Innovation and Technology Studies
Print ISSN: 3007-6919
Online ISSN: 3007-6927
DOI: https://doi.org/10.61784/its4025

PREFIXKD-EDGE: REAL-TIME LOCOMOTION-INTENT PREDICTION WITH PREFIX SELF-DISTILLATION ON LIGHTWEIGHT TEMPORAL NETWORKS

WenBin Shang[1*], TingJie Chen[2]
[1]*University of Glasgow, Glasgow G12 8QQ, U.K.*
[2]*Intel, Shanghai 200333, China.*
[*]*Corresponding Author: WenBin Shang*

Abstract: Edge intent inference must make useful decisions before a sensing window is complete, while respecting tight memory and latency budgets. This paper studies early locomotion-mode prediction from wearable inertial sequences and introduces PrefixKD-Edge, a deployment-oriented framework that combines a compact temporal convolutional backbone, training-only full-window-to-prefix self-distillation, and confidence-gated decision timing. The task is evaluated on the public UCI Human Activity Recognition Using Smartphones dataset using its official subject-disjoint split. We retain three dynamic classes—level walking, stair ascent, and stair descent—and use the six supplied body-acceleration and gyroscope sequence channels rather than the 561 engineered features. A causal four-sample block mean reduces 50-Hz windows to 12.5 Hz, after which predictions are tested at 25%, 50%, 75%, and 100% observation. Across three random seeds, the 7,907-parameter PrefixKD-CNN attains macro-F1 scores of 94.30±1.76%, 97.64±1.62%, 98.00±1.43%, and 98.13±1.33%, respectively. Its measured FP32 TorchScript batch-1 median latency is 0.031–0.034 ms on one server CPU thread. With a confidence threshold of 0.90, the adaptive policy reaches 98.20±1.18% accuracy while observing 42.72±1.97% of the window on average (1.09 s). The same-backbone distillation gain is positive but small, so no statistical-significance claim is made. The results show that hardware-friendly backbones, prefix-aware training, and explicit decision policies can provide a practical accuracy–earliness operating point without adding inference parameters.
Keywords: Early time-series classification; Edge intelligence; Human activity recognition; Knowledge distillation; Lightweight neural networks; Wearable sensing

1 INTRODUCTION

Wearable and mobile sensing applications increasingly require decisions on the device rather than after a complete record has been uploaded. Examples include gait-aware assistance, context-dependent user interfaces, safety monitoring, and adaptive sensor duty cycling. In such settings, the relevant question is not only whether a model is accurate, but how early it becomes sufficiently reliable, how much state it occupies, and whether its actual implementation is fast on commodity processors. Human activity recognition (HAR) research has established strong convolutional and recurrent sequence models, yet most benchmark protocols report one prediction after an entire pre-segmented window [1-9]. This masks the observation delay that dominates many interactive systems.

Early time-series classification explicitly couples accuracy with earliness [10-12]. However, methods designed for large offline datasets often maintain multiple classifiers, search complex stopping policies, or assume that computation is negligible relative to sensing. Edge deployment creates a different design regime: a single compact network should work at several prefix lengths; the training procedure should make short prefixes discriminative; and the stopping rule should be transparent enough to audit. Moreover, parameter count alone is an unreliable proxy for latency because grouped, recurrent, and dilated operators can behave very differently on a real runtime.

This work focuses on early recognition of locomotion mode from smartphone inertial sequences. We use the term intent operationally: the system infers the user's current locomotion mode from the earliest available part of a window, enabling a downstream controller to react before the full 2.56-s window has elapsed. This is not pre-onset forecasting of a future transition, because the UCI HAR windows are already segmented from ongoing activities. The narrower definition avoids overstating what the dataset can support while preserving a meaningful edge decision problem.

The proposed PrefixKD-Edge framework has three components. First, a deterministic four-sample causal block mean lowers the sensing rate from 50 to 12.5 Hz and suppresses short-scale noise. Second, a 7.9k-parameter TinyCNN is trained on both full windows and randomly sampled early prefixes. Its full-window logits serve as a detached teacher for the early-prefix logits, transferring class evidence without introducing a separate teacher model or any inference-time parameters. Third, a confidence gate evaluates predictions at four checkpoints and returns the earliest one that exceeds a threshold, otherwise falling back to the complete window.

The contributions are: (1) a prefix self-distillation objective for a hardware-friendly compact temporal CNN, requiring only two forward passes per training batch and no additional deployment branch; (2) an explicit accuracy–observation policy that converts prefix probabilities into a controllable decision delay; (3) a reproducible subject-disjoint evaluation with three seeds, several lightweight baselines, a same-architecture ablation, measured single-thread TorchScript

latency, parameter counts, and serialized model sizes; and (4) a negative-result-aware analysis showing that a more elaborate grouped multi-scale temporal network is slower and less accurate at the shortest prefix than the simple CNN, despite having a seemingly efficient structure, see Figure 1.

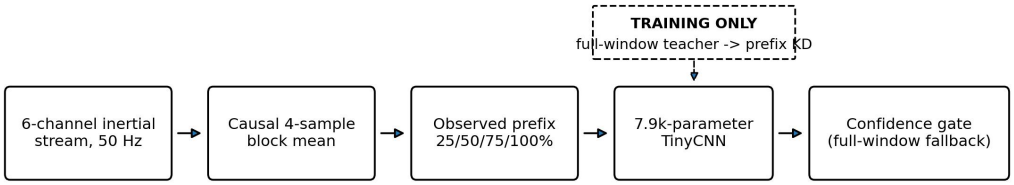

Figure 1 PrefixKD-Edge Overview

Note: The full-window branch is used only as a detached teacher during training; deployment uses the same compact CNN at each observed prefix.

2 RELATED WORK

2.1 Wearable Activity Recognition

Classical wearable HAR pipelines combine windowing, handcrafted statistics, and shallow classifiers; comprehensive surveys describe their sensor, placement, segmentation, and evaluation choices [1-2]. Deep models reduce feature engineering by learning directly from multichannel sequences. Convolutional models have been applied to smartphone HAR, while recurrent and hybrid convolutional–recurrent models capture temporal dependencies [3-8]. DeepSense provides a unified convolutional/recurrent framework for mobile sensing [9]. These works establish strong full-window recognition, but their common evaluation target does not directly measure how performance evolves as only a prefix becomes available.

2.2 Temporal Models for Edge Inference

LSTM and GRU units are standard sequence baselines [13-14]. Temporal convolutional networks (TCNs) use causal or dilated convolutions to obtain long receptive fields and parallel execution [15-17]. MobileNets and MobileNetV2 demonstrate the efficiency of depthwise and inverted-residual convolution for images, and post-training integer arithmetic can further reduce deployment cost [18-20]. TinyML systems such as MCUNet, TensorFlow Lite Micro, and MicroNets emphasize that operator support, memory planning, and platform-aware measurement are as important as nominal model size [21-24]. Motivated by this evidence, our final method uses ordinary dense 1-D convolutions that map efficiently to the tested CPU runtime; grouped multi-scale blocks are retained only as an ablation.

2.3 Early Classification, Distillation, and Exits

Early classification methods optimize when a label should be emitted, not just which label is correct [10-12]. Neural early-exit systems attach intermediate classifiers to a deep network, and selective classification abstains when confidence is insufficient [25-26]. Because confidence can be miscalibrated, thresholds should be reported as operating points rather than universal guarantees [27]. Knowledge distillation transfers soft targets from a teacher to a student [28]; self-distillation uses the model's own stronger predictions or deeper representations [29]. PrefixKD differs from architectural early exits: the same backbone processes every prefix, while the complete-window prediction acts as a training-only teacher for a shorter observation of the same sample.

3 PROBLEM FORMULATION

Let X = [x_1,…,x_T] be a six-channel inertial window with class y in {1,2,3}. The original UCI sequence has T=128 samples at 50 Hz. A causal four-sample block mean produces X-bar with L=32 steps. At checkpoint ratio r in R={0.25,0.50,0.75,1.00}, the classifier receives only the first L_r=max(4, round(rL)) steps. The resulting durations are 0.64, 1.28, 1.92, and 2.56 s. No input after L_r is exposed to the model.

$$z_r = f_\theta(\bar{X}[1{:}L_r]), \quad p_r = \mathrm{softmax}(z_r) \tag{1}$$

Evaluation reports accuracy and macro-F1 independently at every checkpoint. For adaptive inference, confidence c_r=max_k p_r,k. Given threshold tau, the decision ratio is the first checkpoint whose confidence reaches tau; if none does, the full window is used. This policy separates model quality from a deployment preference: a safety-sensitive application can increase tau, whereas a latency-sensitive interface can choose a lower value.

$$r* = \min\{r \in R : c_r \geq \tau\}, \text{ with } r^* = 1 \text{ if no early checkpoint qualifies} \tag{2}$$

4 PREFIXKD-EDGE METHOD

4.1 Causal Sensor Front-End

The six sequence channels are body acceleration along three axes and gyroscope measurements along three axes as supplied by the UCI repository. They are sequence-level inputs, not the dataset's 561 engineered features. Four consecutive samples are averaged with a four-sample buffer. In an online implementation, the operation emits one step every 80 ms and never requires future samples. The reduction decreases the temporal length by four while preserving the 2.56-s physical window. Channel-wise means and standard deviations are estimated only from the seed-specific training subset and applied to validation and test data.

4.2 Hardware-Friendly TinyCNN Backbone

The backbone contains three dense 1-D convolution layers: 6→24 channels with kernel 5, 24→32 with kernel 5, a length-2 max-pool, and 32→32 with kernel 3. Each convolution is followed by batch normalization and ReLU [30]; global temporal average pooling feeds a three-class linear layer. The implementation has 7,907 trainable parameters. Symmetric padding is applied only within the supplied prefix, so the model never observes uncollected samples; the current reference implementation recomputes the network at each checkpoint rather than maintaining a streaming convolution cache.

4.3 Stochastic Multi-Prefix Supervision

Naively evaluating all four prefixes during every update requires four forward/backward paths. We instead always train on the complete window and uniformly sample one early ratio from {0.25,0.50,0.75} for each batch. Across epochs, all early checkpoints receive supervision, while each batch requires two forward passes. Let CE denote cross-entropy. The base multi-prefix objective is the average of the early and complete-window losses.

$$L_{\mathrm{CE}} = 0.5\mathrm{CE}(z_r, y) + 0.5\mathrm{CE}(z_1, y) \tag{3}$$

4.4 Full-to-Prefix Self-Distillation

The complete window contains evidence unavailable to the short prefix. PrefixKD converts this asymmetry into a training signal. The full-window probabilities are detached from the gradient graph and softened by temperature T_d=2. The early prediction is optimized toward this distribution with KL divergence. With alpha=0.35, the total objective is

$$L = L_{\mathrm{CE}} + \alpha T_d^2 \mathrm{KL}\big(\mathrm{softmax}(z_1/T_d)\big\|\mathrm{softmax}(z_r/T_d)\big) \tag{4}$$

The teacher and student share all weights, but detachment prevents the prefix loss from moving the teacher target in the same update. The full branch remains directly supervised by CE. At deployment, the full branch is not an extra network: it is simply the same backbone evaluated only when sufficient observations exist. Consequently, PrefixKD-CNN and TinyCNN have identical trainable parameter counts and nearly identical serialized sizes.

4.5 Confidence-Gated Decision Policy

At run time, the classifier is evaluated at the four fixed checkpoints. The earliest probability maximum exceeding tau is returned. The policy is deliberately simple, reproducible, and independent of the training loss. It does not claim calibrated risk guarantees; instead, we report the empirical accuracy, macro-F1, mean observation ratio, median observation ratio, and distribution of selected checkpoints for thresholds 0.70–0.95. Observation savings refer to sensing and decision delay. Because the reference code recomputes inference at each checkpoint, cumulative compute savings are not claimed.

5 EXPERIMENTAL PROTOCOL

5.1 Dataset and Split

Table 1 Dataset and Evaluation Protocol

Item	Setting
Dataset	UCI HAR smartphone sequences
Classes	Walk / upstairs / downstairs
Development / test	3,285 / 1,387 windows
Input	6 channels; 128 samples at 50 Hz
Front-end	4-sample causal mean; 32 steps
Checkpoints	0.64 / 1.28 / 1.92 / 2.56 s
Seeds	13, 37, 73

UCI Human Activity Recognition Using Smartphones contains recordings from 30 volunteers performing six activities with waist-mounted smartphone inertial sensors [31-32]. We preserve the official subject-disjoint development/test split and select the three dynamic locomotion classes WALKING, WALKING_UPSTAIRS, and

WALKING_DOWNSTAIRS, see Table 1. This yields 3,285 development windows and 1,387 test windows. For each seed, the official development split is divided stratifiably into 2,792 training and 493 validation windows; the held-out official test subjects are never used for normalization, model selection, or threshold training.

5.2 Baselines and Ablations

TinyCNN is the same backbone trained with stochastic multi-prefix cross-entropy but without distillation. TinyGRU uses one 32-unit GRU layer. TinyTCN uses a 24-channel stem and four residual causal blocks with dilations 1, 2, 4, and 8. GMS-CE is an exploratory 24-channel grouped multi-scale network with parallel kernel-3 and kernel-5 branches at the same dilation schedule; GMS-KD adds the same self-distillation objective. All methods receive identical inputs, prefix checkpoints, train/validation splits, optimizer settings, and early-stopping criterion, enabling clean attribution of the PrefixKD term within a backbone.

5.3 Optimization and Model Selection

Models are trained in PyTorch with AdamW, initial learning rate 10^-3, weight decay 10^-4, cosine annealing following the SGDR schedule family, gradient-norm clipping at 5, batch size 256, at most 30 epochs, and patience 6. The selected checkpoint maximizes validation macro-F1 averaged over all four observation ratios [33-35]. We report test mean and sample standard deviation across seeds 13, 37, and 73. The experiment therefore quantifies optimization variability, but three seeds are insufficient for a strong significance claim.

5.4 Latency and Size Measurement

For each representative seed-13 model, batch-1 FP32 inference is traced and frozen with TorchScript. Latency is measured on one thread of an Intel Xeon Platinum 8573C after 80 warm-up iterations, followed by 400 timed iterations per prefix. We report median and 95th-percentile latency. Model size is the serialized PyTorch state dictionary, while parameter count includes trainable parameters. These measurements are reproducible implementation indicators, not direct microcontroller energy or peak-RAM measurements.

6 RESULTS

6.1 Accuracy as Observation Grows

Table 2 and Figure 2 show a consistent early-observation hierarchy. PrefixKD-CNN achieves 94.30±1.76% macro-F1 at 25% observation and rises to 98.13±1.33% at the full window. The compact CNN family is markedly stronger than the GRU at short prefixes: the proposed method exceeds TinyGRU by 11.19 percentage points at 25%, while also using a short, highly parallel convolutional path. TCN is competitive at 50–100%, but trails PrefixKD-CNN by 1.95 points at 25%.

Table 2 Macro-F1 (%) Across Observation Ratios (Mean±SD, Three Seeds)

Model	25%	50%	75%	100%
PrefixKD-CNN	94.30±1.76	97.64±1.62	98.00±1.43	98.13±1.33
TinyCNN	94.10±1.53	97.51±1.70	97.93±1.64	97.96±1.65
TCN	92.34±0.98	97.22±0.60	97.64±1.02	98.10±1.01
GRU	83.11±2.45	91.30±0.70	92.32±1.12	93.07±1.11
GMS-CE	90.94±1.09	96.98±0.55	97.65±0.76	98.12±1.02
GMS-KD	92.14±0.71	96.94±0.88	97.68±0.69	98.10±1.03

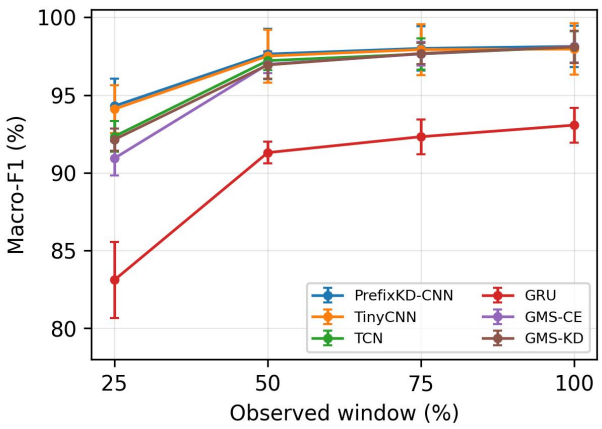

Figure 2 Macro-F1 Versus Observed Window
Note: Error bars are sample standard deviations across three seeds.

The same-backbone comparison is intentionally modest. PrefixKD improves TinyCNN by 0.20, 0.13, 0.08, and 0.17 percentage points at 25%, 50%, 75%, and 100%, respectively. Per-seed differences at 25% are +0.47, -0.01, and +0.13 points. Thus, the average gain is positive but not stable enough to claim statistical significance. The result should be interpreted as a low-cost regularization effect rather than a universal accuracy improvement. In contrast, the grouped multi-scale architecture exhibits a larger within-architecture early gain: GMS-KD improves GMS-CE from 90.94% to 92.14% at 25%, a 1.20-point increase, while their complete-window scores are essentially equal.

6.2 Seed-Level Stability and Model Selection

Seed 13 is the strongest run for both compact CNN variants, while seed 37 is the weakest. Because the subject-disjoint test set is fixed, this spread is caused by the seed-dependent 15% validation split, initialization, and minibatch order rather than by resampling test subjects. The checkpoint selected by average validation macro-F1 occurs between epochs 16 and 19 for PrefixKD-CNN and between epochs 15 and 21 for TinyCNN. Reporting all three seeds prevents the representative confusion matrix or the best run from being mistaken for the expected result.

Model ranking is most stable at the extremes: both CNN variants lead at 25%, and all convolutional models converge near 98% at the full window. The middle checkpoints have overlapping standard-deviation intervals. Accordingly, the paper emphasizes deployment operating points and same-backbone attribution rather than claiming that every pairwise ranking is statistically resolved.

6.3 Why the Simple Backbone Is the Main Method

The grouped multi-scale model was initially considered the primary architecture because grouped temporal branches appear parameter-efficient. Actual measurements reversed that choice. GMS-KD has 14,283 parameters, a 71.32-KiB state dictionary, and 0.403-ms full-window median latency, whereas PrefixKD-CNN has 7,907 parameters, a 38.44-KiB state dictionary, and 0.033-ms latency. The grouped operator is over twelve times slower on the tested CPU despite having fewer parameters than many conventional networks. This negative result supports a systems-level lesson: an edge paper should select operators using measured runtime behavior, not parameter count or asymptotic intuition alone, see Table 3 and Figure 3.

Table 3 Model Size and Full-Window CPU Latency

Model	Params	KiB	Med. ms	P95 ms
PrefixKD-CNN	7,907	38.44	0.033	0.041
TinyCNN	7,907	38.32	0.033	0.041
GRU	3,939	18.29	0.433	0.524
TCN	14,427	73.43	0.168	0.207
GMS-KD	14,283	71.32	0.403	0.577

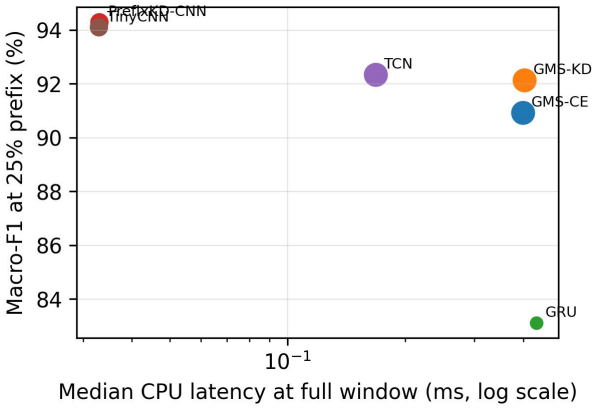

Figure 3 Measured Deployment Trade-Off

Note: Marker area is proportional to parameter count; latency uses a logarithmic axis.

6.4 Confidence-Gated Decision Timing

Table 4 reports the adaptive policy. At tau=0.70, the model observes only 30.37±0.62% of the window on average, but accuracy is 97.33±1.75%. Raising tau to 0.90 increases accuracy to 98.20±1.18% while using 42.72±1.97% of the window, equivalent to 1.09±0.05 s. Relative to waiting 2.56 s, this is a 57.28% reduction in mean observation delay. At tau=0.95, the mean observation rises to 53.44% with no meaningful average accuracy gain over tau=0.90, indicating a useful knee near 0.90 for this dataset, see Figure 4.

Table 4 Confidence-Gated Prefix Selection (Mean±SD, Three Seeds)

tau	Accuracy	Macro-F1	Mean obs.
0.70	97.33±1.75	97.31±1.77	30.37±0.62
0.80	97.89±1.34	97.87±1.35	34.38±1.66
0.90	98.20±1.18	98.18±1.20	42.72±1.97
0.95	98.22±1.24	98.20±1.26	53.44±2.96

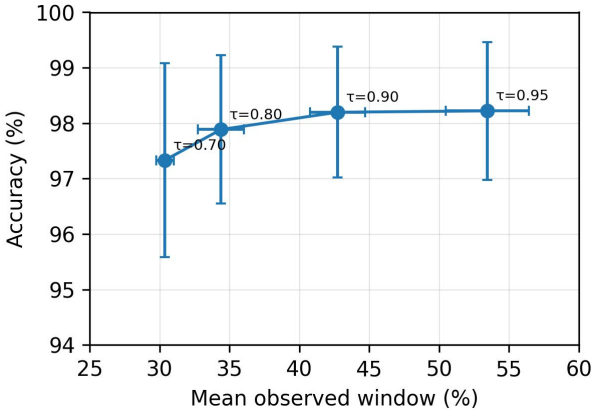

Figure 4 Accuracy–Observation Operating Points for Confidence-Gated Inference
Note: Both axes show mean±SD across seeds.

The adaptive accuracy can slightly exceed the average static full-window accuracy because the policy selects among predictions at multiple prefixes for each sample; a confident early prediction can be correct even when the later prediction changes. This should not be read as future evidence being harmful in general. It is an empirical property of the learned checkpoints and the confidence rule. Calibration methods or a validation-selected threshold would be needed before deploying the confidence value as a risk guarantee [27].

6.5 Class-Level Error Structure

The representative seed-13 confusion matrices show that errors at 25% observation are distributed across all three locomotion modes rather than dominated by one failed class. Row-normalized recalls are 96.6% for level walking, 96.0% for upstairs, and 96.4% for downstairs. At the complete window, the same seed reaches near-perfect recalls, although the cross-seed standard deviation in Table 2 shows that this seed is optimistic and should not be treated as the expected deployment accuracy, see Figure 5.

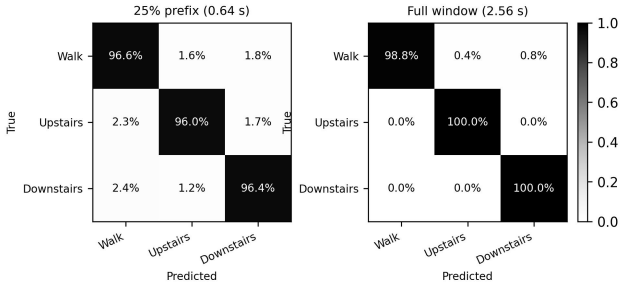

Figure 5 Row-Normalized Confusion Matrices for PrefixKD-CNN, Seed 13

7 DISCUSSION AND LIMITATIONS

7.1 What the Results Support

The experiments support three bounded conclusions. First, useful locomotion recognition is available well before the full UCI window: the proposed model exceeds 94% macro-F1 after 0.64 s. Second, training the same compact network across prefixes is essential to make one model usable at multiple checkpoints, and self-distillation can provide a small no-overhead regularization gain. Third, confidence gating exposes a tunable operating curve rather than hiding decision delay inside a single full-window score. Together, these properties are more relevant to an edge application than maximizing complete-window accuracy alone.

7.2 Dataset Scope

The UCI HAR benchmark is small by modern standards, contains a fixed waist-mounted smartphone placement, and provides pre-segmented windows with 50% overlap. The supplied body-acceleration signals are already separated from gravity by the dataset preprocessing. Consequently, the experiment does not establish robustness to arbitrary device placement, sensor drift, unseen locomotion transitions, or raw unfiltered streams. More importantly, because windows correspond to ongoing activities, this study addresses early mode inference—not anticipation before movement onset. A true intent-forecasting evaluation would require transition-centered labels and a prediction horizon preceding the event.

7.3 Statistical and Deployment Scope

Only three optimization seeds were executed. This is sufficient to reveal meaningful variance and prevent reporting a single favorable run, but insufficient for a well-powered significance test of the 0.20-point same-backbone gain. The server-class CPU benchmark establishes relative operator behavior in the supplied software stack; it does not measure microcontroller latency, energy, cache effects, peak activation memory, or integer quantization. The current decision loop also recomputes the CNN at each checkpoint. A production implementation should cache convolutional state or schedule only one inference when a policy determines that the next checkpoint is needed.

7.4 Future Work

A stronger follow-up should evaluate cross-dataset and cross-placement generalization, collect transition-centered sequences for genuine pre-onset intent prediction, calibrate confidence on a held-out subject group, and measure int8 latency and peak memory on representative microcontrollers. The training objective can also be extended with temporal consistency across adjacent prefixes or uncertainty-aware distillation. Such additions should be accepted only when they improve measured accuracy–earliness or device metrics, rather than architectural novelty in isolation.

7.5 Ethical and Reproducibility Considerations

No new human-subject data were collected. The study uses the public, anonymized UCI repository and reports only aggregate activity labels and model outputs. The available benchmark does not provide the demographic detail needed for subgroup fairness analysis, so the reported averages should not be assumed to hold uniformly across ages, body types, mobility impairments, or device-use patterns. Any safety-critical use would require representative prospective validation and a conservative abstention or fallback mechanism.

Reproducibility artifacts preserve the exact three seeds, training histories, selected epochs, test predictions, confidence values, environment versions, and model checkpoints. The result tables are generated directly from these stored arrays rather than transcribed manually. The archive also retains the exploratory GMS models and their slower measured latency, ensuring that the negative architecture result remains auditable rather than being removed after model selection.

8 CONCLUSION

This paper presented PrefixKD-Edge, a reproducible framework for early locomotion-intent inference using a compact temporal CNN, full-window-to-prefix self-distillation, and confidence-gated decision timing. On the official subject-disjoint UCI HAR split, the 7.9k-parameter model reaches 94.30±1.76% macro-F1 after 0.64 s and 98.13±1.33% on the full 2.56-s window. A threshold of 0.90 yields 98.20±1.18% accuracy at 1.09 s mean observation, while the backbone executes in approximately 0.033 ms per invocation on the measured CPU. The self-distillation gain over the same CNN is small and is reported without a significance claim. The broader result is that an honest edge design should optimize and measure the complete decision pipeline—sensor front-end, prefix training, stopping policy, and runtime operators—rather than equating a complicated lightweight architecture with practical efficiency.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

DATA AND CODE AVAILABILITY

The accompanying archive contains the complete PyTorch experiment script, seed-level predictions, aggregate CSV files, model checkpoints, environment metadata, and figure-generation code. The raw UCI data are not redistributed in the archive; a downloader and exact expected file layout are provided. Running the documented command reproduces all model training, test metrics, complexity summaries, latency measurements, adaptive exits, and confusion matrices.

REFERENCES

[1] Lara O D, Labrador M A. A survey on human activity recognition using wearable sensors. IEEE Communications Surveys & Tutorials, 2013, 15(3): 1192-1209. DOI: 10.1109/SURV.2012.110112.00192.

[2] Bulling A, Blanke U, Schiele B. A tutorial on human activity recognition using body-worn inertial sensors. ACM Computing Surveys, 2014, 46(3): Art. 33. DOI: 10.1145/2499621.
[3] Yang J, Nguyen M N, San P P, et al. Deep convolutional neural networks on multichannel time series for human activity recognition. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2015: 3995-4001.
[4] Hammerla N Y, Halloran S, Plötz T. Deep, convolutional, and recurrent models for human activity recognition using wearables. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2016: 1533-1540.
[5] Ordóñez F J, Roggen D. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors, 2016, 16(1): Art. 115. DOI: 10.3390/s16010115.
[6] Ronao Y S, Cho S B. Human activity recognition with smartphone sensors using deep learning neural networks. Expert Systems with Applications, 2016, 59: 235-244. DOI: 10.1016/j.eswa.2016.04.032.
[7] Murad A, Pyun J-Y. Deep recurrent neural networks for human activity recognition. Sensors, 2017, 17(11): Art. 2556. DOI: 10.3390/s17112556.
[8] Ignatov A. Real-time human activity recognition from accelerometer data using convolutional neural networks. Applied Soft Computing, 2018, 62: 915-922. DOI: 10.1016/j.asoc.2017.09.027.
[9] Yao S, Hu S, Zhao Y, et al. DeepSense: A unified deep learning framework for time-series mobile sensing data processing. In: Proceedings of the International Conference on World Wide Web (WWW), 2017: 351-360. DOI: 10.1145/3038912.3052577.
[10] Xing Z, Pei J, Yu P S. Early classification on time series. Knowledge and Information Systems, 2012, 31(1): 105-127. DOI: 10.1007/s10115-011-0400-x.
[11] Mori U, Mendiburu A, Dasgupta S, et al. Early classification of time series by simultaneously optimizing the accuracy and earliness. IEEE Transactions on Neural Networks and Learning Systems, 2018, 29(10): 4569-4578. DOI: 10.1109/TNNLS.2017.2764939.
[12] Schäfer P, Leser U. TEASER: Early and accurate time series classification. Data Mining and Knowledge Discovery, 2020, 34: 1336-1362. DOI: 10.1007/s10618-020-00690-z.
[13] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735-1780. DOI: 10.1162/neco.1997.9.8.1735.
[14] Cho K. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014: 1724-1734. DOI: 10.3115/v1/D14-1179.
[15] Lea C, Flynn M D, Vidal R, et al. Temporal convolutional networks for action segmentation and detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017: 156-165.
[16] Bai S, Kolter J Z, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271, 2018.
[17] van den Oord A. WaveNet: A generative model for raw audio. arXiv:1609.03499, 2016.
[18] Howard A G. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861, 2017.
[19] Sandler M, Howard A, Zhu M, et al. MobileNetV2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018: 4510-4520.
[20] Jacob B. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018: 2704-2713.
[21] Lin J, Chen W-M, Lin Y, et al. MCUNet: Tiny deep learning on IoT devices. In: Advances in Neural Information Processing Systems, 2020, 33: 11711-11722.
[22] David R. TensorFlow Lite Micro: Embedded machine learning for TinyML systems. In: Proceedings of Machine Learning Systems, 2021, 3: 800-811.
[23] Banbury C R. MicroNets: Neural network architectures for deploying TinyML applications on commodity microcontrollers. In: Proceedings of Machine Learning Systems, 2021, 3.
[24] Banbury C. Benchmarking TinyML systems: Challenges and direction. arXiv:2003.04821, 2020.
[25] Teerapittayanon S, McDanel B, Kung H T. BranchyNet: Fast inference via early exiting from deep neural networks. In: Proceedings of the International Conference on Pattern Recognition (ICPR), 2016: 2464-2469.
[26] Geifman Y, El-Yaniv R. Selective classification for deep neural networks. arXiv:1705.08500, 2017.
[27] Guo C, Pleiss G, Sun Y, et al. On calibration of modern neural networks. In: Proceedings of the International Conference on Machine Learning (ICML), PMLR, 2017, 70: 1321-1330.
[28] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv:1503.02531, 2015.
[29] Zhang L, Song J, Gao A, et al. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019: 3713-3722.
[30] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the International Conference on Machine Learning (ICML), PMLR, 2015, 37: 448-456.
[31] Anguita D, Ghio A, Oneto L, et al. A public domain dataset for human activity recognition using smartphones. In: Proceedings of the European Symposium on Artificial Neural Networks (ESANN), 2013: 437-442.

[32] Anguita D, Ghio A, Oneto L, et al. Human Activity Recognition Using Smartphones. UCI Machine Learning Repository, 2013. DOI: 10.24432/C54S4K.
[33] Loshchilov I, Hutter F. Decoupled weight decay regularization. In: Proceedings of the International Conference on Learning Representations (ICLR), 2019.
[34] Loshchilov I, Hutter F. SGDR: Stochastic gradient descent with warm restarts. In: Proceedings of the International Conference on Learning Representations (ICLR), 2017.
[35] Paszke A. PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, 2019, 32: 8024-8035.