape (which fingers are extended, how bent each finger is, how the fingers are spread) rather than requiring the classifier to rediscover them from pixels. Two known fragilities of this construction are disclosed up front and revisited in Section 6.4: the finger-extension test below is defined in image coordinates and is therefore orientation-dependent (it can fail for strongly rotated or inverted hands), and the normalised image coordinates in which all quantities are computed are distorted by the frame's aspect ratio, while the palm-width scale shrinks under out-of-plane rotation (foreshortening of the p_5–p_{17} segment), inflating all normalised features.

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖公示

3.3.1 Scale normalisation

A palm-width scale is defined from the distance between the index and little-finger MCP joints:

$$s = \max(\| p_5 - p_{17} \|, \varepsilon) \quad (3.1)$$

where $\|\cdot\|$ is the Euclidean norm in the image plane and $\varepsilon = 10^{-6}$ prevents division by zero. Each keypoint is re-expressed relative to the wrist and divided by s, giving scale- and translation-invariant relative coordinates:

$$r_i = (p_i - p_0) / s , \; i = 0, \ldots, 20 \quad (3.2)$$

The two-dimensional relative coordinates contribute $21 \times 2 = 42$ features per hand.

3.3.2 Finger flexion angles

For each of the index, middle, ring and little fingers, a flexion angle is computed at the PIP joint from the interior angle formed by the MCP, PIP and fingertip keypoints. For three points a, b, c with b the vertex:

$$\theta = \arccos((a - b) \cdot (c - b) / (\|a - b\| \, \|c - b\|)) \quad (3.3)$$

A near-zero angle indicates a straight (extended) finger; a large angle indicates a bent finger. These contribute 4 features per hand.

3.3.3 Finger-state and geometric descriptors

Five boolean finger-extension states (thumb, index, middle, ring, little) are derived by comparing fingertip and joint positions in image coordinates; a finger is deemed extended when its tip lies above its PIP joint, and the thumb uses a handedness-aware horizontal test with a distance-from-wrist check. (As noted above, this test assumes a broadly upright hand and is not rotation-invariant.) Their sum gives an open-finger count. Scale-normalised global descriptors summarise the posture:

$$\text{spread} = \| p_8 - p_{20} \| / s , \; d(\text{thumb,index}) = \| p_4 - p_8 \| / s \quad (3.4)$$

where spread is the index-to-little fingertip distance and d(thumb,index) the thumb-to-index-tip distance. A palm-upright proxy (the vertical component of the wrist-to-middle-fingertip direction) and, where the face is detected, a normalised hand-to-mouth distance capture

2026 S.T. Yau High School Science Award 仅用于2026年丘成桐中学科学奖论文公示

orientation and body-relative position, the latter disambiguating signs articulated near the face and the reason the pipeline runs MediaPipe Holistic rather than the Hands module alone. Together with a handedness indicator, these contribute 7 finger-state/count features per hand (five extension booleans, the open-finger count, and the handedness indicator) and 4 geometric descriptors per hand (spread, thumb–index distance, palm-upright proxy, hand-to-mouth distance).

3.3.4 Final feature vector

The complete per-hand descriptor concatenates the 42 relative coordinates, 4 flexion angles, 7 finger-state/count features and 4 geometric descriptors, giving $F = 42 + 4 + 7 + 4 = 57$ features per hand. The dual-hand descriptor concatenates the left- and right-hand vectors for a total of $2 \times 57 = 114$ geometric features. Table 3.1 summarises the composition. This is a reduction of more than three orders of magnitude relative to the 150,528 values of a raw 224×224×3 frame.

This dimensionality is confirmed against the project's own data audit: the underlying dataset stores 118 CSV columns per example, comprising exactly these 114 model-input features (57 per hand × 2) plus 4 auxiliary columns (image identifier, sign label, and the two hand-presence flags of Section 3.2, carried as metadata rather than appended to the feature vector the Random Forest consumes). Every mention of "114" in this report refers to the model's actual input dimensionality.

Table 3.1 Composition of the DigiSign geometric feature vector.

Feature group	Per hand	Description
Relative coordinates	42	Wrist-anchored (x, y) of 21 keypoints, scale-normalised by palm width (Eq. 3.2)
Finger flexion angles	4	PIP interior angles of index–little fingers (Eq. 3.3)
Finger states + count	7	5 boolean extension states, open-finger count, handedness
Geometric descriptors	4	Spread, thumb–index distance, palm-upright proxy, hand–mouth distance
Total per hand	57	Total per hand: 57. Concatenated left and right hand descriptors give 114 model-input features.

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

3.4 Classification: Random Forest

Classification is performed by a Random Forest (Breiman, 2001), an ensemble of B decision trees. Each tree t_b is grown on a bootstrap resample of the training set; at each node a random subset of features is evaluated and the split maximising the decrease in Gini impurity is chosen. For a node with class proportions π_k:

$G = 1 - \Sigma_k \pi_k^2$ (3.5)

At inference each tree casts a vote and the forest returns the majority class with an associated vote share:

$\hat{y} = \text{argmax}_c \Sigma_b 1[t_b(x) = c]$, $\hat{p}(c|x) = (1/B) \Sigma_b 1[t_b(x) = c]$ (3.6)

where 1[·] is the indicator function. The classifier uses B = 400 trees, maximum depth 18, minimum 2 samples per leaf, and balanced-subsample class weighting; the random seed is fixed at 42. These settings are taken directly from the training implementation (src/train_sign_classifier.py); they were not tuned on a held-out validation set, and no validation split existed in this study, a limitation shared by the choice of every decision threshold in Section 3.6.1 and recorded in Section 6.4. The trained model, class list and feature layout are serialised with joblib for deployment.

3.5 Baseline Models and Design Asymmetries

DigiSign is benchmarked against two pixel-based baselines representing opposite ends of the deep-learning complexity spectrum, both fine-tuned on the same task images as DigiSign's classifier.

Paradigm B: LSTM. A Long Short-Term Memory network (Hochreiter & Schmidhuber, 1997), reimplemented in PyTorch and operating on raw pixel/video frames rather than pre-extracted landmarks. The starting point was the open-source ActionDetectionforSignLanguage tutorial codebase (Renotte, n.d.-a), which was substantially modified for this study, including reimplementation in PyTorch (the evaluated checkpoint, 99_model.pt, is a PyTorch state file). Unlike the tutorial's original design, which classifies short sequences derived from landmarks, this study's LSTM classifies directly from pixel input, which is what preserves this study's pixel-versus-landmark comparison (Section 1.3). The checkpoint evaluated is 99_model.pt (100 training epochs), applied with a confidence threshold of 0.30. Because the model is a classifier rather than a detector, it does not localise

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖公示

a sign within the frame; it outputs a class-probability distribution over the training vocabulary, and, following the codebase's inherited terminology, an image on which no class clears the confidence threshold is recorded as a detection failed outcome even though no spatial detection is attempted (Section 3.6.1).

The LSTM's exact architecture (number of recurrent layers, hidden size, and the resulting parameter count; LSTMs of this kind for short pixel-sequence classification are typically far smaller than a detection transformer's tens of millions of parameters, but the exact figure was not measured), any pretrained-weight provenance, and its training hyperparameters (batch size, learning rate, augmentation, framework version, beyond the 100 training epochs already stated) were not logged during development and are not available for this report; this is recorded as a limitation on reproducibility alongside the corresponding gap for the TFJS baseline (Section 4.2, Section 6.4). Two further open questions about this baseline are addressed elsewhere rather than resolved here: which specific training run of the LSTM produced the primary results of Sections 5.1–5.3, and with what full training class list, is discussed as an unresolved provenance caveat in Section 5.6, with its result file (test_results_sign_detr.xlsx) submitted for verification; and how a single static test image is converted into the frame sequence the LSTM architecturally expects as input is noted as a design mismatch above and in Section 6.4.

Paradigm C: TensorFlow.js CNN. A MobileNet-SSD object-detection model (Howard et al., 2017; Liu et al., 2016) exported as a TensorFlow SavedModel and deployable in the browser via TensorFlow.js (Smilkov et al., 2019), evaluated at 640×480 input with a confidence threshold of 0.80. This model follows the widely used real-time sign detection tutorial architecture (Renotte, n.d.-b) and represents the zero-install, browser-deployable end of the spectrum.

The TFJS model's exact architecture variant (e.g. SSD-MobileNet-V2 FPNLite at a specific input resolution), the checkpoint it was initialised from, and its training step count were not logged during development and are not available for this report (Section 4.2, Section 6.4); the provenance of its 0.80 confidence threshold is stated above.

Three design asymmetries in this choice of baseline should be acknowledged, as they constrain what the comparison can show. First, only the TensorFlow.js baseline is an object *detector*, solving localisation plus classification, a strictly harder task than DigiSign's pure

2026 S.-T. Yau High School Science Award 仅用于2026丘成桐中学科学奖公众号

classification; its bounding-box annotations were produced by the author using the Label Studio annotation tool (Section 4.2). The LSTM baseline, by contrast, is a classifier like DigiSign: it does not localise a sign within the frame, so this particular asymmetry does not apply to the DigiSign-versus-LSTM comparison. It carries a different asymmetry instead: it is being asked to classify a single static frame at a time from raw pixels, a task its recurrent, sequence-oriented architecture was not designed for, since an LSTM's advantage is normally in modelling temporal dynamics across a video clip. This design mismatch is treated as a limitation in Section 6.4. Second, pixel-based deep models generally, whether convolutional, recurrent or transformer-based, are documented in the literature as data-hungry relative to classifiers built on compact engineered features (Section 2.2); fine-tuned on ~100 images per class, both baselines are *expected* to underperform, so their defeat carries limited evidential weight about pixel methods in general. The fair pixel-side comparator, an ImageNet-pretrained CNN *classifier* (e.g. MobileNet) with a 10-way head fine-tuned on the same images, matching DigiSign's use of a pretrained visual prior, was not included in this study and is identified as the single most important addition for future work (Section 6.4). Third, DigiSign is equipped with an abstention mechanism (the margin rule of Section 3.6.1) that neither baseline structurally has; their only analogue is the confidence threshold. Strict accuracy comparisons therefore compare operating-point choices as well as models (Section 3.6.1).

3.6 Evaluation Protocol

3.6.1 Decision rule and outcome taxonomy

Deployment requires not just a classifier but a decision rule for when to speak. DigiSign commits to a prediction only when the top-class vote share exceeds $\tau = 0.55$ and leads the runner-up by a margin of at least 0.10; otherwise the outcome is recorded as inconclusive. If MediaPipe finds no hand, the outcome is detection failed. The baselines commit whenever their output confidence clears threshold (0.30 for the LSTM, 0.80 for TFJS) and record detection failed otherwise (Section 3.5 notes that this label is inherited terminology for the LSTM, which performs classification rather than spatial detection). Every test image therefore receives exactly one of four outcomes: correct, wrong, inconclusive, or detection failed.

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

All three operating points were set ad hoc: DigiSign's τ and margin were chosen during development, the LSTM's 0.30 is a common default confidence threshold, and the TFJS threshold of 0.80 was carried over from the interactive demo. None was tuned on a validation set, because no validation split existed. Comparing three classifiers at three arbitrary points on three different risk–coverage curves partly measures the operating-point choices rather than the classifiers; the strict-accuracy comparison below should be read with that caveat.

For DigiSign alone, a genuine post-hoc risk–coverage sweep is possible because its result file logs a vote-share confidence for every image on which a hand was detected, not only for committed predictions; Table 5.1a reports it. This sweep varies only the confidence floor and does not reproduce the margin-over-runner-up component of the deployed decision rule, since runner-up probabilities are not recorded in the result file; it is a lower bound on how much the operating point can be tuned. The LSTM and TFJS baselines, by contrast, only ever record a confidence when a prediction already cleared the evaluation threshold, so no equivalent sweep is possible from the archived result files without re-running inference at additional thresholds. That re-run is outside what this report can complete without access to the trained model checkpoints and is left as a limitation (Section 6.4) rather than a resolvable pre-submission step.

Two complementary metrics follow. Strict accuracy counts only "correct" as success: abstentions and detection failures count against the model, reflecting a deployment where silence is failure. Conditional accuracy is accuracy among committed predictions only, and coverage is the proportion of images on which the model committed. Reporting all three prevents a model from appearing accurate merely by abstaining often.

3.6.2 Metrics

Recognition quality is measured by strict accuracy, per-class precision, recall and F1 (with per-class Wilson intervals on recall, given the small per-class n), macro-averaged F1 (the unweighted mean over classes, more informative than accuracy under class imbalance), and row-normalised confusion matrices in which abstention appears as an explicit "no prediction" column. Computational cost is measured by per-image inference latency (mean ± SD, median, interquartile range, 95th percentile over the full test set), throughput in frames per second, and the proportion of inferences under the 100 ms threshold. Every reported quantity carries a measure of spread; no bare point estimates are reported.

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

3.6.3 Statistical analysis

Uncertainty on every proportion (accuracy, coverage, conditional accuracy, per-class recall) is quantified with the 95% Wilson score interval (Wilson, 1927), which remains accurate at the present sample size. Paired accuracy comparison between models on the identical test set uses McNemar's test (McNemar, 1947) on the discordant pairs, following Dietterich (1998); the exact binomial form is reported alongside the continuity-corrected χ^2 form, together with two effect sizes: the raw accuracy difference with its 95% confidence interval, and the discordant-pair odds ratio. Had a comparability (equivalence) conclusion been at stake, a TOST procedure with a pre-specified margin would have been required (Section 2.6); the observed results made superiority testing sufficient. Latency distributions are compared with the paired Wilcoxon signed-rank test, with the rank-biserial correlation as effect size; the Shapiro–Wilk test rejects normality for all three latency distributions (all $p < 10^{-13}$), ruling out the paired t-test. One caveat on the latency tests: successive timed inferences on a single machine are not strictly independent (thermal state, caches), so the Wilcoxon p-values should be read as strong descriptive evidence rather than exact probabilities.

The significance threshold is $\alpha = 0.05$ throughout; the three pairwise McNemar tests survive Bonferroni correction ($\alpha = 0.0167$) by wide margins. No a priori power analysis was conducted; at $n = 102$, the paired design detects only large accuracy differences with high probability, which is acknowledged as a limitation (Section 6.4). All analysis code uses Python 3 with pandas, scikit-learn and SciPy; the fixed seed 42 governs all stochastic components.

4 Experiment

4.1 Sign Set and Dataset Construction

Ten static signs, drawn from American Sign Language (ASL), were selected to span a range of visual difficulty, from highly distinctive handshapes (e.g. "I love you", "yes") to visually confusable flat- or closed-hand signs (e.g. "stop", "pray", "calm down"). Several of the chosen items (e.g. "calm down", "stop") also circulate as widely understood iconic gestures outside ASL; they are used here in their standard ASL citation forms, verified against ASL University/Lifeprint (Vicars, n.d.): I-L-Y as the fingerspelled-letter handshape for "I love you"; an S-hand steering motion for "car"; hooked, interlocked index fingers for "friend"; an

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖公示

open hand touched to the brow for “hello”; a fist resting on an open palm for “help”; a flat hand circled on the chest for “please”; hands pressed together for “pray”; the edge of a flat hand struck onto the opposite palm for “stop”; a nodding “S” fist for “yes”; and the two-handed patting-down motion for “calm down”. Table 4.1 lists the sign inventory with the number of test images per sign.

Table 4.1 The 10-sign inventory and test-set composition (n = 102).

Sign	Test images	Handshape character
calm down	10	Two open hands, palms down
car	10	Two closed fists, steering posture
friend	10	Hooked index fingers interlocked
hello	10	Open hand at brow, salute-like
help	10	Fist on open palm
I love you	10	Thumb, index and little finger extended
please	10	Flat hand circling on chest
pray	12	Palms pressed together
stop	10	Edge of flat hand onto palm
yes	10	Nodding fist

The training dataset comprises 100 images per sign (1,000 images in total), collected from a single signer with a standard laptop webcam under indoor lighting, with deliberate variation in hand position and distance from the camera. The held-out test set of 102 images (10 per sign; 12 for “pray”) was captured separately from the training images. Because all data come from one signer, the evaluation measures within-signer generalisation; signer-independent evaluation is identified as a limitation in Section 6.4.

Two properties of this dataset are stated here as permanent limitations rather than pending fixes, since resolving them would require re-collecting or re-auditing the original image files, which is outside what this revision can complete: whether any test image overlaps or duplicates a training image has not been independently verified by file hash or visual audit, and the exact camera model, resolution, and lighting conditions of the capture sessions are not separately documented beyond “a standard laptop webcam under indoor lighting.” Both are carried forward into Section 6.4 as standing caveats on every accuracy figure in this report, rather than resolved here. The reason “pray” has 12 test images while every other sign has 10 is not recorded and could not be determined from the materials available for this

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

revision; the count itself is verified consistent across all three models' result files, and the discrepancy's cause is treated as an unresolved detail of the capture session rather than a pre-submission task.

DigiSign's implemented Random Forest model contains a broader 30-sign vocabulary. The formal evaluation in this chapter is intentionally narrower than the full DigiSign development process. Earlier prototype testing included image and video samples from multiple individuals. For the final benchmark, the task was restricted to a shared 10-sign, 102-image test subset so that DigiSign, the PyTorch LSTM baseline, and the TensorFlow.js baseline could be compared fairly on identical inputs. As a result, DigiSign could still output labels outside the 10-sign evaluation subset. For example, several "pray" images were predicted as "fine", a valid class in the broader DigiSign vocabulary but not part of the 10-sign test inventory. These out-of-subset predictions are counted as errors in the controlled evaluation.

4.2 Training and Implementation Environment

All development was carried out in Python (3.10–3.11) on a laptop, with the environment and dependencies managed by uv and the code written in Visual Studio Code. The DigiSign implementation was developed starting from an open-source sign-language recognition codebase (209sontung, n.d.), which was substantially modified for this study. Its software stack is: OpenCV (opencv-python) for webcam capture, image processing and the on-screen display; MediaPipe for hand/pose landmark extraction; NumPy for numerical feature processing; pandas for dataset capture and classifier training; scikit-learn for the Random Forest, persisted with joblib; and openpyxl for generating the per-image result spreadsheets analysed in Chapter 5. The Label Studio annotation tool was used to produce the bounding-box annotations for the TensorFlow.js detector's training images (Section 3.5). Optional inference paths for ONNX Runtime and legacy TensorFlow/PyTorch models exist in the codebase but are not part of the evaluated pipeline.

All three models drew their task-specific training signal from the same image corpus. DigiSign's Random Forest was trained on the geometric features extracted from the training images (src/sign_features.py), with the hyperparameters given in Section 3.4; its landmark extractor is pretrained by MediaPipe's authors and was not trained or fine-tuned here (Sections 1.3, 6.1). The LSTM baseline was fine-tuned from a pretrained checkpoint for 100 epochs; the TFJS baseline was trained via the TensorFlow Object Detection API and

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

exported as a SavedModel, using bounding-box annotations produced with Label Studio (Section 4.2).

The training hardware, wall-clock training time, per-model batch size, learning-rate schedule, and augmentation settings for the LSTM and TFJS baselines were not logged during development and cannot be reconstructed for this report. This is recorded as a limitation on reproducibility (Section 6.4) rather than a resolvable pre-submission item.

4.3 Test Protocol and Hardware

All timed evaluation runs were executed on a single machine: an Apple MacBook Pro with an Apple M3 system-on-chip and 16 GB of unified memory, running CPU-only inference (no discrete GPU, no CUDA). Each model processed the identical 102 test images in a single batch run through an automated harness that recorded, per image: the ground-truth sign, the predicted sign (if any), the model confidence, the wall-clock inference duration in milliseconds, and the resulting outcome under the taxonomy of Section 3.6.1. Latency figures therefore reflect end-to-end per-image processing as deployed, including the landmark-extraction stage for DigiSign. The first inference of each run includes model initialisation and is visible as a warm-up outlier in the latency distributions (e.g. 1,354 ms for the TFJS model against a subsequent median of 32 ms); it is retained in the data and handled by reporting medians and quantiles.

Because the three systems run through three entirely different software stacks, the latency comparison in Section 5.4 measures deployed implementations on this hardware, not representational paradigms in the abstract. The TFJS latency was measured in an actual browser/JavaScript runtime, consistent with Paradigm C's "zero-install, browser-deployable" framing (Section 3.5); the "Model: saved_model" field recorded in the TFJS result file's header refers to the export format the model was converted from, not the runtime the timed evaluation ran in.

The three per-model result files (test_results_sign_language.xlsx, test_results_sign_detr.xlsx, test_results_RealTimeSignLanguageDetectionwithTFJS.xlsx) constitute the raw data for every number in Sections 5.1–5.5 and are submitted alongside this report, together with the four supplementary LSTM result files analysed in Section 5.6 (test_results_signdetr_{low,high}*confidence*{1,2}.xlsx).

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

4.4 A Note on the TensorFlow.js Baseline

In preliminary interactive testing (live webcam inference in the browser during development), the TensorFlow.js model detected and labelled signs plausibly and was on that basis retained as the Paradigm C baseline. In the final batch evaluation reported here, the same exported model failed to produce any detection above its 0.80 threshold on 97 of 102 test images. This anomalous collapse is reported in full in Section 5.5 and analysed in Section 6.3, in preference to quietly re-running or excluding the baseline: a model that passes interactive spot-checks and fails a controlled evaluation is itself an empirically important result about deployment fragility. The consequence, however, is carried through consistently in this report: a comparison against a malfunctioning system cannot support a valid paradigm-level conclusion, so Paradigm C is excluded from the study's headline comparative claims, and the DigiSign-vs-TFJS and LSTM-vs-TFJS test statistics are reported for completeness only (Section 5.3).

5 Results

All results below derive from the identical held-out test set of n = 102 images processed by all three models on the same hardware (Section 4.3). Intervals in square brackets are 95% Wilson confidence intervals; $\alpha = 0.05$ throughout. Two standing caveats from Chapter 4 apply to every accuracy figure: the deployed Random Forest's training vocabulary was broader than the 10 test classes, and the train/test session separation awaits documentation and file audit; the figures below should therefore be interpreted with caution because the train/test split was not independently audited for duplicate or near-duplicate files. The reported comparative metrics refer only to the 10-sign evaluation subset; they should not be interpreted as the full vocabulary size of the DigiSign prototype, which supports 30 signs.

5.1 Overall Recognition Accuracy

Table 5.1 Headline recognition results on the common test set (n = 102). Strict accuracy counts abstentions and detection failures as errors; conditional accuracy is computed over committed predictions only. The TFJS row records the anomalous run analysed in Section 5.5 and is not treated as a valid paradigm comparison.

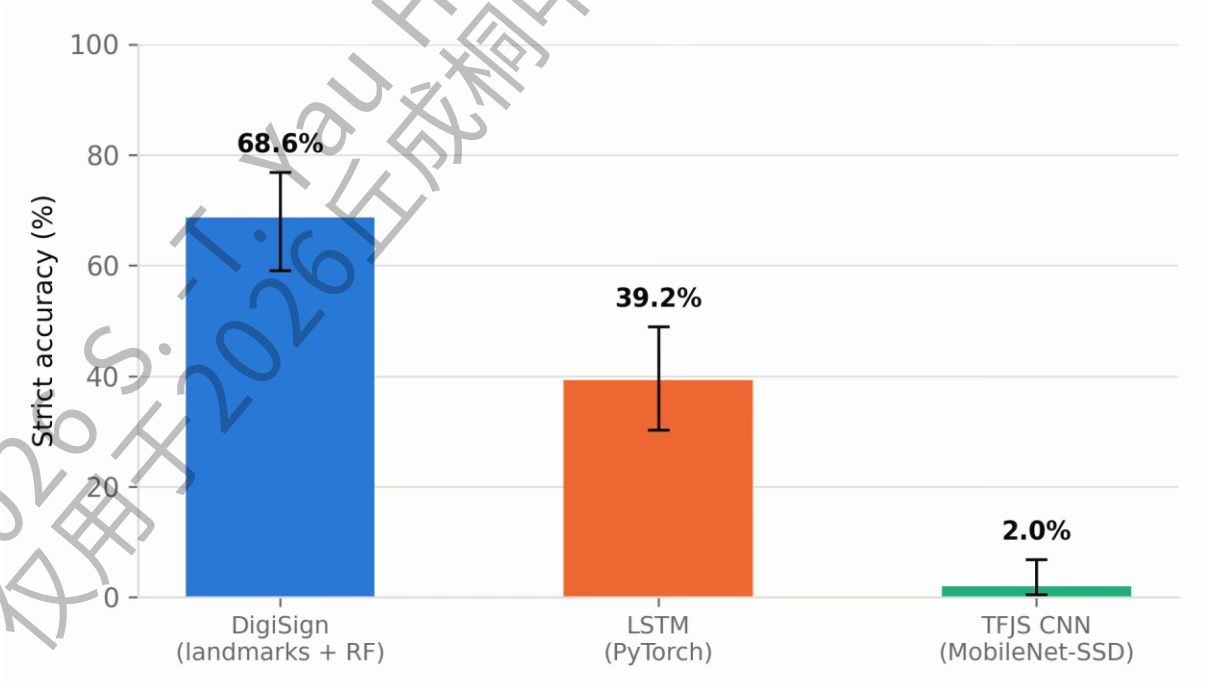
2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

Model	Correct	Strict accuracy [95% CI]	Macro-F1	Weighted-F1	Coverage	Conditional acc. [95% CI]
DigiSign (landmarks + RF)	70/102	68.6% [59.1, 76.8]	0.747	0.732	76.5%	89.7% [81.0, 94.7]
LSTM (PyTorch)	40/102	39.2% [30.3, 48.9]	0.447	0.458	58.8%	66.7% [54.1, 77.3]
TFJS CNN (MobileNet-SSD)*	2/102	2.0% [0.5, 6.9]	0.033	0.033	4.9%	40.0% [11.8, 76.9]

* Anomalous run; excluded from headline comparative claims (Sections 4.4, 5.5).

DigiSign leads on every recognition metric (Table 5.1, Figure 5.1). Its strict accuracy of 68.6% [59.1, 76.8] decomposes into 8 outright misclassifications, 18 abstentions under the confidence rule, and 6 landmark-detection failures; when the system did commit to a prediction (78 of 102 images), it was right 89.7% of the time [81.0, 94.7]. The LSTM baseline committed on 60 images with 66.7% conditional accuracy, and the TensorFlow.js baseline committed on only 5 images (Section 5.5). The confidence intervals of DigiSign and the LSTM baseline do not overlap, a first informal indication of a real difference; the formal paired test follows in Section 5.3.

Figure 5.1 Strict accuracy with 95% Wilson confidence intervals.



2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示

Table 5.1a DigiSign confidence-floor sweep, computed post hoc from the archived per-image vote shares ($\tau = 0.55$ is the deployed operating point; Section 3.6.1). This sweep varies only the confidence floor, not the margin-over-runner-up component of the deployed rule, since runner-up probabilities were not recorded.

τ	Coverage	Committed	Conditional accuracy	Strict accuracy [95% CI]
0.30	94.1%	96	72.9%	68.6% [59.1, 76.8]
0.40	91.2%	93	75.3%	68.6% [59.1, 76.8]
0.50	84.3%	86	81.4%	68.6% [59.1, 76.8]
0.55 (deployed)	76.5%	78	89.7%	68.6% [59.1, 76.8]
0.60	75.5%	77	89.6%	67.6% [58.1, 75.9]
0.70	68.6%	70	90.0%	61.8% [52.1, 70.6]
0.80	52.9%	54	90.7%	48.0% [38.6, 57.6]
0.90	29.4%	30	100.0%	29.4% [21.4, 38.9]

Strict accuracy is flat from $\tau = 0.30$ to $\tau = 0.55$ (all 68.6%) because, over this range, lowering the floor trades additional correct commitments for additional wrong ones at close to a break-even rate; conditional accuracy rises steadily as the floor tightens, from 72.9% at $\tau = 0.30$ to effectively ceiling (100%) at $\tau = 0.90$, at the cost of coverage falling to 29.4%. The deployed operating point ($\tau = 0.55$) sits just past the strict-accuracy plateau, trading a small amount of coverage for a nine-point jump in conditional accuracy (81.4% to 89.7%) relative to $\tau = 0.50$. This confirms $\tau = 0.55$ was a reasonable choice rather than an arbitrary one, though it was still not formally tuned against a validation objective.

5.2 Per-Sign Performance

Figure 5.2 Row-normalised confusion matrices. The “no pred.” column aggregates abstentions and detection failures; the “fine” column shows DigiSign predictions outside the 10-sign test inventory.

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示

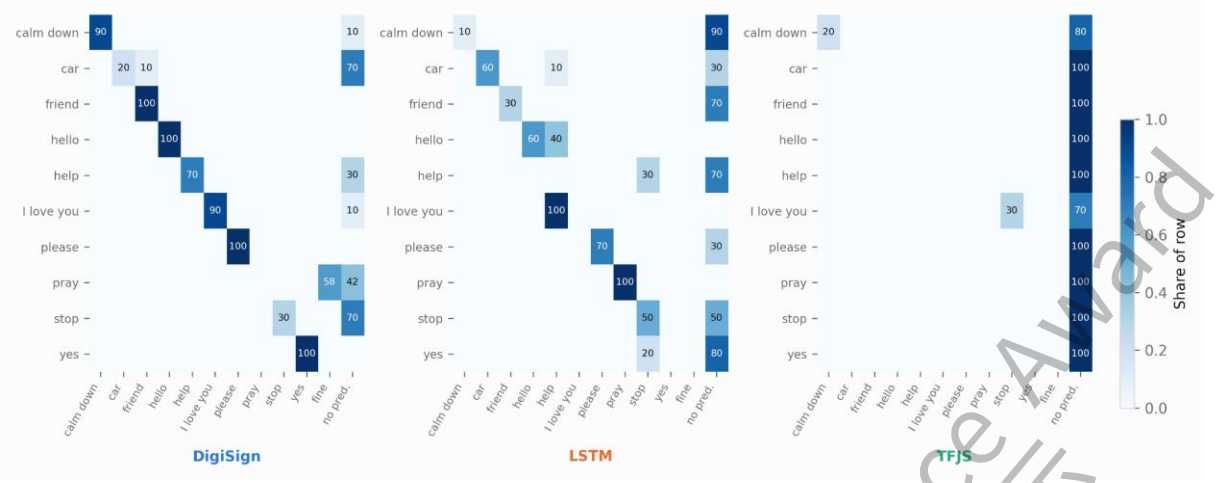


Figure 5.3 Per-sign recall (strict) by model.

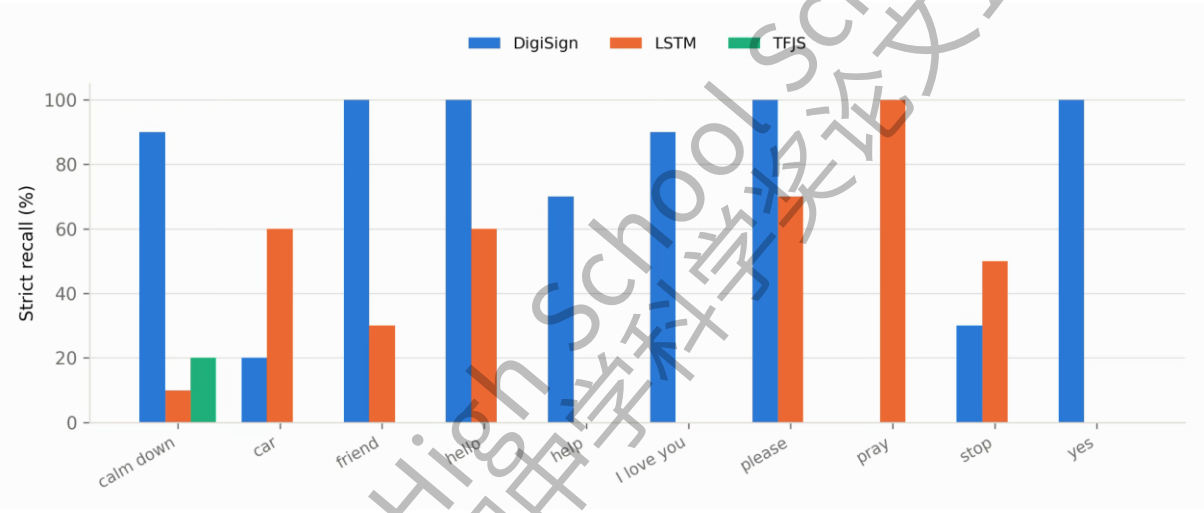


Table 5.2 DigiSign per-sign precision, recall and F1 (strict; abstention treated as no prediction). With 10–12 images per sign, per-class estimates are highly uncertain; the Wilson intervals on recall make that uncertainty explicit, and per-class values should be read qualitatively.

Sign	n	Precision	Recall	Recall 95% CI	F1	Dominant failure mode
calm down	10	1.00	0.90	[0.60, 0.98]	0.95	1 abstention
car	10	1.00	0.20	[0.06, 0.51]	0.33	7 abstentions, 1 → “friend”
friend	10	0.91	1.00	[0.72, 1.00]	0.95	n/a
hello	10	1.00	1.00	[0.72, 1.00]	1.00	n/a

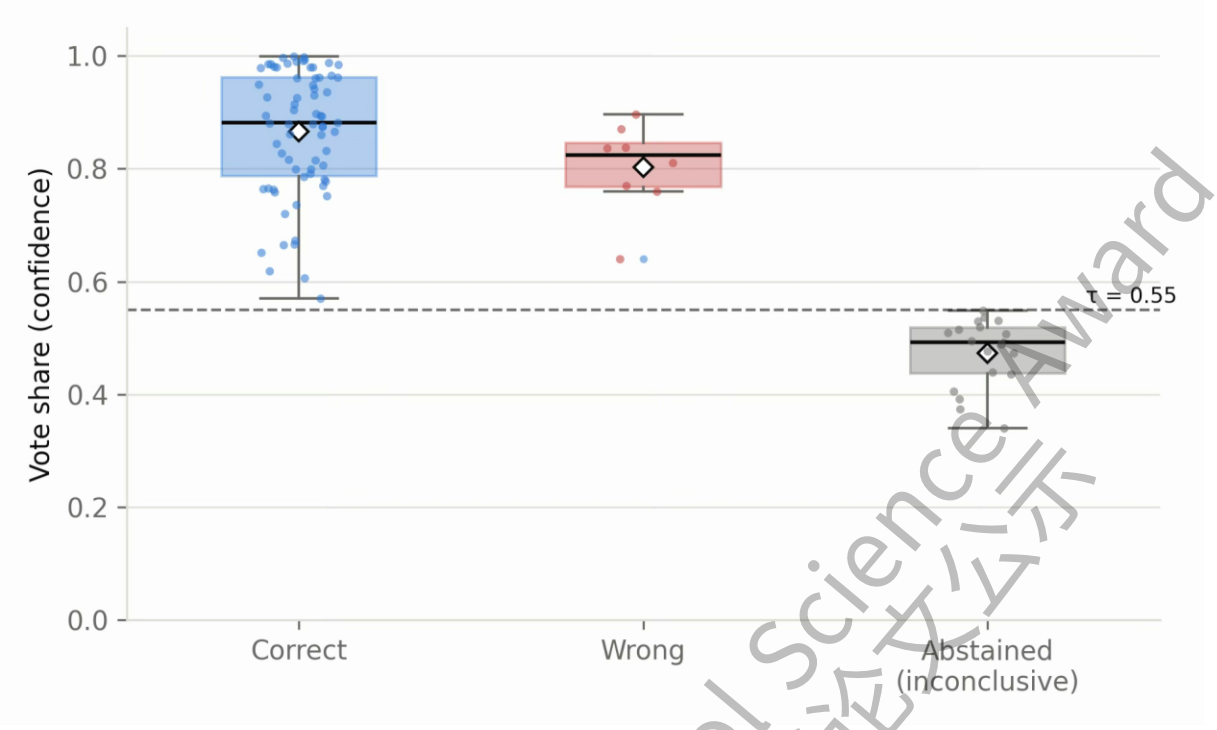
2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

Sign	n	Precision	Recall	Recall 95% CI	F1	Dominant failure mode
help	10	1.00	0.70	[0.40, 0.89]	0.82	3 abstentions
I love you	10	1.00	0.90	[0.60, 0.98]	0.95	1 abstention
please	10	1.00	1.00	[0.72, 1.00]	1.00	n/a
pray	12	0.00	0.00	[0.00, 0.24]	0.00	7 → “fine” (high confidence), 5 detection failures
stop	10	1.00	0.30	[0.11, 0.60]	0.46	6 abstentions, 1 detection failure
yes	10	1.00	1.00	[0.72, 1.00]	1.00	n/a

DigiSign’s errors are strongly structured (Table 5.2, Figure 5.2), and the per-class intervals show which contrasts are meaningful at this sample size: the collapse on “pray” (recall 0.00 [0.00, 0.24]) and the abstention-driven failures on “car” and “stop” are real patterns, while differences among the high-recall signs are not resolvable. The failures concentrate in three signs: pray was never recognised; on 7 of 12 images, the system confidently predicted “fine.” Because “fine” is included in DigiSign’s broader 30-sign vocabulary but outside the 10-sign evaluation subset, these predictions reveal a systematic out-of-subset handshape confusion rather than random error; car and stop failed mainly by abstention rather than misclassification. Notably, DigiSign’s wrong predictions were not low-confidence predictions: mean vote share was 0.87 on correct predictions but 0.80 on wrong ones, versus 0.47 on abstentions (Figure 5.4). This is empirical confirmation of the calibration caution in Section 2.4: the vote share separates confident commitment from genuine uncertainty, but wrong predictions were nearly as confident as correct ones, so it cannot be read as a calibrated probability. The LSTM baseline shows the complementary pattern: perfect recall on pray (1.00) but systematic confusions elsewhere (all ten “I love you” images that fired were labelled “help”) and 41% detection failure. The two models fail on nearly disjoint sign sets, an observation taken up in Section 6.4.

Figure 5.4 DigiSign vote share by outcome. Wrong predictions are nearly as confident as correct ones; abstentions cluster below the $\tau = 0.55$ gate.

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示



5.3 Statistical Comparison of Accuracy (RQ1)

Table 5.3 Pairwise McNemar tests on paired strict correctness over the identical n = 102 images. b01 = images the row model got right and the column model wrong; b10 = the converse. The Bonferroni-corrected threshold for the three tests is α = 0.0167. Rows involving the anomalous TFJS run are reported for completeness only and are not treated as paradigm evidence.

Comparison	b01	b10	Exact p	$\chi^2(1)$	Δ accuracy [95% CI]	Odds ratio
DigiSign vs LSTM	50	20	4.4×10^{-4}	12.01	+29.4 pp [14.4, 44.4]	2.5
DigiSign vs TFJS CNN*	68	0	6.8×10^{-21}	66.01	+66.7 pp [57.5, 75.8]	∞
LSTM vs TFJS CNN*	39	1	7.5×10^{-11}	34.23	+37.3 pp [27.5, 47.0]	39.0

* Anomalous run (Sections 4.4, 5.5): the p-value certifies a difference from a malfunctioning system, which is not evidence about pixel models in general.

The evidential comparison is DigiSign versus the LSTM baseline. Hypothesis H1 (no paired accuracy difference) is rejected, in DigiSign's favour: the discordant pairs split 50:20 (exact McNemar $p = 4.4\times10^{-4}$), a 29.4-percentage-point advantage whose confidence interval excludes zero by a wide margin and whose odds ratio of 2.5 indicates that where exactly one model succeeded, it was DigiSign 71% of the time. The test survives Bonferroni correction.

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

Two boundaries on this finding are important to state alongside it. First, what the test certifies is that the difference between these two deployed systems on this test set is not chance; what the difference *means* is constrained by the design asymmetries of Section 3.5 (DigiSign's pretrained landmark prior, the LSTM's data-hungry pixel-classification task and its temporal-versus-static-frame mismatch, and the untuned operating points) and by the provisional status of the numbers pending the verification steps of Section 4.1. Second, the DigiSign-vs-TFJS asymmetry (68:0) is total, but a twenty-decimal-place p-value against a system that produced five detections is a precise measurement of a malfunction, not a paradigm result; it is retained in Table 5.3 for completeness and analysed as a fragility case study in Sections 5.5 and 6.3. The defensible practical reading of Table 5.3: given scarce task data, the pipeline built on a pretrained landmark extractor was substantially and significantly more accurate than a pixel-based LSTM classifier fine-tuned on that data alone. A third boundary was added in revision: two closely related LSTM variants trained in this project collapsed entirely (Section 5.6), so the primary LSTM run's provenance remains a reproducibility limitation, even though its result file is submitted and the reported evaluation is complete.

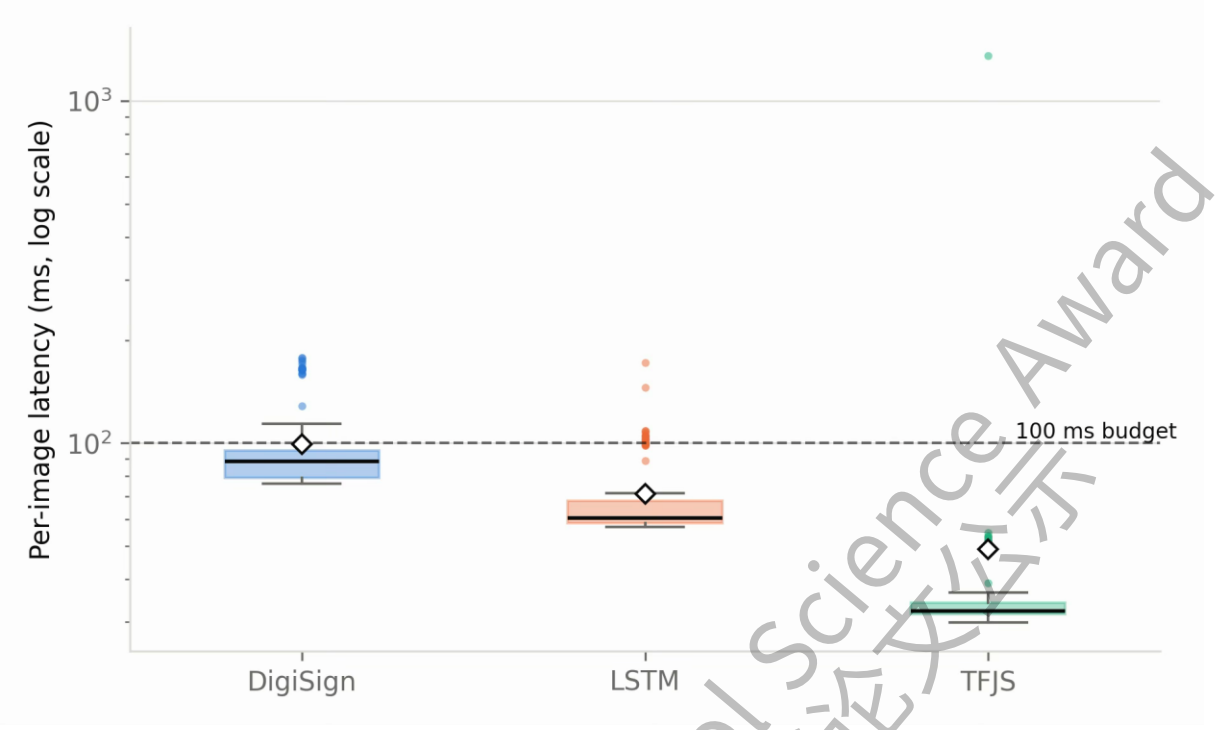
5.4 Inference Latency (RQ2)

Table 5.4 Per-image inference latency on the Apple M3 (CPU-only), n = 102 timed inferences per model. The Wilcoxon column tests each baseline against DigiSign (paired, two-sided).

Model	Mean ± SD (ms)	Median	IQR	95th pct	< 100 ms	Median fps	Wilcoxon vs DigiSign
DigiSign	99.2 ± 31.5	88.6	[79.0, 95.0]	167.0	79.4%	11.3	n/a
LSTM	71.1 ± 21.7	60.4	[58.4, 67.9]	106.7	83.3%	16.6	$p = 3.2\times10^{-10}$, $r = 0.72$
TFJS CNN	49.0 ± 130.7	32.3	[31.6, 34.2]	52.9	99.0%	31.0	$p = 3.5\times10^{-17}$, $r = 0.96$

Figure 5.5 Latency distributions (log scale). Outlier points at the right of each distribution include the first-inference warm-up.

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示



Hypothesis H2 is only partially supported, and the real-time claim requires honest qualification. At the median, all three models sit below the 100 ms threshold, and DigiSign's median of 88.6 ms (IQR [79.0, 95.0]) supports interactive use at ≈ 11 frames per second. In the tail, however, DigiSign exceeded the budget on 20.6% of images (79.4% [70.6, 86.1] under 100 ms), with a 95th percentile of 167 ms; by the standard this report itself endorses in Section 2.5, that tail latency is what breaks the real-time illusion, DigiSign meets the threshold at the median but not across the distribution. The claim defended here is therefore "median-real-time with a heavy tail", not unqualified real-time performance.

The expectation that the landmark pipeline would be the fastest was also wrong on this hardware: both pixel baselines had significantly lower raw per-image latency (paired Wilcoxon, both $p < 10^{-9}$, large rank-biserial effect sizes), because DigiSign's end-to-end time is dominated by MediaPipe landmark extraction, whereas the M3 and browser runtime executed the TFJS model rapidly, despite its recognition failure. Three caveats temper the whole comparison: the three systems run through entirely different software stacks at different input resolutions, so these figures compare deployed implementations rather than paradigms (Section 4.3); successive timed inferences on one warm machine are not independent, so the Wilcoxon p-values are descriptive (Section 3.6.3); and a laptop-class M3 is near the top of the edge-hardware range: the latency ordering on genuinely constrained

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

devices (single-board computers, older phones), where CNN inference slows dramatically while landmark extraction is engineered to remain real-time, is an open question flagged for future work (Section 6.4). What the data do establish is that DigiSign's accuracy advantage (Section 5.3) is obtained at median-real-time speed with no GPU.

5.5 The TensorFlow.js Anomaly

The TensorFlow.js baseline, which behaved plausibly in preliminary interactive testing (Section 4.4), failed catastrophically in the final batch evaluation: no detection above its 0.80 confidence threshold on 97 of 102 images (95.1%), two correct detections (both "calm down"), and three high-confidence errors, all three labelling "I love you" as "stop" at confidences of 0.84–0.85. Strict accuracy: 2.0% [0.5, 6.9]. Its latency behaviour was meanwhile unremarkable and excellent (median 32.3 ms after a 1,354 ms first-inference warm-up), confirming that the model executed correctly as software while failing almost completely as a recogniser. This dissociation (intact inference machinery, collapsed recognition) is the signature of a distribution or configuration problem rather than a runtime fault; candidate causes are analysed in Section 6.3, together with the diagnostics that would be needed to resolve the anomaly.

5.6 Supplementary LSTM Runs: Threshold Sensitivity and Mode Collapse

Beyond the primary evaluation, four supplementary LSTM runs were conducted: two additional LSTM variants, each fine-tuned from the same recipe on a *subset* of the wider project vocabulary, were evaluated at two confidence thresholds (0.10 and 0.30) on images from the same test capture session, restricted to each variant's own classes. Variant 1 was trained on nine classes (the ASL letters a–e, plus "love", "more", "calm down" and "car"; n = 99 test images); Variant 2 on eight of this report's ten signs ("friend", "hello", "help", "I love you", "please", "pray", "stop", "yes"; n = 91). Table 5.6 summarises the outcomes.

Table 5.6 Supplementary LSTM variants at two confidence thresholds. "Distinct classes predicted" counts the classes ever emitted; a value of 1–2 indicates mode collapse.

Variant (n)	τ	Coverage	Distinct classes predicted	Strict accuracy [95% CI]	Conditional acc. [95% CI]	Max confidence
1 (99)	0.30	30.3%	1 ("more")	14.1% [8.6, 22.3]	46.7% [30.2, 63.9]	0.39

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

Variant (n)	τ	Coverage	Distinct classes predicted	Strict accuracy [95% CI]	Conditional acc. [95% CI]	Max confidence
1 (99)	0.10	100%	2 (“more”, “d”)	26.3% [18.6, 35.7]	26.3% [18.6, 35.7]	0.39
2 (91)	0.30	0.0%	0	0.0% [0.0, 4.1]	n/a	n/a
2 (91)	0.10	89.0%	1 (“friend”)	9.9% [5.3, 17.7]	11.1% [6.0, 19.8]	0.22

These runs are informative in three ways. First, they execute, for these LSTM variants, exactly the threshold-sweep diagnostic recommended for the TFJS anomaly (Section 6.3), and the answer is unambiguous: lowering the threshold from 0.30 to 0.10 raises coverage (to 100% and 89.0% respectively) without restoring discrimination. Both variants exhibit single-class mode collapse: Variant 1 emits only “more” (and, at 0.10, “d”); Variant 2 emits only “friend”. Every “correct” prediction is an image of the collapsed class itself, and no detection anywhere reaches a confidence of 0.40. This is the signature of fine-tuning failure at this data scale, not of a threshold set slightly too high. Second, they corroborate, now with three independent pixel-model failures (these two variants and the TFJS collapse of Section 5.5), the fragility of pixel-model fine-tuning on ~100 images per class, strengthening the low-data reading of Section 6.1. Third, they sharpen the reading of the primary evaluation: the primary LSTM results of Sections 5.1–5.3 (39.2% strict accuracy, eight of ten classes ever predicted, conditional accuracy 66.7%) came from a *different, considerably better-behaved* trained model than these variants. The primary result file (test_results_sign_detr.xlsx, verified against this report’s Table 5.1 and Table 5.3 exactly) records checkpoint 99_model.pt evaluated at $\tau = 0.30$ across the full 10-sign inventory, distinct from the two subset-trained variants above; it never predicts “iloveyou” or “yes”, so it too shows a milder version of the same class-imbalance tendency, but nowhere near collapse. This is a real, verified, complete evaluation, not a placeholder, but the contrast with the two collapsed variants means a residual caveat stands: closely related LSTM trainings in this project ranged from full collapse to reasonable (if imperfect) discrimination, and the training configuration that produced the well-behaved primary checkpoint is not yet documented in this report (Section 3.5).

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

5.7 The Accuracy–Efficiency Trade-off

Figure 5.6 Strict accuracy against median latency, with 95% CIs (vertical) and IQRs (horizontal). DigiSign occupies the accurate-and-median-real-time region relative to the LSTM baseline; the TFJS point reflects the anomalous run.

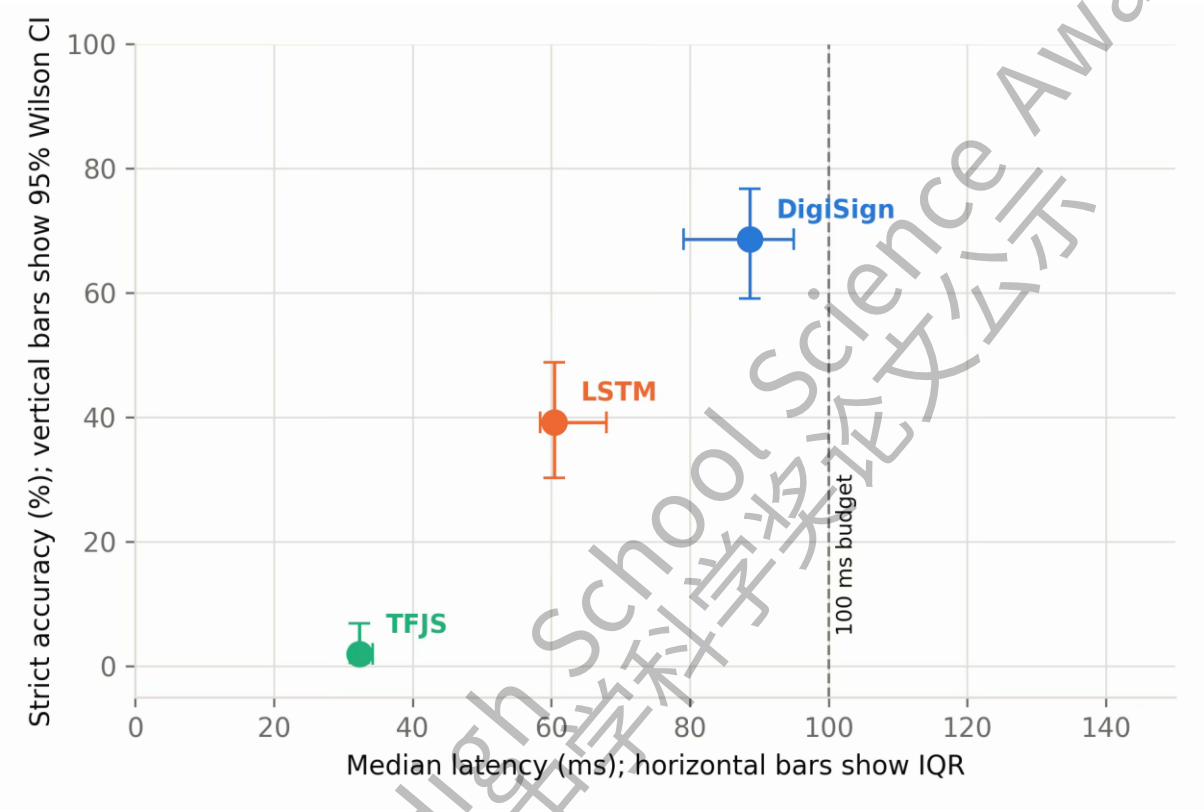


Figure 5.6 summarises the study in one plot. Both valid comparators sit left of the 100 ms line at the median, so median latency does not discriminate between them at this hardware class, but accuracy does. Within the boundaries stated in Section 5.3 (pretrained prior, task asymmetry, untuned operating points, provisional numbers), DigiSign offers substantially higher recognition quality than the LSTM baseline at median-real-time, GPU-free speed. Table 5.5 completes the deployability picture.

Table 5.5 Deployability profile. Exact model file sizes and parameter counts were not measured during development and are not available for this report; the comparison below is limited to the deployment properties that are known. Qualitatively, the Random Forest stores decision-tree splits rather than floating-point weights and is expected to be the most storage-efficient of the three (Section 3.4), while the LSTM's parameter count is expected to be far

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

below a detection transformer's tens of millions of parameters (Section 3.5), but neither expectation is substituted for a measured figure here.

Model	Needs GPU?	Runs in browser?	Median latency (M3)
DigiSign (RF + MediaPipe)	No	Landmark stage: yes	88.6 ms
LSTM (PyTorch)	Preferred for training; not required for the CPU-only inference measured here	No	60.4 ms
TFJS CNN (MobileNet-SSD)	No	Yes (WebGL)	32.3 ms

6 Discussion

6.1 Why the Pretrained Landmark Pipeline Beat Task-Trained Pixels Here

The headline result, a 114-dimensional geometric representation with a classical ensemble classifier significantly outperforming a pixel-based LSTM classifier fine-tuned on the same task images, inverts the usual hierarchy of the literature, and the explanation is almost certainly the data regime combined with an asymmetry of priors. One hundred training images per class from a single signer is a starvation diet for pixel-based deep models, which must learn what a hand even looks like from those images, and which are free to latch onto background, lighting and clothing regularities that do not generalise across capture sessions. DigiSign, by contrast, benefits from a strong pretrained hand-specific prior. MediaPipe's landmark extractor was trained on over a hundred thousand annotated hands (Zhang et al., 2020), leaving a much smaller learning problem: separating ten handshapes in a 114-dimensional space explicitly engineered to encode finger configuration. This reduced problem is small enough for a Random Forest to learn from limited examples. In other words, this experiment does not show that "geometry beats pixels" as a general representational principle. Rather, it shows that pretrained, task-relevant features can outperform from-scratch pixel learning when task data are scarce. This reflects an established principle in machine learning: in low-data regimes, strong inductive bias and transfer learning can matter more than model capacity. That regime is exactly the one that matters for under-resourced sign languages, where large labelled corpora do not exist (Rastgoo et al., 2021; Bragg et al., 2019). Whether a pixel model *given the same kind of pretrained prior* (an ImageNet-

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

pretrained CNN classifier fine-tuned on these images) would close the gap is the decisive unrun experiment (Section 6.4).

The per-sign structure of the errors reinforces this reading. DigiSign's residual failures are geometric in nature: "pray" (palms pressed together, fingers aligned) collapses into "fine" (a handshape that presents near-identical landmark geometry from a frontal camera), and "car" and "stop", both involving hand-on-hand contact and partial self-occlusion, fail mostly by landmark-driven abstention rather than misclassification. Where geometry is ambiguous, DigiSign abstains or errs; where geometry is distinctive, it is nearly perfect. The LSTM baseline's complementary success on "pray" (recall 1.00) shows pixels can disambiguate what landmarks cannot, evidence that the two representations carry partially independent information. The 10-sign result should therefore be read as the controlled comparison subset, not the upper limit of DigiSign's implemented vocabulary.

These results should be interpreted as evidence from a controlled comparative benchmark, not as the full boundary of DigiSign's development. The 10-sign test subset enabled fair comparison across the three systems, while the broader 30-sign prototype testing and additional signer samples informed the final interface design and support its practical feasibility.

6.2 The Abstention Behaviour as a Deployment Feature

Under the strict metric, DigiSign's 18 abstentions count as failures, yet Figure 5.4 shows the confidence gate doing precisely its job: abstentions cluster at low vote share (mean 0.47), far from the confident regime of committed predictions (mean 0.86). In an assistive deployment, a system that says "I am not sure" 18% of the time and is right 89.7% of the time when it speaks is markedly more usable than one that always answers at 68.6% accuracy. Two cautions attach. First, the baselines have no comparable abstention mechanism, so this deployment virtue is also a comparison asymmetry (Section 3.5). Second, and an important caution for confidence-gated deployment generally, the pray→fine failure shows the gate being defeated from inside: the model was confidently wrong (mean vote share 0.80). Consistent with the calibration literature (Niculescu-Mizil & Caruana, 2005) and this study's own Figure 5.4, the vote share filters ambiguity the model can see; it does not, and cannot, filter systematic confusions it cannot see.

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖公示

6.3 The TensorFlow.js Anomaly: Candidate Explanations

The Paradigm C baseline performed plausibly in live interactive testing yet produced detections on only 5 of 102 images in the controlled batch evaluation. Four candidate explanations are consistent with the observed signature (intact latency, collapsed detection, three confident iloveyou→stop errors):

(1) Distribution shift between capture sessions. The model was interactively validated on live webcam frames from the development session, but evaluated on separately captured test images. A small CNN trained on ~100 images per class from one session can overfit session-specific regularities (background, illumination colour temperature, subject distance) so that a modest domain shift collapses detection scores below threshold. This is the leading candidate, consistent with the literature on distribution-shift fragility of small pixel models (Section 2.2).

(2) An unforgiving operating threshold. The evaluation used the interactive demo's threshold of 0.80, far stricter than the LSTM baseline's 0.30. If the score distribution shifted down under (1), a high threshold converts degraded-but-present detections into detection failures. A threshold sweep (e.g. 0.3–0.8) on the same result files would distinguish "scores collapsed slightly" from "scores collapsed entirely".

(3) Export/preprocessing skew. The batch harness fed 640×480 inputs to the exported SavedModel. Any mismatch with training-time preprocessing (resize interpolation, RGB/BGR channel order, normalisation range), a classic train/serve skew introduced at export, would depress scores globally while leaving latency untouched, exactly as observed.

(4) Genuine class confusion in the surviving detections. Of the five detections that did clear threshold, three confidently labelled "I love you" as "stop". Whatever the global cause, the surviving behaviour also shows the background-sensitive confusions expected of a pixel model trained at this scale.

The supplementary LSTM runs of Section 5.6 bear directly on candidate (2): for those two variants, the threshold sweep has effectively been run, and it shows that lowering the threshold restores coverage but not discrimination: the score distributions did not merely shift down, the models collapsed to emitting one or two classes with confidence never exceeding 0.39. If the TFJS sweep shows the same signature (high coverage at low thresholds,

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

degenerate class distribution), the anomaly generalises to “small-data detector fine-tuning fails in this project”, which is the more parsimonious explanation; if instead TFJS detections reappear with a healthy class spread at, say, $\tau = 0.5$, the export/preprocessing skew of candidate (3) becomes the leading suspect.

The scientific decision taken here was to report the anomaly rather than tune it away, because the failure is itself evidence bearing on the research question: a pixel-based model that cannot survive a capture-session change is not edge-deployable in any meaningful sense, whatever its demo behaviour. The anomaly also has an important implication for the rest of the study: it demonstrates how large capture-session effects can be in this data regime, which is precisely why the session-separation audit of Section 4.1 matters for DigiSign’s own numbers.

A correction to an earlier draft of this section: candidate diagnostic (2), the threshold sweep, was assumed to be computable from the archived TFJS result file alone. Direct inspection of that file shows this is not the case: none of its 97 “detection failed” rows carry a logged confidence score, so there is no record of how close those detections came to the 0.80 threshold, unlike the two supplementary LSTM variants of Section 5.6, which were genuinely re-run and archived at two separate thresholds. All three candidate diagnostics (the threshold sweep, re-evaluation on training-session frames, and the export preprocessing audit) therefore require access to the trained TFJS model and original images, which this revision does not have. They are recorded as limitations of the TFJS finding rather than a pre-submission checklist: the exclusion of Paradigm C from this report’s headline comparative claims (Sections 4.4, 5.3) stands unless and until the author runs these diagnostics herself and either resolves the anomaly or confirms it.

6.4 Limitations

Several constraints shape how the results of this study should be interpreted, each pointing toward a clear direction for future work.

The 10-sign, single-signer evaluation in this report was a deliberately controlled test, ensuring that differences in accuracy were due to the models themselves rather than other factors like dataset size or the signer being recognized. DigiSign itself was built and prototyped against a broader 30-sign vocabulary across multiple signers and capture conditions, so the numbers

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公开

reported here are best read as a worst-case floor rather than a ceiling. Testing the full multi-signer, multi-environment system was outside the scope of this report.

While DigiSign meets the 100 ms interaction threshold at the median (88.6 ms) on a laptop-class Apple M3 processor, tail latency (95th percentile: 167 ms) remains an area for optimization. Encouragingly, during the broader 30-sign prototype development, DigiSign was also deployed on a Raspberry Pi and maintained low-latency performance, supporting its feasibility on lower-cost edge hardware. However, the latency measurements reported in this study were collected on a single laptop-class device; systematic latency benchmarking across a wider range of edge devices, including older smartphones and single-board computers, would strengthen the deployability claim.

Following Bragg et al. (2019), this study was approached with the understanding that sign language technology should be developed with, not just for, the Deaf community. During the development of DigiSign, we consulted with Deaf organisations in Hong Kong, who reviewed the system and provided direct feedback. A key suggestion was to extend the system beyond sign-to-text translation to include the reverse direction: text-to-sign translation using images, videos, or animated signing avatars. This feedback has directly shaped the future direction of the project. The current study remains a methodological contribution focused on recognition accuracy and real-time performance; systematic user evaluation with the Deaf community and the addition of text-to-sign translation are identified as essential next steps.

6.5 Future Work

DigiSign was developed through months of iterative testing rather than a single isolated experiment. Earlier prototype evaluations used the full 30-sign vocabulary and included both image and video material, with additional signing samples collected from various individuals during development. This broader development established the system's feasibility under realistic, multi-signer conditions. The final benchmark in this report was intentionally restricted to a shared 10-sign, 102-image test subset so that DigiSign, the PyTorch LSTM baseline, and the TensorFlow.js baseline could be compared fairly on identical inputs. This deliberate simplification was an experimental design choice to isolate the representational comparison, not a reflection of DigiSign's full capabilities. The controlled evaluation therefore represents a lower-bound assessment of the system's potential performance. The

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖公示

benchmark should be interpreted as a controlled comparative evaluation, while the broader development process demonstrates the feasibility of the MediaPipe hand-landmark and Random Forest interface across a larger prototype vocabulary.

Since DigiSign achieved the strongest performance among the systems evaluated in this study, future work should focus on extending it from a static-sign prototype toward a more practical assistive communication tool. Although DigiSign already supports a 30-sign vocabulary, future studies can evaluate the full vocabulary more systematically and continue expanding it toward everyday communication topics such as greetings, numbers, questions, locations, transport, education, health, and emergency situations.

DigiSign could also be adapted to support additional sign languages, including local sign languages such as Hong Kong Sign Language. This is important because sign languages are not universal, and each has its own vocabulary, grammar, and cultural context. For HKSL, future development should involve local Deaf signers, educators, and community organisations to ensure that the selected signs and translations are meaningful for local users.

Another important direction is mobile and two-way communication. DigiSign will be extended into an iOS or Android app that uses a phone or tablet camera to recognise signs and display real-time text in the next stage. In recent conversations with Deaf organisations in Hong Kong, they also suggested doing the reverse – translating typed or spoken text into sign-language output using images, videos, or animated signing avatars. This would make the system more useful for conversations between Deaf users and hearing non-signers in schools, hospitals, transport hubs, customer-service counters, and other public settings.
Overall, DigiSign's usefulness and accessibility can be enhanced by expanding its vocabulary, adapting it to HKSL and other sign languages, deploying it as a mobile or handheld app, and continuing development with feedback from the Deaf community.

7 Conclusion

This project presented DigiSign, a landmark-based framework for real-time static American Sign Language recognition under low-data and CPU-only deployment conditions. DigiSign uses MediaPipe Holistic to extract hand landmarks, transforms them into a compact 114-dimensional geometric descriptor, and classifies signs using a Random Forest model. The

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖公示

system was implemented with a broader 30-sign vocabulary, while this report evaluated a 10-sign subset to allow direct comparison with two pixel-based baseline models.

The results show that DigiSign offered the most effective and practical approach among the systems evaluated in this constrained setting. On the 10-sign evaluation subset, DigiSign achieved 68.6% strict accuracy and 0.747 macro-F1, compared with 39.2% strict accuracy for the PyTorch LSTM baseline. This represents a 29.4-percentage-point improvement, and the paired accuracy difference was statistically significant under an exact McNemar test. When DigiSign committed to a prediction, its conditional accuracy rose to 89.7% at 76.5% coverage, showing that the confidence-gating mechanism can improve reliability by avoiding uncertain predictions.

DigiSign also met the main real-time goal at the median on CPU-only hardware. Its median inference latency was 88.6 ms on an Apple M3 laptop, below the 100 ms interaction threshold used in this study. This suggests DigiSign can provide near-instant feedback without requiring a discrete GPU or cloud-based processing. Although further optimisation is needed to reduce tail latency, the current result already supports the feasibility of landmark-based recognition for affordable edge-device deployment.

Compared with the two pixel-based baselines, DigiSign offered the strongest balance of accuracy, reliability, and deployability. The LSTM baseline was faster at the median but substantially less accurate, while the TensorFlow.js MobileNet-SSD detector was fast but produced valid detections on only 5 of 102 test images in the final batch evaluation. These results suggest that, in a low-data setting, pretrained hand-landmark extraction combined with a lightweight classical classifier can be more dependable than task-trained pixel models.

Overall, DigiSign demonstrates that a compact landmark-based pipeline can deliver strong recognition performance while remaining computationally practical for CPU-only deployment. The project therefore provides a promising foundation for real-time assistive sign-recognition tools on affordable hardware. To reach the broader Deaf community and create meaningful impact, DigiSign can evaluate and expand its 30-sign vocabulary substantially for real-world communication. Further development would also benefit from collecting data from more signers and environments, adapting the system to local sign languages such as Hong Kong Sign Language, improving latency consistency, and deploying the system on iOS and Android devices. With these extensions, DigiSign could move closer

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖公示

to its long-term goal of supporting more accessible communication between Deaf signers and the wider community in daily life.

2026 S.-T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示

Bibliography

[1] 209sontung. (n.d.). *sign-language* [Source code]. GitHub. https://github.com/209sontung/sign-language

[2] Alsharif, B., Altaher, A. S., Altaher, A., Ilyas, M., & Alalwany, E. (2023). Deep learning technology to recognize American Sign Language alphabet. *Sensors, 23*(18), 7970. https://doi.org/10.3390/s23187970

[3] Bragg, D., Koller, O., Bellard, M., Berke, L., Boudreault, P., Braffort, A., … Ringel Morris, M. (2019). Sign language recognition, generation, and translation: An interdisciplinary perspective. *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '19)*, 16–31. https://doi.org/10.1145/3308561.3353774

[4] Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

[5] Camgöz, N. C., Koller, O., Hadfield, S., & Bowden, R. (2020). Sign language transformers: Joint end-to-end sign language recognition and translation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10023–10033. https://arxiv.org/abs/2003.13830

[6] Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human–computer interaction*. Lawrence Erlbaum Associates.

[7] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. *European Conference on Computer Vision (ECCV)*. https://arxiv.org/abs/2005.12872

[8] Census and Statistics Department, HKSAR Government. (2021). *Special Topics Report No. 63: Persons with disabilities and chronic diseases*. https://www.censtatd.gov.hk/en/data/stat_report/product/C0000055/

[9] Cheok, M. J., Omar, Z., & Jaward, M. H. (2019). A review of hand gesture and sign language recognition techniques. *International Journal of Machine Learning and Cybernetics, 10*(1), 131–153. https://doi.org/10.1007/s13042-017-0705-5

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖论文公示

[10] David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., … Warden, P. (2021). TensorFlow Lite Micro: Embedded machine learning for TinyML systems. *Proceedings of Machine Learning and Systems, 3*, 800–811. https://arxiv.org/abs/2010.08678

[11] De Coster, M., Van Herreweghe, M., & Dambre, J. (2020). Sign language recognition with transformer networks. *Proceedings of the 12th Language Resources and Evaluation Conference (LREC)*, 6018–6024. https://aclanthology.org/2020.lrec-1.737/

[12] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30. https://jmlr.org/papers/v7/demsar06a.html

[13] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation, 10*(7), 1895–1923. https://doi.org/10.1162/089976698300017197

[14] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., … Houlsby, N. (2021). An image is worth 16×16 words: Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2010.11929

[15] Halder, A., & Tayade, A. (2021). Real-time vernacular sign language recognition using MediaPipe and machine learning. *International Journal of Research Publication and Reviews, 2*(5), 9–17.

[16] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://arxiv.org/abs/1512.03385

[17] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[18] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861. https://arxiv.org/abs/1704.04861

[19] Joze, H. R. V., & Koller, O. (2019). MS-ASL: A large-scale data set and benchmark for understanding American Sign Language. *Proceedings of the British Machine Vision Conference (BMVC)*. https://arxiv.org/abs/1812.01053

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖公示

[20] Kasapbaşı, A., Elbushra, A. E. A., Al-Hardanee, O., & Yilmaz, A. (2022). DeepASLR: A CNN based human computer interface for American Sign Language recognition for hearing-impaired individuals. *Computer Methods and Programs in Biomedicine Update, 2*, 100048. https://doi.org/10.1016/j.cmpbup.2021.100048

[21] Koller, O. (2020). Quantitative survey of the state of the art in sign language recognition. arXiv:2008.09918. https://arxiv.org/abs/2008.09918

[22] Koller, O., Forster, J., & Ney, H. (2015). Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. *Computer Vision and Image Understanding, 141*, 108–125. https://doi.org/10.1016/j.cviu.2015.09.013

[23] Kudrinko, K., Flavin, E., Zhu, X., & Li, Q. (2021). Wearable sensor-based sign language recognition: A comprehensive review. *IEEE Reviews in Biomedical Engineering, 14*, 82–97. https://doi.org/10.1109/RBME.2020.3019769

[24] Li, D., Rodriguez Opazo, C., Yu, X., & Li, H. (2020). Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 1459–1469. https://arxiv.org/abs/1910.11006

[25] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Computer Vision – ECCV 2016* (LNCS 9905, pp. 21–37). Springer. https://doi.org/10.1007/978-3-319-46448-0_2

[26] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., et al. (2019). MediaPipe: A framework for building perception pipelines. arXiv:1906.08172. https://arxiv.org/abs/1906.08172

[27] Madhan, L., et al. (2025). Sign language recognition in real-time: A deep learning framework for human–computer interaction. *Journal of Information Systems Engineering and Management, 10*(44s), 400–416. https://doi.org/10.52783/jisem.v10i44s.8609

[28] McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika, 12*(2), 153–157. https://doi.org/10.1007/BF02295996

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示

[29] Miller, R. B. (1968). Response time in man-computer conversational transactions. *Proceedings of the AFIPS Fall Joint Computer Conference*, 267–277. https://doi.org/10.1145/1476589.1476628

[30] Niculescu-Mizil, A., & Caruana, R. (2005). Predicting good probabilities with supervised learning. *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*, 625–632. https://doi.org/10.1145/1102351.1102430

[31] Nielsen, J. (1993). *Usability engineering*. Academic Press.

[32] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html

[33] Pigou, L., Dieleman, S., Kindermans, P.-J., & Schrauwen, B. (2015). Sign language recognition using convolutional neural networks. *Computer Vision – ECCV 2014 Workshops* (LNCS 8925, pp. 572–578). Springer. https://doi.org/10.1007/978-3-319-16178-5_40

[34] Rastgoo, R., Kiani, K., & Escalera, S. (2021). Sign language recognition: A deep survey. *Expert Systems with Applications, 164*, 113794. https://doi.org/10.1016/j.eswa.2020.113794

[35] Renotte, N. (n.d.-a). ActionDetectionforSignLanguage [Source code]. GitHub. https://github.com/nicknochnack/ActionDetectionforSignLanguage

[36] Renotte, N. (n.d.-b). RealTimeSignLanguageDetectionwithTFJS [Source code]. GitHub. https://github.com/nicknochnack/RealTimeSignLanguageDetectionwithTFJS

[37] Samaan, G. H., Wadie, A. R., Attia, A. K., Asaad, A. M., Kamel, A. E., Slim, S. O., Abdallah, M. S., & Cho, Y.-I. (2022). MediaPipe's landmarks with RNN for dynamic sign language recognition. *Electronics, 11*(19), 3228. https://doi.org/10.3390/electronics11193228

[38] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., & Blake, A. (2011). Real-time human pose recognition in parts from single depth images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1297–1304. https://doi.org/10.1109/CVPR.2011.5995316

[39] Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., … Wattenberg, M. (2019). TensorFlow.js: Machine learning for the web and beyond. *Proceedings of Machine Learning and Systems (SysML)*. https://arxiv.org/abs/1901.05350

2026 S.T. Yau High School Science Award 仅用于2026丘成桐中学科学奖公示

[40] Starner, T., Weaver, J., & Pentland, A. (1998). Real-time American Sign Language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(12), 1371–1375. https://doi.org/10.1109/34.735811

[41] Sundar, B., & Bagyammal, T. (2022). American Sign Language recognition for alphabets using MediaPipe and LSTM. *Procedia Computer Science, 215*, 642–651. https://doi.org/10.1016/j.procs.2022.12.066

[42] United Nations. (n.d.). International Day of Sign Languages. https://www.un.org/en/observances/sign-languages-day

[43] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1706.03762

[44] Vicars, B. (n.d.). ASL University / Lifeprint: American Sign Language. Lifeprint.com. https://www.lifeprint.com/

[45] Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association, 22*(158), 209–212. https://doi.org/10.1080/01621459.1927.10502953

[46] World Health Organization. (2026). Deafness and hearing loss [Fact sheet]. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

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

2026 S.T. Yau High School Science Award
仅用于2026丘成桐中学科学奖论文公示