Design of a Lightweight Sign Language Recognition Model for On-Device Deployment

Chenxi Zhang*
Honors College, Ningbo University of Finance and Economics, Ningbo, Zhejiang, China
**Corresponding Author*

Abstract: This paper studies offline and real-time sign language recognition with a wrist-worn device. The model is selected according to the operators supported by the runtime. Recurrent structures are not used because static unrolling increases the INT8 model size to 988.2 KB; the final design uses five operator types: one-dimensional convolution, depthwise separable convolution, global average pooling, a fully connected layer and Softmax, with 11 366 parameters and an INT8 size of 29.3 KB. On the subject-disjoint split of UCI HAR, accuracy is 92.81% in floating point and 89.16% after INT8 quantization, statistically indistinguishable from a plain one-dimensional convolution with three times the compute ($p = 0.579$), while an iso-compute variant is 1.72 percentage points higher ($p = 0.037$). On the ESP32-S3, a single inference takes 15.94 ms with a 37.1 KB Tensor Arena and no PSRAM. On a self-collected 12-class dataset covering two subjects and seven sessions, same-subject, same-hand cross-session accuracy is 82.41% with the sensor left in place and rises to 90.29% after merging two easily confused classes; once the sensor is re-donned, cross-subject, cross-hand and cross-day accuracy falls to 28.94%–38.30%, and recovers to 77.49%–86.90% after gravity alignment and classifier-head fine-tuning using two samples per class. The results show that a model designed for a limited operator set can maintain its accuracy while using about one third of the computing cost. This advantage appears to come more from the model structure than from the computing budget.

Keywords: Sign Language Recognition; Model Quantization; Depthwise Separable Convolution; TinyML; On-Device Inference

1. Introduction

Sign language is an important way for deaf and hard-of-hearing people to communicate. An estimated 1.57 billion people were living with hearing loss in 2019, of whom about 430 million had moderate or higher hearing loss [1]. Previous studies have used inertial sensors for wearable sign language recognition [2,3], However, when a model is deployed on a microcontroller with only 512 KB of SRAM, the main limitation is not only its accuracy, but also the integer operators supported by the runtime. The target runtime is TensorFlow Lite Micro (TFLM) [4], and the target device is the ESP32-S3. The process starts with model selection and ends with on-device testing: first list the available operators, then remove structures that cannot be deployed, and finally design and quantize the model within these limits. The final model uses only five operator types, 11 366 parameters and an INT8 size of 29.3 KB; on ESP32-S3 a single inference takes 15.94 ms with a 37.1 KB Tensor Arena and no PSRAM; on the subject-disjoint split of UCI HAR [5], accuracy is 92.81% in floating point and 89.16% after quantization.

An important part of this selection is excluding recurrent networks, and the measurements support this decision. Sign language has clear temporal features, so a recurrent structure may seem like the natural first choice [6]; but it cannot be exported directly. Conversion succeeds only after the 128 time steps are statically unrolled, and unrolling grows the model from 431.7 KB to 988.2 KB, and after quantization its size reaches 2.3 times the floating-point size—the opposite of the expected effect of quantization. The accuracy is also lower: 85.77% in floating point, 8.40 percentage points below the iso-compute variant, with a 9.59-percentage-point drop after quantization, more than twice that of the convolutional models. Another problem appeared at this stage: conversion succeeded, yet the first firmware flashing attempt still failed because an operator inserted automatically by the framework had not

 Copyright @ STEMM Institute Press

been registered, and AllocateTensors aborted. Successful conversion does not mean that the model can be deployed.

To distinguish the effect of architecture from that of computing cost, four architectures are compared under one protocol, each repeated with five random seeds. The five-operator model is statistically indistinguishable from a plain one-dimensional convolution with three times the compute ($p = 0.579$), while the iso-compute widened variant is 1.72 percentage points higher ($p = 0.037$), which suggests that the difference may come more from the architecture than from the computing budget. However, this conclusion also has clear limits: the classes in UCI HAR are daily activities rather than sign language; when the self-collected 12-class wrist-motion dataset is used instead, with two subjects over seven sessions, same-subject same-hand cross-session accuracy is 82.41%, and after re-donning the sensor, cross-subject, cross-hand and cross-day accuracy falls to 28.94%–38.30%; gravity alignment alone restores 53.00%–54.02%, and one calibration of two samples per class brings it back to 77.49%–86.90%. This indicates that the domain shift is mainly addressed through lightweight adaptation during deployment, rather than by changing the model architecture.

The main contributions of this study can be summarised in three parts: an operator-first selection procedure and its evidence chain, which establishes what can be built for a given runtime; the five-operator model and its measurements, which evaluates how well the resulting model performs; and a four-architecture, five-seed comparison with an iso-compute control, which tests whether the advantage comes from architecture or computational budget. These three parts connect the design constraints with the experimental results. This also distinguishes the study from ordinary model compression — which mainly considers whether a model can be made smaller, while this study first considers whether it can run on the target device. Section 2 describes the design and quantization, Section 3 reports the comparison and the on-device measurements, and Section 4 concludes.

2. System and Model Design

2.1 Overall Architecture

The system has three stages: wearable data acquisition, on-device inference and recognition-result output. Each stage has its own constraints. The acquisition stage mainly depends on the sensor and communication bus: a wrist inertial measurement unit (IMU) outputs six-axis data at 50 Hz, 12 bytes per frame, and delivers them over I2C to the host. The inference stage is limited by computing power and memory: the host is an ESP32-S3 with 512 KB of SRAM at 240 MHz, the model must run entirely in integer arithmetic on chip, and intermediate tensors must not spill out of internal SRAM. The output stage is mainly limited by bandwidth: the output consists only of a class index and confidence score, which can be returned over a serial link. The three stages use fixed interfaces, so a change in one key parameter can affect the later stages: a higher sampling rate alters the window length, and a longer window raises the Tensor Arena footprint. For this reason, the design considers the available operators first, the model structure next and the inference time last. The overall framework is shown in Figure 1.

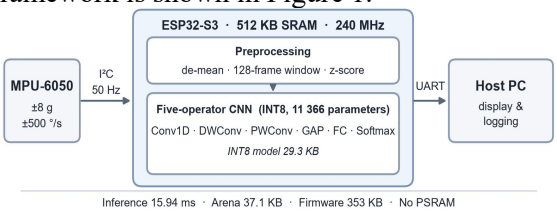


Figure 1. System Framework

2.2 Data Acquisition and Preprocessing

The data-acquisition process focuses on one question: can the same movement be performed and recorded consistently? The sensor is an MPU-6050 six-axis inertial unit, and only the three acceleration and three angular-rate channels are used, because wrist motion in sign language is captured by these two signal types, whereas a magnetometer, although it gives absolute orientation, is disturbed by ferromagnetic objects indoors and cannot be relied upon. The ranges are set to ±8 g and ±500 °/s, the former covering the usual peak acceleration during wrist movements and the latter covering angular velocities during rapid rotations. The impact peak during a slapping movement can reach 13.9 g, beyond the ±8 g range, and one sample was saturated in 7 of the 10 repetitions; the range was not raised to avoid this, because doing so would cost resolution for small movements. The sampling rate is set to 50 Hz, the same as the UCI HAR baseline for comparison; the fundamental frequency of wrist motion is of the order of a few hertz, so the 25

Hz Nyquist frequency provides ample margin. The device is placed on the back of the wrist, so it can record forearm rotation as well as components of finger motion transmitted through the wrist.

The preprocessing process contains three steps, each addressing a different type of variation. The first step is windowing: the input is fixed at 128 frames, i.e. 2.56 s at 50 Hz, which is expected to cover most of a movement — a shorter window truncates the movement, a longer one admits the silent interval between adjacent movements; movements in the self-collected data last 1.5–3 s, and 2.56 s accommodates most of them. The second step is normalization: per-channel z-score standardization removes differences of scale, computed from training-set statistics during training and from the target user's own calibration samples at deployment (Section 3.5); reusing training-set statistics after a change of wearer allows per-channel means and variances to shift, so that standardization amplifies the domain shift instead of removing it. The third step is data augmentation: time warping (0.9–1.1×) simulates variation in execution speed, additive Gaussian noise ($\sigma = 0.03$) simulates the sensor noise floor, and amplitude scaling (0.9–1.1×) simulates differences in movement amplitude across users; all three are applied to the training set only.

2.3 Model Design under Operator Constraints

The design is carried out in four steps, which gradually reduce the range of candidate models: the first fixes which operators may be used and confines candidate structures to what the runtime can express; the second removes structures that look suitable but cannot be deployed; the third fixes the topology within what remains; the fourth resolves how that topology is deployed and executed in integer precision. This order is important because leaving the first three decisions until deployment would increase the amount of repeated work. Step one: enumerate the available operators. The target runtime is TensorFlow Lite Micro (TFLM), and the target processor is the Xtensa core on ESP32-S3. The inspection shows that the runtime supports only a limited number of integer operators on Xtensa [4], which is much smaller than the set of operators available in a training framework: convolution, depthwise convolution, fully connected, pooling and activation are all available, whereas tensor reshaping, dynamic indexing and sequence control flow are only partially supported or absent altogether. This difference mainly comes from the memory limits of a microcontroller: TFLM uses static memory planning, all tensors are allocated at once during AllocateTensors, and shapes must be known at compile time; operations whose shapes must be inferred at run time are therefore only partially supported. This results in a clear requirement: the operator graph must be fully determined at compile time.

Step two: exclude recurrent structures. A recurrent model may seem suitable at first: sign language is a typical temporal signal, LSTM models long-range dependencies through gating and is widely adopted [6], and an earlier version of this work used that architecture. However, three problems were found during porting. First, it cannot be deployed directly: conversion experiments show that the recurrent structure cannot be exported to the target format, the error reporting an unsupported tensor-list operator, and only static unrolling of the 128 time steps makes export possible, at which point the model grows from 431.7 KB to 988.2 KB. Second, the quantization loss is large: the measured drop is 9.56–9.59 percentage points, well above the 3.65–4.40 points of the convolutional structures, because quantization error accumulates over 128 time steps. Third, step-by-step serial computation forces repeated reads and writes of intermediate tensors, which on a device with 512 KB of SRAM both occupies space and slows execution. These problems lead to one conclusion: this architecture is unsuitable for deployment on the target runtime.

Step three: fix the five-operator architecture. The final operator set includes one-dimensional convolution, depthwise separable convolution, global average pooling, fully connected and Softmax, The model is arranged in the following order: one-dimensional convolution to expand the channels → depthwise separable convolution for feature extraction → pointwise convolution to increase channel dimensionality → repeat once → global average pooling → fully connected layer → Softmax, with the tensor shape at each layer shown in Figure 2. Three design details allow the model to use integer-only inference. First, convolution carries no bias, and batch normalization is folded into the preceding weights at inference time, thereby eliminating an additional addition operation; folding must precede quantization, otherwise the

 Copyright @ STEMM Institute Press

quantization grid used during training would no longer match the one used during deployment. Second, the activation is ReLU6, whose output is bounded by [0, 6], reducing the risk that outliers will excessively widen the INT8 quantization scale [7,8]. Third, global average pooling replaces flattening: a conventional flatten of a 128 × 96 feature map would produce tens of thousands of input dimensions, and the fully connected layer would then account for most of the model parameters, which contradicts its role as a classifier.

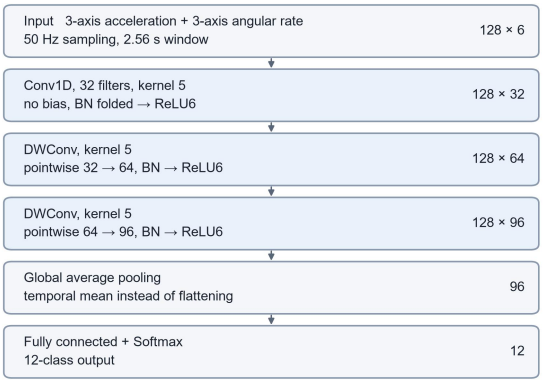


Figure 2. Structure of the Five-Operator Model: Tensor Shapes and Operator Composition at Each Layer

Step four: the quantization scheme. The quantization process uses two stages: post-training quantization (PTQ) PTQ is the main method, while quantization-aware fine-tuning (QAT) is used for comparison. PTQ estimates the activation range layer by layer using a calibration set, derives scale and zero point from it, and maps 32-bit floating point to 8-bit integers; its accuracy depends on whether the calibration samples cover the actual input range, and 300 training samples, about 4% of the training set, form the calibration set here. The latter applies both weight-level and activation-level fake quantization to the batch-normalization-folded model — per channel for weights, per tensor for activations — so that training becomes aware of quantization error; the order of these steps is important, because only the folded weights are the ones that actually take part in integer arithmetic at deployment [9]. Which of the two prevails is decided by experiment, and the result is reported in Section 3.3.

2.4 Model Lightweighting Pipeline

After training, three optimization steps are applied. Each step focuses on a different part of the model. The first step reduces the bit width: INT8 quantization lowers each weight from 4 bytes to 1, a theoretical 4× compression. The second step simplifies the computation graph: redundant operators are removed, which reduces graph nodes rather than weights, and the benefits are reflected in firmware size and scheduling overhead. The third step generates the deployment code: the model is converted to C++ for ESP32-S3 and embedded in the firmware as a flatbuffer, a step that only determines how the model is loaded. Combined, the model falls from 51.3 KB in floating point to 29.3 KB in INT8, a ratio of 1.8× rather than the theoretical 4×: in the flatbuffer of a small model, operator descriptions and tensor metadata are a fixed overhead that dilutes the saving from weight compression. Recurrent structures go the other way: unrolling makes graph nodes grow linearly with sequence length, and the INT8 size rises to 2.3 times the floating-point size.

3. Experiments

3.1 Dataset and Experimental Setup

Data and split. Public sign language datasets are all image- or video-based — the largest word-level dataset, WLASL, is built from videos [10] — and differ substantially in form from IMU data, so the controlled experiments are carried out on UCI HAR: 30 subjects, 50 Hz sampling and 128-frame windows, exactly the input of the model here (128 × 6) [5]. The official split is subject-disjoint (21 subjects and 7352 samples for training, 9 subjects and 2947 samples for testing), and no test subject appears during training, so the reported accuracy reflects cross-subject generalization; the self-collected dataset is validated in Section 3.5. Training setup. TensorFlow 2.21, batch size 32, Adam (learning rate 1×10^{-3}, gradient clipping 1.0), cosine annealing to zero, at most 60 epochs, early stopping on validation accuracy (patience 12) with restoration of the best weights, and a cross-entropy loss with label smoothing (0.1). Gradient clipping is essential for a recurrent network unrolled over 128 steps: without it the gradient norm fluctuates violently and training does not converge. Statistical protocol. Each configuration is trained independently with five seeds (42, 123, 2026, 7, 3407), which control initialization, split and augmentation sampling; means ± standard deviations are reported, architectures are compared with paired t-tests ($\alpha = 0.05$), and hyperparameters are fixed before

Copyright @ STEMM Institute Press

the experiments.

3.2 Candidate Architectures and Complexity Comparison

This study evaluates four architectures: a recurrent network (the intuitive option), a plain one-dimensional convolution (the improved option), the five-operator model (the proposed option, 1.23 M MACs) and an iso-compute widened version of it. The fourth exists because, without a compute control, the accuracy gap between the original model and a three-times-compute baseline cannot be attributed to architecture rather than to insufficient compute. The iso-compute version widens the channels from 32–64–96 to 64–128–148, so that its MACs differ from the plain convolution by 1.3% and its parameters by 6.6%. The complexity of the four is compared in Table 1.

Table 1. Complexity Comparison of the Four Candidate Architectures

Architecture	Parameters	MACs	FP model	INT8 model	Ratio
LSTM (unrolled)	22 726	2.30 M	431.7 KB	988.2 KB	0.4
Plain 1-D convolution	30 982	3.79 M	125.8 KB	42.6 KB	3.0
Five-operator (ours)	11 366	1.23 M	51.3 KB	29.3 KB	1.8
Five-operator, iso-compute	33 038	3.84 M	133.1 KB	56.2 KB	2.4

Note: sizes are measured after training (seed 42); all four architectures were exported to floating-point and INT8 files from the same batch-normalized model, so the rows are comparable.

As shown in Table 1, the five-operator model performs the same task with 37% of the parameters and 32% of the MACs of the plain one-dimensional convolution. One point to note is that the INT8 size of the recurrent network is larger than its floating-point size: after unrolling, the number of graph nodes grows linearly with sequence length, and the overhead from operator descriptions and tensor metadata outweighs the savings from weight compression, so the ratio falls to 0.4. Quantization gains are therefore not unconditional; they presuppose a sufficiently compact architecture.

3.3 Effect of Quantization on Accuracy

Each architecture is trained using five random seeds, resulting in 20 training runs in total; each run yields floating-point, PTQ and QAT states, 60 results in all, summarised in Table 2.

Table 2. Four Architectures × Three Quantization States (60 Results, Mean ± Standard Deviation over Five Seeds)

Architecture	FP	PTQ	Drop/pp	QAT	Drop/pp
LSTM (unrolled)	85.77 ± 2.64%	76.18 ± 2.78%	9.59	76.21 ± 3.15%	9.56
Plain 1-D convolution	92.45 ± 0.74%	88.75 ± 0.76%	3.69	88.50 ± 0.67%	3.95
Five-operator (ours)	92.81 ± 0.75%	89.16 ± 0.79%	3.65	88.48 ± 0.71%	4.33
Five-operator, iso-compute	94.17 ± 0.59%	89.77 ± 1.47%	4.40	88.71 ± 1.45%	5.46

Note: the drop is the difference between floating-point accuracy and the accuracy of that quantization state, in percentage points (pp).

The proposed model performs better when the computing cost is similar. With parameters (33 038 against 30 982, a 6.6% difference) and MACs (3.84 M against 3.79 M) closely matched, the iso-compute version reaches 94.17% in floating point, 1.72 percentage points above the plain one-dimensional convolution, with a paired t-test giving $p = 0.037$, significant at $\alpha = 0.05$. This control weakens the explanation of insufficient compute, and the gap more likely comes from architecture; but widening the channels also changes the feature dimensionality, so the attribution has limits (see Section 4).

The accuracy does not show a clear decrease even when the computing cost is reduced to about one third. The original-size model reaches 92.81% with 11 366 parameters and 1.23 M MACs, 0.36 percentage points from the plain one-dimensional convolution at three times the compute (92.45%), with $p = 0.579$; no significant difference was detected between the two. Taken together, the proposed architecture is more accurate at equal compute and shows no detectable loss at lower compute.

The recurrent network has two clear disadvantages. Its floating-point accuracy of 85.77% is 8.40 percentage points below the iso-compute version ($p = 0.0019$) and 6.68 points below the plain convolution; its INT8 model is 988.2 KB, about 34 times that of the five-operator model. This baseline was given gradient clipping and cosine annealing and repeated over five seeds, so its disadvantage is hard to explain by insufficient training.

 Copyright @ STEMM Institute Press

The accuracy drop does not simply follow the number of parameters. It is 9.59 points for the recurrent network, 4.40 for the iso-compute version, 3.69 for the plain convolution and 3.65 for the five-operator model. The first may stem from the accumulation of quantization error over 128 time steps, the last from large per-channel differences in dynamic range that a single scale cannot cover, so the cost of quantization must be analysed together with the type of architecture.

Quantization-aware fine-tuning. Weight-level and activation-level fake quantization were both applied to the folded model, yet QAT did not outperform PTQ on any architecture except the recurrent network, where it gained 0.03 percentage points (88.48% against 89.16% for the five-operator model). One possible reason is that activation fake quantization uses the ReLU6 prior bound, which is inconsistent with the range the converter derives from the calibration set, and therefore imposes no effective constraint. Further improvement with QAT will require a more complete framework.

3.4 On-Device Measurements

The measurements were carried out on an ESP32-S3 (dual-core 240 MHz, 8 MB flash, 8 MB PSRAM) with TensorFlow Lite Micro as the runtime and esp-nn integer vector acceleration enabled. A fixed test vector was used for 500 inference runs. The on-board high-resolution timer recorded each run, and the median value was used. The model measured is the one trained with seed 42 under the protocol of Section 3.1 and exported after PTQ, whereas Table 2 reports means over five seeds; for a fixed structure, latency is determined mainly by the operator graph and the compute, and varies little with the weight values (only the seed-42 model was measured here), so the conclusions drawn from Table 2 are also applicable to Table 3. Table 3 lists the measured results, while Table 4 compares the four architectures using the same method.

Table 3. Measured On-Device Metrics on ESP32-S3

Metric	Measured
Single-inference latency (median)	15.94 ms
Latency range (min–max, 500 runs)	15.92–15.97 ms
P99 latency	15.97 ms
Tensor Arena peak	37.1 KB (64 KB allocated, 57.9% utilization)
Firmware size	353 KB
Registered operators	7
PSRAM usage	0

Note: latency is measured from writing the quantized input tensor into the buffer to dequantizing the output, excluding sensor sampling and data transfer. Jitter over 500 runs is 0.05 ms (0.3%); the CPU is yielded once every 20 inferences to avoid the watchdog, outside the timed interval. The on-board output was class 4 with confidence 0.754, consistent with the ground-truth label and identical to the host-side output for this sample, which serves as a preliminary check of numerical consistency on device; the Tensor Arena lies entirely in internal SRAM, so the model can migrate to low-cost parts without PSRAM.

Table 4. On-Device Inference Comparison of the Four Candidate Architectures

Architecture	INT8 accuracy	Model size	Median latency	P99 latency	Relative latency
Five-operator (ours)	89.16%	29.3 KB	15.94 ms	15.97 ms	1.00
Plain 1-D convolution	88.75%	42.6 KB	20.64 ms	20.66 ms	1.29
Five-operator, iso-compute	89.77%	56.2 KB	28.99 ms	29.02 ms	1.82
LSTM (unrolled)	76.18%	988.2 KB	Failed (no result)	—	—

Note: accuracy is taken from the five-seed means of Table 2; latency is measured on ESP32-S3 with the seed-42 model over 500 consecutive inferences, and for a fixed structure depends on the operator graph and the compute, varying little with the weight values; relative latency is normalized to the proposed model; the recurrent network failed in the tensor allocation stage and therefore has no latency figure.

Table 4 compares all four architectures on the same board and under the same measurement protocol. The five-operator model and the plain one-dimensional convolution have almost the same INT8 accuracy (89.16% and 88.75%), yet the proposed model completes inference 22.8% faster; the iso-compute version is the most accurate but takes 1.82 times as long to infer as the proposed model, so matching MACs does not bring a proportional latency benefit. The recurrent network could not complete inference at all: its unrolled graph contains a dynamic-shape chain built through SHAPE and FILL, and the runtime aborts during tensor preparation;

even with the Tensor Arena enlarged to 1 MB and placed in PSRAM it still fails — the problem is not memory capacity, but the runtime's handling of dynamic-shape operators during tensor preparation.

3.5 Behavioral Validation on a Self-Collected Dataset

A 12-class wrist-motion dataset was collected to test the method with the actual sensor pipeline. The sensor is an MPU-6050 with ranges of ±8 g and ±500 °/s and a sampling rate of 50 Hz, worn on the back of the wrist and usable on either hand; subjects performed each of the 12 classes 10 times under a unified protocol with 2 s of silence between classes. Two subjects and seven sessions were recorded: three left-hand and one right-hand session for subject 1, one left-hand and two right-hand sessions for subject 2, giving 936 samples after bias removal, gravity normalization and 128-frame windowing. The test conditions are grouped according to whether the sensor was removed and worn again: same-session, cross-session and cross-domain (cross-subject, cross-hand and cross-day), and the recognition results for each condition are given in Table 5, while the confusion matrix for the cross-session A→B condition is shown in Figure 3. All results below are means ± standard deviations over seeds 0, 1 and 2.

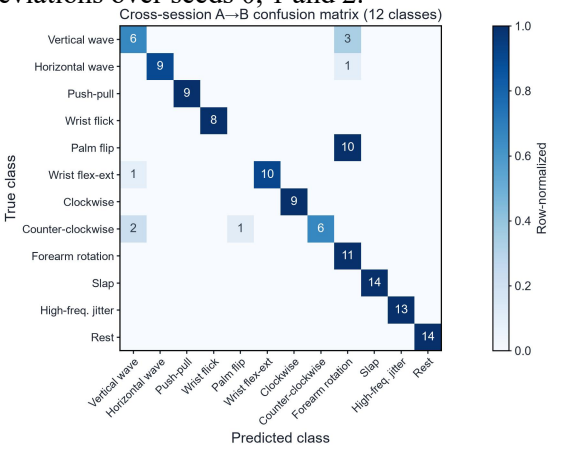


Figure 3. Confusion Matrix for the Cross-Session A→B Condition (One Representative Run, Seed 0)

Table 5. Recognition Results on the Self-Collected 12-Class Dataset

Setting	Classes	Accuracy	Remark
Same-session, 5-fold (session A)	12	92.25 ± 2.69%	3 seeds
Cross-session (same subject, same hand, sensor not removed)	12	82.41 ± 2.99%	session B unseen in training
Cross-session, 11 classes merged	11	90.29 ± 2.98%	palm flip merged with pronation
Cross-subject (re-donned)	12	38.30 → 54.02 → 77.49%	two samples per class
Cross-hand (re-donned)	12	28.94 → 53.00 → 86.90%	two samples per class
Cross-day (re-donned next day)	12	34.66 → 38.38 → 81.31%	two samples per class
All sessions, 5-fold	12	97.37 ± 0.56%	upper reference
Cross-session (INT8)	12	86.61%	28.4 KB, 11 948 parameters

Note: the three cross-domain rows report accuracy in the order raw coordinates → gravity alignment → after calibration; calibration uses two samples per class and the normalization statistics are estimated from the calibration samples; the remaining conditions are means over three seeds.

When the same subject uses the same hand without removing the sensor, accuracy decreases but remains within a usable range. Five-fold cross-validation within session A gives 92.25 ± 2.69%, and the cross-session A→B condition falls to 82.41 ± 2.99%. Both sessions were performed by the same subject in one wearing state without removing the sensor in between, so the decrease mainly reflects natural variation in amplitude and tempo and contains no change of coordinate frame; this also provides the baseline for what follows — any condition markedly below 82.41% can be attributed to a change of wearing state. Figure 3 shows where the loss occurs: the recall of palm flip is 0, all 10 samples being classified as forearm pronation — both share the pronation/supination degree of freedom and differ only in whether the motion is a single reciprocation or a continuous rotation in one direction, and the natural variation of amplitude and tempo between sessions is enough to swamp that difference; merging the two classes raises accuracy from 82.41% to 90.29%. Three further samples of vertical waving were misclassified as slapping, both being fast vertical movements that differ only in impact strength. The separability constraint of the class set therefore comes not only from the sensor modality but from the degrees of freedom themselves.

After the sensor is removed and worn again, accuracy drops to around 30%. Cross-subject drops to 38.30 ± 7.43%, cross-hand to 28.94 ± 6.71% and cross-day to 34.66 ± 6.08%, only 3.5–4.6 times the 12-class chance level of 8.3%.

 Copyright @ STEMM Institute Press

The cross-day condition deserves particular attention: same subject, same hand, adjacent days, identical movement protocol and execution habits, with re-donning as the main variable, yet accuracy is on a par with the cross-subject condition. What breaks the model is therefore not the difference between individuals but the wearing state itself; by contrast, five-fold cross-validation over all sessions reaches 97.37 ± 0.56%, which shows that the classes themselves are separable in the feature space.

This result is related to the rotation and mirroring of the coordinate system: the model learns its criteria from the absolute readings of acceleration and angular rate, whereas the directions of the three axes are fixed by how the device is attached to the wrist, so after re-donning the yaw about the gravity axis changes and the projection of the same movement onto each axis changes with it, invalidating the criteria. The cross-hand condition exposes a deeper cause — the relation between the two hands' frames is a mirroring rather than a rotation, the same movement has opposite handedness in the two frames, and clockwise and counter-clockwise circles are therefore the pair most easily swapped, which cannot be corrected by rotation alone. Gravity alignment alone (rotating the window-averaged gravity direction to +z) restores cross-subject and cross-hand to 54.02% and 53.00%, while cross-day stays at 38.38%, consistent with the judgement that gravity alignment corrects only the axial direction and leaves yaw and mirroring untouched.

For this reason, a simple user-specific calibration is added during deployment after gravity alignment. The target user performs each of the 12 classes twice (about 2 minutes); the 24 samples estimate the normalization statistics, and only the last fully connected layer (12 × 96 parameters) is fine-tuned while the remaining weights are frozen. Cross-subject rises to 77.49 ± 4.45%, cross-hand to 86.90 ± 7.64% and cross-day to 81.31 ± 6.82%. Calibration and test samples are strictly disjoint and the statistics are estimated from the former alone; when the test samples are used instead, all three change by less than 0.6 percentage points, so the conclusion does not depend on test-set information. The improvement becomes smaller as more samples are used: one sample per class already gives 72.83%, 83.68% and 72.52%, and three add only 2.4, 1.9 and 1.6 points, so two are taken as the operating point — even one sample, under a minute, is enough to lift cross-subject from 38% to 73%. This process requires one forward pass and one backward pass on the device, with no retraining and no data upload, and calibration is needed once whenever the wearing state changes.

This study has three main limitations. First, only two subjects took part, so the cross-domain and calibration results are preliminary and their validity needs confirmation with more subjects. Second, the impact peak of slapping reaches 13.9 g, beyond the ±8 g range, and 7 of 10 repetitions saturated one sample. Third, each class has 65–101 samples, and the dataset is positioned as a validation of link feasibility rather than a performance evaluation. As for quantization, the INT8 model on this dataset is 28.4 KB with 11 948 parameters, reaching 86.61% cross-session, comparable to the 85.83% of the floating-point model with the same seed.

4. Conclusion

This study proposes an operator-first model-selection procedure and validates it with a four-architecture, five-seed comparison. The five-operator model achieves 92.81% floating-point and 89.16% quantized accuracy with 11 366 parameters, shows no detectable difference from a plain one-dimensional convolution with three times the compute ($p = 0.579$), and the iso-compute version is 1.72 percentage points higher ($p = 0.037$); on ESP32-S3 a single inference takes 15.94 ms with a 37.1 KB Tensor Arena and no PSRAM. These results suggest that the model structure should be designed according to the available operators, while keeping its accuracy at a reasonable level.

There are still three limitations in this study. First, the gap between the data domain and sign language. All controlled experiments were carried out on UCI HAR. Its subject-disjoint split reflects cross-subject generalization, but its classes are daily activities rather than sign language. Both are six-axis inertial time series, so conclusions about architecture and quantization behaviour transfer, while absolute accuracy cannot be extrapolated directly to sign language. The self-collected data further show that cross-subject, cross-hand and cross-day accuracy is far below same-session accuracy, and that one calibration of two samples per class restores 77%–87%, so the remedy for domain shift lies mainly not in the architecture but in a

Copyright @ STEMM Institute Press

lightweight adaptation at deployment. Second, the construction of the iso-compute control. The iso-compute version matches the budget by widening channels, which also changes the feature dimensionality, so that comparison proves superiority at an equal budget rather than at any size. Third, power and cross-platform comparison. Power consumption was not evaluated in this round; wearable devices are power-sensitive and the metric is left for later work. All four architectures were measured by this work on one platform and one split and are mutually comparable, but the figures quoted from the literature come from different hardware platforms and should be compared with caution. Future work will implement the calibration process in firmware, include more subjects, measure power consumption and further test the effect of full quantization-aware training.

References

[1] L. M. Haile, K. Kamenov, P. S. Briant, et al. Hearing loss prevalence and years lived with disability, 1990-2019: findings from the Global Burden of Disease Study 2019. The Lancet, 2021, 397(10278):996-1009.

[2] S. Sharma, R. Gupta, A. Kumar. A TinyML solution for an IoT-based communication device for hearing impaired. Expert Systems with Applications, 2024, 246:123147.

[3] Y. Gu, C. Zheng, M. Todoh, et al. American sign language translation using wearable inertial and electromyography sensors for tracking hand movements and facial expressions. Frontiers in Neuroscience, 2022, 16:962141.

[4] R. David, J. Duke, A. Jain, et al. TensorFlow Lite Micro: Embedded machine learning on TinyML systems. Proceedings of Machine Learning and Systems, 2021, 3:800-811.

[5] D. Anguita, A. Ghio, L. Oneto, et al. A public domain dataset for human activity recognition using smartphones. Proceedings of the 21st European Symposium on Artificial Neural Networks, Bruges, 2013:437-442.

[6] S. Hochreiter, J. Schmidhuber. Long short-term memory. Neural Computation, 1997, 9(8):1735-1780.

[7] A. G. Howard, M. Zhu, B. Chen, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[8] M. Sandler, A. Howard, M. Zhu, et al. MobileNetV2: Inverted residuals and linear bottlenecks. Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, 2018:4510-4520.

[9] B. Jacob, S. Kligys, B. Chen, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, 2018:2704-2713.

[10] D. Li, C. Rodriguez, X. Yu, et al. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. Proceedings of the 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), Snowmass, 2020:1448-1458

 Copyright @ STEMM Institute Press