CHALMERS
UNIVERSITY OF TECHNOLOGY

An Edge AI Test Bench for Unsupervised Anomaly Detection

Master's thesis in Mechanical Engineering

Wentao Chen Chengyu Zhu

Department of Mechanical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026

Master's thesis 2026

An Edge AI Test Bench for Unsupervised Anomaly Detection

Wentao Chen
Chengyu Zhu

Department of Mechanical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

An Edge AI Test Bench for Unsupervised Anomaly Detection
Wentao Chen
Chengyu Zhu

Supervisor: Silvan Marti, Mechanical Engineering
Examiner: Björn Johansson, Mechanical Engineering

Master's Thesis 2026
Department of Mechanical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

An Edge AI Test Bench for Unsupervised Anomaly Detection
Wentao Chen
Chengyu Zhu
Department of Mechanical Engineering
Chalmers University of Technology

Abstract

Condition monitoring of rotating machinery requires time-series representations that are compact enough to run on edge hardware, stable enough to generalize across production runs, and informative enough to support downstream anomaly detection without labeled fault examples. This thesis addresses all three requirements on a purpose-built, low-cost CNC-inspired test bench equipped with a brushless DC spindle and three stepper feed axes, instrumented for synchronized current, vibration, and speed sensing at approximately 91 Hz. A three-tier edge platform—ESP32-P4 acquisition node, Raspberry Pi 3 gateway, and Raspberry Pi 5 inference node—acquires a dataset of 94 636 samples across ten labeled operating cycles.
Four encoders spanning a wide capacity range—per-channel summary statistics, FFT amplitude bins, the self-supervised TS2Vec encoder, and three sizes of the pre-trained MOMENT transformer—are evaluated on 20 public UCR and UEA datasets as a cross-benchmark reference and on the CNC bench as the target domain, using four axes: a supervised linear probe, unsupervised clustering, six mode-aware geometry metrics, and a CPU edge-deployment benchmark.
The main findings are as follows. First, encoder rankings are dataset-dependent: TS2Vec leads on cross-benchmark accuracy but is outperformed on CNC by both MOMENT-large (0.850 accuracy) and the parameter-free Summary baseline (0.844), a reversal explained by the small CNC training set and the high mode-discriminability of the raw sensor channels. Second, geometric stability and label-aware accuracy rank encoders differently: MOMENT's embedding space is roughly an order of magnitude more stable across production runs than Summary's, making it the better foundation for run-disjoint anomaly detection despite similar classification scores. Third, post-training INT8 quantization collapses MOMENT's accuracy to chance when applied naively; restricting INT8 to the FFN linears preserves FP32 accuracy at all three model sizes with a 1.33–1.82× disk reduction and 1.25–1.55× latency reduction, and the small and base variants run comfortably within the per-window budget on the reference CPU.

Keywords: edge AI, condition monitoring, time-series representation learning, unsupervised anomaly detection, MOMENT, TS2Vec, model quantization

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Silvan Marti, for his invaluable guidance, continuous encouragement, and immense knowledge throughout my research. His insightful advice and rigorous academic attitude have been instrumental in the completion of this thesis. I would also like to extend my sincere thanks to my examiner, Professor Björn Johansson, for his constructive suggestions during my studies.
This work was supported by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), which provided the essential computational resources.
Last but not least, I would like to express my heartfelt gratitude to my respective families and friends for their unconditional love, patience, and emotional support over the years. Thank you to all the opponents and attendees for taking the time to review this thesis and attend my defense.

Chengyu Zhu, Gothenburg, June 2026

Acknowledgements

This thesis would not have been possible without the support and encouragement of many people, to whom I owe a great deal of gratitude.

I am profoundly thankful to my supervisor, Silvan Marti, whose patient mentorship, keen technical insight, and reliable guidance shaped both this project and my development as a researcher. Beyond pointing me in the right direction, he ensured I had the hardware and technical resources needed to bring the work to life — something I will not forget.

My sincere appreciation also goes to my examiner, Professor Björn Johansson, who took the time to engage carefully with this work at both the mid-term presentation and the final defense. The perspectives and suggestions he raised pushed me to think more rigorously and present my findings more clearly.

This study was supported by computational resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS).

Several people made the day-to-day experience of this project far more enjoyable and manageable. My partner Toria Sun offered encouragement at every turn and kept my spirits up through the harder moments. Weishuo Sun stepped in generously when the 3D printing demanded more than one pair of hands, and his contribution to the physical build of the test bench was genuinely important. Siyuan Chen offered timely and thoughtful advice at key points in the project — I am grateful for his willingness to share his expertise.

I would also like to thank the opponents and everyone who attended the defense for their active participation and the constructive questions they raised. Their engagement with the work meant a great deal to me.

To my friends — Yiran Duan, Xuqi Fu, Yiming Wang, Wenbo Tan and many other fellow master's students — thank you for the shared stress, the shared laughs, and everything in between.

Last of all, I thank my parents. Their belief in me has never wavered, and their support — given freely and without condition — is the reason I was able to see this through.

Wentao Chen, Gothenburg, June 2026

Contents

1

Introduction

1.1 Motivation

Modern manufacturing depends on large populations of electric motors and motion systems that drive spindles, feed axes, pumps and conveyors. When one of these components degrades unexpectedly, the consequence is rarely local: an unplanned stop can idle an entire production line, and the resulting downtime, scrap and emergency maintenance translate directly into economic loss. Continuous condition monitoring—observing a machine's behaviour through sensors in order to detect incipient faults—has therefore become a standard part of industrial operation rather than an optional add-on, and is supported by a mature body of international standards and decades of deployment experience [1, 2].

The conventional way to act on this sensor data is to stream it to a central server or to the cloud for analysis. At scale, however, this architecture is constrained by communication latency, network dependence, and concerns about transmitting operational data off-site. These pressures motivate *edge* processing, in which analysis is performed on compute resources placed close to the machine, so that decisions can be made locally and only compact results need to leave the device [3]. Running modern machine-learning models under the memory, compute and latency budgets of such edge hardware is the central practical tension this thesis addresses.

Two further difficulties make the problem harder in realistic settings. First, faults are rare: a reliable machine produces very few labeled failure examples, so methods that require large, balanced, human-labeled fault datasets are impractical [4]. Second, the signals are heterogeneous and operating-condition dependent: a vibration or current pattern that is normal in one operating mode may look anomalous in another, which complicates any monitoring scheme built on a single steady-state assumption. This thesis responds to both points with an *unsupervised* approach—learning a representation of normal behaviour from unlabeled data—evaluated on a purpose-built physical test bench that reproduces these conditions in a controlled way.

1.2 Related Work

Condition monitoring is a mature, standardized practice. Machine condition monitoring is institutionalized to the point of being governed by a hierarchy of international standards, with ISO 17359 providing general guidelines whose stated

scope spans essentially all machine types [1]. Industrial practitioners report more than three decades of condition-based maintenance deployment, with documented gains in reliability, availability and operating cost [2]. On the technical side, the field has long relied on a small set of complementary sensing modalities: vibration analysis remains the most widely used technique, frequently combined with spectral processing to expose fault signatures, alongside current and speed measurements [5, 6, 7]. The test bench in this thesis captures exactly this standard triad—spindle current, structural vibration, and rotational speed.

Classical monitoring assumes steady conditions that real operation violates. A recurring assumption in the classical literature is that measurements are taken under stable, predetermined operating conditions; the same standards that codify condition monitoring also caution that changing operating conditions make readings difficult to compare and to trend [1, 2]. Statistical process-control methods inherit related limitations, performing poorly when process variables are nonlinear and cross-correlated rather than independent and stationary [8]. A monitoring platform that deliberately cycles through distinct operating modes—idle, spindle-only, single-axis, and combined motion—therefore exercises precisely the regime where the steady-state assumption breaks down, and where a mode-aware representation is needed.

Label scarcity makes supervised fault detection impractical. The structural rarity of faults is well documented. Reliable machines provide few opportunities to observe anomalies, so the collection of labeled failure data is necessarily limited [4]. Surveys of Industry 4.0 benchmarks find that very few publicly available datasets satisfy the requirements for supervised failure prediction [9], and real operational datasets are extremely imbalanced—for instance, only a handful of catastrophic failures across many months of continuous operation [10]. These observations justify training exclusively on anomaly-free data and using the geometric properties of the resulting embedding space as proxies for downstream anomaly-detection viability, which is the evaluation strategy adopted here. A labeled anomaly campaign on the bench is the natural next step once a representation is selected.

Learned representations are promising, but their value over simple baselines is an open question. Deep learning offers a principled response to label scarcity: a large family of time-series anomaly-detection methods learns a model of normal behaviour and flags deviations from it, without requiring labeled faults [11]. Self-supervised representation learners such as TS2Vec [12] and pre-trained time-series foundation models such as MOMENT [13] are recent, prominent examples. However, recent work cautions that complex, expensive models do not reliably outperform far simpler alternatives on time-series tasks [14]. Whether a learned representation actually beats a strong, low-cost baseline on a given dataset is therefore an empirical question—one that this thesis addresses directly by benchmarking learned encoders against zero-parameter baselines on the same data.

1.3 Purpose and Goal

The purpose of this thesis is to design and build a controlled edge test bench for unsupervised condition monitoring, and to use it as a platform for evaluating compact time-series representations under realistic hardware constraints. To contextualize the CNC results, the same encoders are first evaluated on 20 public UCR and UEA time-series datasets as a cross-benchmark reference, establishing how encoder rankings on standard benchmarks relate to their rankings on the in-domain target. Rather than reporting a single accuracy number, the work assesses each candidate representation along four complementary evaluation dimensions, which together form the methodological core of the thesis:

1. a **supervised probe**, in which a simple classifier is trained on the frozen representation to measure how linearly separable the operating modes are;
2. **unsupervised clustering**, which tests whether the representation organises the data into coherent operating-mode groups without using labels;
3. **mode-aware geometry**, a set of label-light metrics that quantify how cleanly the representation separates and stabilizes operating modes; and
4. an **edge-deployment benchmark**, which measures inference latency and model footprint on the target Raspberry Pi inference node.

Relying on several views rather than one is a deliberate choice: recent analyses show that single-metric evaluation on time-series anomaly benchmarks can be highly misleading [15], and that window and protocol design can matter as much as model architecture [16]. Stating these four dimensions up front frames the entire evaluation methodology as a central contribution of the thesis rather than an afterthought.

1.4 Contributions

The main contributions of this thesis are:

- **A controlled, low-cost edge test bench.** A 3D-printed, CNC-inspired bench combining a brushless DC spindle and three stepper feed axes, instrumented for synchronized multimodal sensing and driven through a programmable sequence of labeled operating modes, providing a reproducible source of ground-truth-labeled data for unsupervised monitoring research.
- **A three-tier edge acquisition and inference platform.** A distributed architecture in which an ESP32-P4 node performs real-time, deterministic acquisition and control, a Raspberry Pi gateway forwards the data over Ethernet, and a Raspberry Pi 5 node runs the representation encoders and downstream monitoring models.
- **A multi-view evaluation of compact representations across domains.** A comparative study of self-supervised and pre-trained time-series encoders against zero-parameter baselines, evaluated on 20 public UCR and UEA datasets as a cross-benchmark reference and on the CNC bench as the target domain,

assessed along the four evaluation dimensions above including their feasibility under edge deployment constraints.

The remainder of this thesis is organized as follows. Chapter 2 provides the technical background on condition monitoring, time-series representation learning, and evaluation. Chapter 3 describes the design of the test bench, the embedded acquisition and control firmware, and the representation-learning and anomaly-detection pipeline. Chapter 4 reports the experimental results, Chapter 5 discusses them, and Chapter 6 concludes.

2

Technical Background

This chapter introduces the concepts required to follow the remainder of the thesis. It first reviews the fundamentals of machine condition monitoring and the sensing modalities on which it relies (Section 2.1), then surveys the time-series representation methods that are relevant to the work, together with the simple baselines against which such methods are compared (Section 2.3), and finally discusses why evaluating time-series anomaly detection is itself a non-trivial problem (Section 2.5). The chapter describes existing knowledge only; the specific design choices made in this thesis are presented in Chapter 3.

2.1 Condition Monitoring Fundamentals

Condition monitoring (CM) is the practice of observing one or more physical signals from a machine in order to assess its health and to detect incipient faults before they lead to failure. It underpins condition-based maintenance, in which maintenance actions are scheduled according to the observed state of the equipment rather than at fixed intervals. The discipline is mature and standardized: ISO 17359 sets out general guidelines for establishing a condition-monitoring program and is positioned as applicable across machine types, sitting at the top of a family of more specific standards [1]. Industrial reviews report several decades of condition-based maintenance practice, with documented benefits in reliability, availability and operating cost [2].

Sensing modalities. Classical condition monitoring relies on a small set of complementary sensing modalities, each sensitive to a different aspect of machine behaviour. *Vibration* is the most widely used, since mechanical faults—imbalance, misalignment, bearing and gear defects—excite characteristic structural vibrations; analysis is typically performed in the frequency domain, where fault signatures appear at predictable frequencies, and envelope (demodulation) analysis is the established benchmark technique for rolling-element bearing diagnosis [5, 6]. *Motor current* provides an electrical view of the same machine: load changes and certain mechanical faults modulate the current drawn by a drive, so current monitoring complements vibration without requiring access to the rotating structure. *Rotational speed*, often obtained from a tachometer or encoder, provides the kinematic reference needed to interpret the other signals, since many fault frequencies scale with shaft speed. Reviews of deployed systems note that, although vibration is

dominant in practice, it carries operational costs—correct sensor placement requires expertise and early-stage faults can be difficult to detect—which motivates combining several modalities [7].

The steady-state assumption and its limits. A recurring premise of classical CM is that measurements are most meaningful when taken under stable, repeatable operating conditions; the standards themselves recommend monitoring at predetermined operating states and note that varying conditions make readings harder to compare and to trend [1, 2]. Many machines, however, operate across distinct regimes—idle, partial load, full load, and transitions between them—so a signal pattern that is normal in one regime can resemble a fault in another. This operating-condition dependence is one reason data-driven methods have gained traction in condition monitoring, as they can in principle learn the relationship between operating regime and signal behaviour directly from data [11].

2.2 Unsupervised Anomaly Detection

An anomaly is an observation that deviates markedly from the expected behaviour of a system. In condition monitoring the anomalies of interest are faults, which are by their nature rare: a reliable machine spends almost all of its operating life in a healthy state, so labeled fault examples are scarce and the full set of possible failure modes cannot be enumerated in advance. This makes supervised classification—which requires many labeled examples per fault class—impractical for realistic deployments.

The dominant alternative is *unsupervised* anomaly detection, in which a model of normal behaviour is built from anomaly-free data and any sufficiently large deviation from that model is flagged. A recent survey of deep learning for time-series anomaly detection identifies several broad families that share this normal-only training principle [11]. *Reconstruction-based* methods, such as autoencoders, learn to compress and reconstruct normal data; an input that the model cannot reconstruct faithfully yields a large error and is treated as anomalous. *Prediction-based* methods learn to forecast the next value from past context and use the prediction error as the anomaly signal. *Distance- and density-based* methods operate in a representation space and flag points that lie far from, or in low-density regions relative to, the bulk of the training data; isolation forests and nearest-neighbor distances are common examples.

A practical consequence common to all families is that the quality of the underlying representation strongly influences detection performance: if the embedding collapses unrelated modes onto the same region, or if the centroid of a given mode drifts across production runs, a distance-based detector built on that representation will produce unreliable scores. This links the anomaly-detection problem directly to the representation-learning and geometric-stability questions addressed in this thesis.

2.3 Time-Series Representation Learning

The signals produced by condition monitoring are multivariate time series. A central problem is therefore how to turn a window of raw, multi-channel signal into a compact, fixed-length vector—a *representation* or *embedding*—that preserves the information relevant to downstream tasks such as clustering or anomaly detection. This subsection summarizes the families of methods relevant to this thesis. Their use in this work, including configuration, is described in Chapter 3.

Hand-crafted and spectral features. The classical approach extracts features designed by domain knowledge. Summary statistics (means, variances, and higher-order moments per channel) capture the amplitude distribution of a signal, while a discrete Fourier transform exposes its frequency content. Because rotating machinery produces energy at characteristic frequencies, spectral features are not arbitrary descriptors but the same quantities that underpin classical vibration diagnosis [5]. Such features require no training and are computationally cheap, which makes them natural reference points: data-centric analyses argue that, when data is clean and well-curated, careful feature extraction can capture much of the discriminative information that more complex models seek [17].

Self-supervised representation learning. A more recent approach learns representations directly from unlabeled time series. Contrastive methods train an encoder so that different views or sub-segments of the same series map to nearby points while unrelated segments map apart. TS2Vec is a representative example: it trains a temporal convolutional encoder with a hierarchical contrastive objective that encourages representations to be consistent across multiple temporal scales, and produces a single fixed-length vector per window by aggregating across time [12]. Because such encoders are trained without labels, they fit naturally into the unsupervised setting where labeled faults are scarce.

Pre-trained time-series foundation models. A further development is the time-series *foundation model*: a large model pre-trained once on a broad corpus of time series and then reused across tasks and datasets, by analogy with foundation models in language and vision. MOMENT is a representative open family of such models, pre-trained on a large collection of public time series using a masked reconstruction objective in which randomly masked sub-series (patches) must be reconstructed from their context; the pre-trained encoder can then produce embeddings for new data, in principle without further training [13]. The appeal of this paradigm is zero- or few-shot reuse, but its benefits on a given target domain are not guaranteed.

A caution on complexity. The assumption that larger, pre-trained models are necessarily better for time series has been challenged. Ablation studies on language-model-based forecasters report that removing or replacing the heavy pre-trained component often leaves accuracy unchanged or even improves it, at a fraction of the computational cost [14], and related work finds that such models still struggle to

reason about time series in a zero-shot manner [18]. The methodological consequence is that a learned or pre-trained representation should be compared against simple, low-cost baselines on the same data, since superiority cannot be assumed a priori.

2.4 Edge AI and Embedded Inference

The representation methods described above are conventionally run on servers with abundant compute. In condition monitoring there are strong reasons to move the computation closer to the machine: transmitting high-rate raw sensor data to a central server or to the cloud incurs communication latency and bandwidth cost, creates a dependence on network availability, and raises concerns about exposing operational data. *Edge computing* addresses this by performing analysis on resources located near the data source, so that decisions are made locally and only compact results are transmitted [19].

Pushing machine learning to the smallest embedded devices is the goal of *TinyML*, which seeks to run inference—and increasingly even training—on microcontrollers with only kilobytes to a few megabytes of memory and no dedicated accelerator [20]. Frameworks have been developed to deploy and, in some cases, train neural networks under these constraints on resource-limited hardware [21]. Because off-the-shelf deep models rarely fit such budgets directly, *model compression* is central to edge deployment. Two widely used techniques are *pruning*, which removes weights or structures that contribute little to the output, and *quantization*, which represents weights and activations at lower numerical precision—for example using 8-bit integers (INT8) instead of 32-bit floating point (FP32)—reducing both memory footprint and inference time, at the risk of accuracy loss if applied without care. Post-training quantization applies these reductions to a pre-trained checkpoint without retraining; its effect on model quality depends strongly on the numerical sensitivity of each layer, and different sub-modules of the same network can behave very differently under the same quantization scheme.

2.5 Evaluation Challenges in Time-Series Anomaly Detection

Evaluating time-series anomaly detection is itself difficult, and naive evaluation can be actively misleading. A widely cited analysis argues that several popular anomaly-detection benchmarks are flawed—through mislabeled points, trivially detectable anomalies, and other defects—to the extent that reported progress on them may be partly illusory, and that a near-perfect score on such a benchmark should invite suspicion rather than confidence [15]. Two consequences follow for any careful evaluation. First, because ground-truth labels may themselves be unreliable, it is valuable to complement label-based scores with label-free measures of representation quality, such as internal clustering criteria that assess how compact and well-separated groups are without reference to labels. Second, results can depend as strongly on experimental-protocol choices—window length, overlap, and how data is split into

training and evaluation portions—as on the model itself, so these choices must be reported and justified [16]. These observations motivate evaluating a representation from several complementary angles rather than trusting any single metric.

Label-aware measures. When ground-truth class labels are available, two families of measures are useful. *Linear-probe accuracy* trains a simple classifier (here a support-vector machine) on a frozen representation and reports held-out accuracy; it asks whether a linear boundary separates the classes and gives a label-aware upper bound on what a non-parametric detector could achieve. Because anomalies are rare and the positive class is a small minority, plain accuracy is dominated by the majority-normal class and is an unreliable summary. The *area under the precision–recall curve* (AUPRC) avoids this by summarizing the precision–recall trade-off across all decision thresholds, giving equal weight to every operating point; a random classifier on a dataset with positive rate ρ attains AUPRC $\approx \rho$, so AUPRC provides a meaningful baseline-free reference under class imbalance. The *adjusted rand index* (ARI) measures the agreement between an unsupervised clustering of the data and the true labels, corrected for chance agreement; it ranges from -1 (worse than chance) through 0 (chance) to 1 (perfect).

Label-free (internal) clustering measures. When labels are absent or unreliable, internal indices evaluate the geometry of a partition directly. The *silhouette coefficient* compares each point's mean distance to its own cluster against its mean distance to the nearest other cluster; values close to 1 indicate that the point is well-matched to its cluster and far from neighbors. The *Davies–Bouldin index* computes for each cluster the worst-case ratio of within-cluster scatter to between-cluster distance; lower values indicate more separated clusters. The *Calinski–Harabasz index* (also known as the variance ratio criterion) is the ratio of the between-cluster dispersion to the within-cluster dispersion summed over clusters; higher values indicate more compact, well-separated clusters. These three indices assess different aspects of cluster geometry and typically agree in direction but not necessarily in rank.

3

Method

The proposed system is organized as a three-tier edge platform that separates real-time signal acquisition, network gatewaying, and machine-learning inference. At the lowest tier, an **ESP32-P4** microcontroller drives the mechatronic test bench and samples all physical quantities at a fixed rate. The acquired data stream is transported over a $100\,\mathrm{Mbit\,s^{-1}}$ Ethernet network through a **Raspberry Pi 3** gateway node to a **Raspberry Pi 5** inference node, on which the representation-learning and anomaly-detection models are executed. This separation keeps the deterministic, timing-critical control and sampling loop physically isolated from the comparatively heavy and non-deterministic inference workload, while the switched Ethernet fabric provides low-latency communication between the nodes. The remainder of this chapter details the physical bench and its instrumentation (Section 3.1), the embedded acquisition and control firmware (Section 3.2), the CNC dataset construction (Section 3.3), the four encoders under evaluation (Section 3.4), and the evaluation protocols along the four axes introduced in Section 1.3 (Section 3.5).

3.1 Mechatronic Test Bench

To obtain repeatable multimodal data under controlled operating conditions, a simplified mechatronic test bench was designed and built to emulate the principal motion components of a three-axis computer numerical control (CNC) machine. The bench reproduces the four main actuated degrees of freedom of such a machine—a rotating spindle and three orthogonal linear feed axes—using low-cost, widely available components. This abstraction retains the characteristic electrical and vibration signatures of each operating mode while remaining inexpensive, safe, and fully programmable, which makes it suitable as a controlled platform for generating labeled monitoring data.

3.1.1 Design Requirements and Component Selection

The bench was guided by a small set of requirements that follow from its intended role. It had to (i) reproduce the principal motion components of a three-axis CNC machine—a rotating spindle and three orthogonal feed axes; (ii) capture the standard condition-monitoring sensing modalities, namely current, vibration and speed, so that each operating mode produces a characteristic multimodal signature; (iii) be programmable, so that labeled operating sequences and controlled perturbations

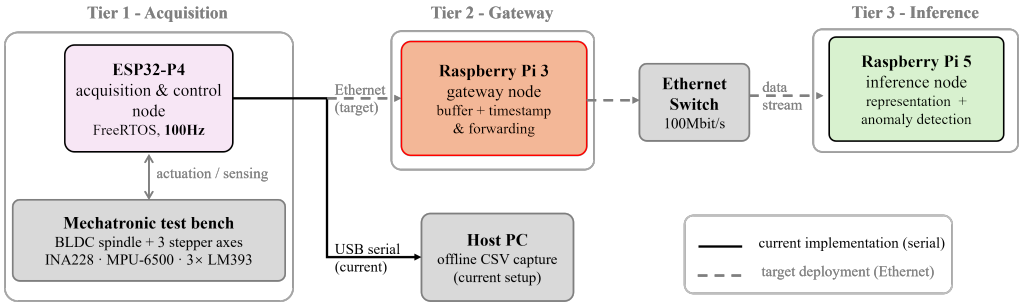

Figure 3.1: Three-tier edge architecture: the ESP32-P4 acquisition node connects through an Ethernet switch to the Raspberry Pi 3 gateway and the Raspberry Pi 5 inference node.

can be generated repeatably; and (iv) remain low-cost and reproducible, built from widely available components so that the platform can be rebuilt by others. Component availability was an explicit constraint: wherever possible, parts already present in the laboratory or obtainable cheaply off the shelf were used, in keeping with the goal of a low-cost, reproducible bench rather than a bespoke instrument. The choices below are stated together so that the rationale for the instrumentation is explicit rather than implied.

Actuators. The spindle is emulated by a brushless DC (BLDC) motor. A BLDC motor was preferred over a brushed or hobby-servo alternative because it exposes a frequency-generator (FG) tachometer output, which provides a speed measurement at no additional sensor cost, and because its current draw under PWM speed control gives a realistic spindle current signature. The three feed axes use 28BYJ-48 geared stepper motors driven by ULN2003 boards. These were chosen primarily for availability and cost: the 28BYJ-48 together with its matched ULN2003 driver is among the cheapest and most widely available stepper units, is straightforward to drive from 5 V logic, and—being a stepper—moves in discrete, countable steps, which makes commanded position trivial to track in firmware.

Position feedback. Although a stepper motor's commanded step count is known in firmware, an open-loop 28BYJ-48 has no position feedback and can *lose steps* under excessive load, abrupt acceleration, or mechanical obstruction, so the commanded count can silently diverge from the true shaft position. To observe this, each axis is fitted with an independent optical sensor—an LM393-based H2010 slotted photo-interrupter with a slotted disc—that produces one pulse per slot as the shaft turns. Comparing this measured pulse count against the commanded step count makes lost steps and stalls directly observable, which is valuable both for verifying normal motion and for detecting the kind of mechanical fault this thesis targets. A photo-interrupter was selected over a Hall-effect or rotary-encoder alternative because it is inexpensive, non-contact, and produces a clean digital pulse that can be counted directly on a GPIO interrupt without additional interface circuitry.

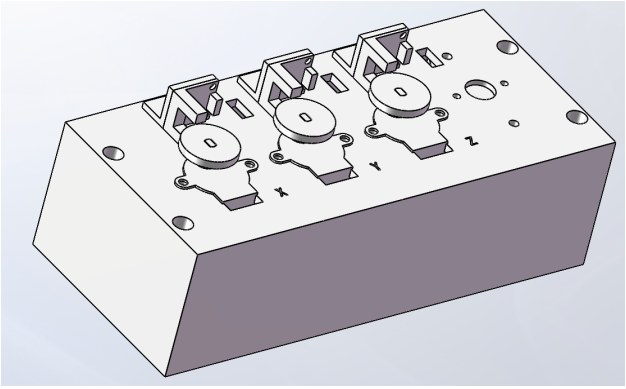

Figure 3.2: CAD model of the 3D-printed enclosure showing the spindle aperture (top), the three axis apertures with engraved index scales, and the sensor mounting positions.

3.1.2 Mechanical Structure

The mechanical frame was designed in CAD and fabricated by fused-deposition 3D printing as a single enclosure that houses all actuators and position sensors (Figure 3.2). The enclosure provides fixed mounting seats for the spindle motor and the three stepper motors, and integrates optical apertures so that each axis is monitored by its own photo-interrupter. The front face is engraved with the X, Y and Z axis labels and graduated index marks (positions 1–11) around each axis aperture, which serve as visual references during calibration and manual verification of motion.

The spindle is emulated by a 24 V brushless DC (BLDC) motor, while each of the three linear feed axes is emulated by a geared stepper motor. Mounting all actuators in a single rigid printed body deliberately couples their mechanical vibration into a common structure; the inertial sensor mounted on the frame therefore observes the superposition of all active sources, reproducing—at small scale—the vibration cross-coupling that complicates condition monitoring on real machine tools. The fabricated bench is shown in Figure 3.3.

3.1.3 Actuators

Spindle (BLDC motor). The spindle is driven by the brushless DC motor body of an NFP-GM37-BL3650 unit, with its GM37 reduction gearbox removed so that the bare BL3650 motor shaft acts directly as the spindle. It is powered from a dedicated external 24 V supply. The motor exposes a five-wire interface: a power pair (V^+/GND), a direction line, a 0 V to 5 V PWM speed-control line, and a frequency-generator (FG) tachometer output. The FG line produces six square-wave pulses per motor revolution and is used as a single-wire speed encoder, allowing the rotational speed to be recovered from the pulse frequency (Section 3.2.4). Because the gearbox has been removed, the FG output corresponds directly to the spindle shaft speed. In the experiments reported here the direction line is held in a fixed state, so the spindle rotates in a single direction throughout; speed is the only

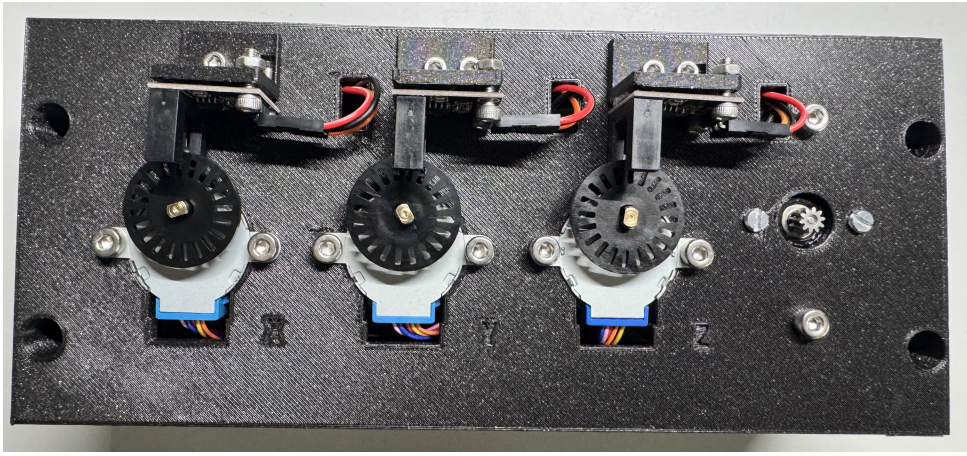

Figure 3.3: Photograph of the fabricated mechatronic test bench. The three 28BYJ-48 geared stepper motors (labeled X, Y, and Z) are visible in the lower section of the enclosure, each paired with an LM393/H2010 slotted optical photointerrupter for position feedback. The three servo-style ULN2003 driver boards are mounted above them. A small pinion gear (upper right) marks the BLDC spindle motor shaft. All components are housed within the single fused-deposition 3D-printed enclosure described in the text.

commanded spindle variable.

Linear feed axes (28BYJ-48 stepper motors). Each of the X, Y and Z feed axes is realized with a 28BYJ-48 unipolar geared stepper motor (nominal gear ratio 1:64), powered from a separate external 5 V supply and driven through a ULN2003 Darlington-array driver board. The motors are commanded in eight-step half-step mode. With 64 half-steps per internal motor revolution and a 1:64 gear reduction, one full revolution of the output shaft corresponds to $64 \times 64 = 4096$ commanded half-steps, which defines the relationship between commanded steps and physical output-shaft rotation. The manufacturer's nominal reduction ratio of 1:64 is used throughout; although the true ratio of the 28BYJ-48 is sometimes quoted as 1:63.684, the nominal value is sufficient for the relative, mode-discriminative analysis pursued here and is adopted for consistency. Driving the three axes from independent tasks allows arbitrary combinations of single-axis and simultaneous multi-axis motion to be programmed (Section 3.2.5).

3.1.4 Sensing

The bench is instrumented to capture three complementary modalities that are standard in machine condition monitoring: spindle current and speed, structural vibration, and per-axis position feedback.

- **Spindle current — INA228.** A Texas Instruments INA228 high-side current/power monitor measures the spindle supply current. It is connected in series with the 24 V spindle rail (its $\mathrm{Vin}^+/\mathrm{Vin}^-$ terminals bridge a shunt resistor), so it observes the true motor current independently of the logic supply. The INA228 communicates over $\mathrm{I}^2\mathrm{C}$ and is powered from the ESP32-P4 board

Table 3.1: Instrumentation summary.

Sensor	Device	Measured quantity	Interface
Current monitor	INA228	Spindle supply current (24 V rail)	I²C @ 400 kHz
IMU	MPU-6500	3-axis acceleration + 3-axis angular rate	I²C @ 400 kHz
Spindle tacho	Motor FG out	Spindle speed (6 pulses/rev)	GPIO ISR (pulse count)
Axis encoder (×3)	LM393/H2010	Per-axis slot count (20 slots/rev)	GPIO ISR (pulse count)

3.3 V rail.

- **Vibration — MPU-6500 IMU.** An InvenSense MPU-6500 six-axis inertial measurement unit, mounted on the frame, provides three-axis acceleration and three-axis angular rate. It shares the same I²C bus as the INA228 and captures the combined vibration signature of all active actuators.
- **Position feedback — LM393/H2010 photo-interrupters.** Each feed axis carries an LM393-based H2010 slotted optical photo-interrupter that produces one digital pulse per slot of a 20-slot encoder disc. One output-shaft revolution therefore yields 20 pulses, giving an independent, sensor-side measurement of axis motion that can be cross-checked against the commanded step count.

This combination yields a synchronized, multimodal description of machine state: an electrical channel (current), a kinematic channel (commanded steps, measured slots, spindle FG frequency) and an inertial channel (acceleration and gyroscope), all time-stamped on a common clock.

3.1.5 Electrical Design and Wiring

The electrical layout follows two principles: a single shared I²C bus for the two digital sensors, and strict separation of the actuator power rails from the logic supply, with all grounds tied to a single common reference. The INA228 and MPU-6500 are connected in parallel on one I²C bus, with the ESP32-P4 acting as bus master (SDA on GPIO7, SCL on GPIO8) at 400 kHz. Both devices, together with the photo-interrupters, are powered from the board 3.3 V rail.

The spindle and the stepper motors are driven from independent external supplies (24 V and 5 V respectively) rather than from the microcontroller, both to provide sufficient current and to limit electrical noise on the logic rail. The INA228 is inserted on the high side of the 24 V spindle rail. Crucially, the external 24 V supply ground, the 5 V stepper supply ground and the ESP32-P4 ground are bonded to a common ground, which is required both for valid current sensing and for the digital

Table 3.2: Signal and power connections (ESP32-P4 as the central node).

Subsystem	Connection	ESP32-P4 pin
I^2C bus	INA228 SDA, MPU-6500 SDA	GPIO7 (SDA)
I^2C bus	INA228 SCL, MPU-6500 SCL	GPIO8 (SCL)
Logic power	INA228 VIN, MPU-6500 VCC, LM393 VCC	3V3
Common ground	All sensor GND, ULN2003 GND, 24 V−, 5 V−	GND
Spindle current	Shunt across INA228 Vin+/Vin− on 24 V rail	(I^2C readout)
Spindle control	Direction (fixed), PWM speed, FG tacho	GPIO23 / GPIO54 / GPIO48
Stepper X driver	ULN2003 IN1–IN4	GPIO2 / 3 / 4 / 5
Stepper Y driver	ULN2003 IN1–IN4	GPIO6 / 20 / 21 / 22
Stepper Z driver	ULN2003 IN1–IN4	GPIO24 / 33 / 26 / 27
Stepper power	ULN2003 VCC (×3)	Board 5V
Photo-interrupters	LM393 OUT — $X/Y/Z$	GPIO36 / 32 / 25

control signals to share a reference. The wiring is summarized in Table 3.2.

Two practical characteristics of the spindle drive are noted here because they affect the interpretation of the measured signals (and are revisited in the discussion). First, the motor's PWM speed-control input is specified for a 0 V to 5 V signal, whereas the ESP32-P4 GPIO drives it at 3.3 V logic level without level shifting in the present setup; the duty-cycle-to-speed mapping is therefore not guaranteed to span the motor's full nominal range and may be non-linear near the top of the range. Second, with the gearbox removed the FG output reflects the spindle shaft speed directly, but it is taken from the motor's internal commutation and is therefore subject to electrical noise at high speed. Both points are treated as known measurement-system limitations rather than defects, as the monitoring task operates on the observed signals as they are produced.

3.2 Data Acquisition and Control Firmware

The ESP32-P4 firmware fulfils two roles simultaneously: it executes the programmed motion sequence that exercises the bench, and it samples all sensor channels at a fixed rate, emitting a fully labeled time-series record. The firmware is built on the FreeRTOS real-time kernel, which allows the time-critical sampling loop, the per-axis motion generation, and the high-level test sequence to run as independent, prioritized tasks with deterministic timing.

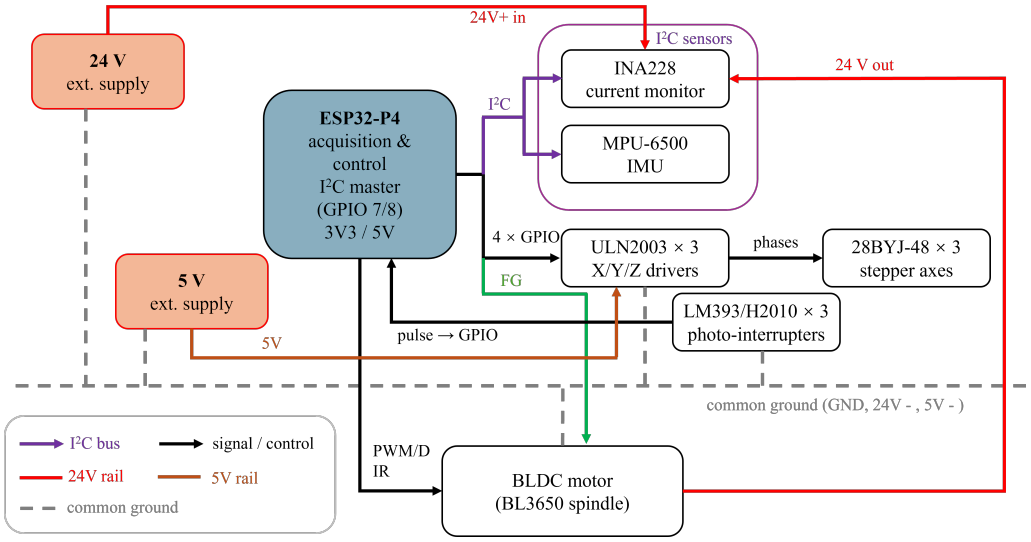

Figure 3.4: Wiring block diagram of the acquisition node, showing the shared I²C bus, the three ULN2003-driven stepper axes, the photo-interrupter inputs, and the isolated 24 V / 5 V power rails referenced to a common ground.

3.2.1 Task Architecture

The firmware is decomposed into five concurrent tasks (Figure 3.5). Three identical *stepper tasks* (one per axis) generate the half-step waveforms for the ULN2003 drivers; a single *data-logger task* performs periodic sampling and output; and the *main control task* implements the high-level state machine that sequences the experiment. The logger runs at the highest application priority so that its sampling period is protected from jitter introduced by the other tasks. Pulse-counting from the FG tachometer and the three photo-interrupters is handled entirely in hardware interrupt service routines (ISRs), decoupling event counting from the sampling period.

3.2.2 Motion Generation

Each stepper task owns the four control lines of one axis and walks an eight-entry half-step excitation table. When the axis is commanded to move, the task advances one half-step every 4 ms and increments a per-axis commanded-position counter; when the axis is idle, all four phase outputs are de-energized to avoid unnecessary heating and vibration. Because each axis is an independent task sharing only a boolean motion flag with the controller, single-axis motion and coordinated multi-axis motion are produced by simply asserting the corresponding flags. The 4 ms half-step interval sets the nominal feed rate of each axis and was chosen as a compromise between smooth motion and reliable operation of the geared 28BYJ-48 motors.

3.2.3 Spindle Speed Control

Spindle speed is set through the ESP32-P4 LEDC peripheral, configured for a 20 kHz PWM carrier with 10-bit duty resolution on the PWM line. The controller specifies

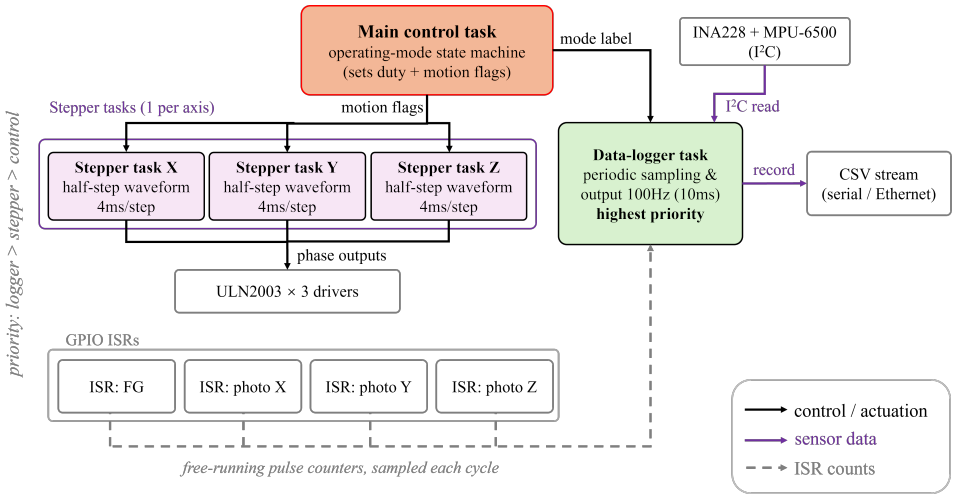

Figure 3.5: FreeRTOS task and interrupt architecture: three stepper tasks, one 100 Hz logger task, the control state machine, and four GPIO interrupt sources (FG and three photo-interrupters).

speed as a duty-cycle percentage, which is mapped linearly onto the 10-bit duty register; two operating levels are used in the experiments, a reduced level for spindle-only running and a high level for the combined feed mode. As noted in Section 3.1.5, the PWM line is driven at 3.3 V without level translation, so the commanded duty percentage is treated as the spindle command variable rather than as a calibrated speed.

3.2.4 Sensor Acquisition and Signal Conditioning

The logger task samples at a fixed rate of 100 Hz (a 10 ms period), enforced with a periodic FreeRTOS delay-until primitive so that successive samples are equally spaced regardless of the work done within each cycle. On every cycle the task reads the inertial unit and the current monitor over I^2C, derives the spindle speed from the accumulated FG pulses, snapshots the kinematic counters, and appends one record to the output stream.

- **Inertial data.** Three-axis acceleration and three-axis angular rate are read from the MPU-6500 in a single burst transaction and reported as raw signed 16-bit values, preserving the full sensor resolution for downstream scaling.
- **Spindle current.** Rather than relying on a programmed calibration register, the current is computed directly from the INA228 shunt-voltage register. The 20-bit signed shunt-voltage reading is sign-extended and scaled by the device's 312.5 nV least-significant-bit weight to obtain the shunt voltage, and the current is then recovered by Ohm's law from the known 0.015 Ω shunt resistance. This physical, register-level conversion makes the current channel independent of any device calibration step.
- **Spindle speed.** The FG interrupt increments a free-running pulse counter. Each cycle the logger forms the pulse increment since the previous cycle and multiplies by the sampling rate to obtain the instantaneous FG frequency

Table 3.3: One cycle of the operating-mode sequence.

Step	Operating mode	Duration	Active actuators
1	`MODE_IDLE`	10 s	None
2	`MODE_SPINDLE_ONLY_LOW`	20 s	Spindle (low speed)
3	`MODE_IDLE`	5 s	None
4	`MODE_X_AXIS_ONLY`	10 s	X feed
5	`MODE_Y_AXIS_ONLY`	10 s	Y feed
6	`MODE_IDLE`	5 s	None
7	`MODE_Z_AXIS_ONLY`	5 s	Z feed
8	`MODE_LINEAR_FEED`	30 s	Spindle (high) + X + Y + Z
9	`MODE_IDLE`	10 s	None

in hertz. Because the motor emits six pulses per revolution and the spindle is coupled directly to the motor shaft (the gearbox having been removed), the spindle speed in revolutions per minute is obtained directly as the FG frequency multiplied by ten.

- **Axis position.** Two independent position descriptors are recorded per axis: the commanded half-step count maintained by the stepper task, and the measured slot count accumulated by the photo-interrupter ISR. Recording both allows commanded and realized motion to be compared, which is useful for detecting lost steps or mechanical stalls.

3.2.5 Operating-Mode State Machine and Data Schema

The high-level behavior of the bench is governed by a deterministic state machine in the main control task. Each state, or *operating mode*, fixes the spindle duty and the motion flags of the three axes for a defined duration, and simultaneously updates the textual labels that are written into the data stream. The experiment consists of a fixed sequence of eight modes (Table 3.3) that progresses from rest, through spindle-only and individual-axis motion, to a combined linear-feed mode in which the spindle and all three axes operate together. This sequence is repeated for a configurable number of cycles to collect a statistically meaningful number of segments per mode.

Every 10 ms the logger emits one comma-separated record containing 26 fields, which fall into four groups: (*i*) timing and identification (timestamp, run index, sample index); (*ii*) labels (operating mode, event, transition and anomaly labels); (*iii*) spindle channels (PWM command, current, FG frequency); and (*iv*) per-axis and inertial channels (commanded positions, motion flags, slot counts, and the six IMU axes). The event label marks discrete transitions—such as the start or stop of a motion phase—as single-sample markers, while the operating-mode label persists for the duration of each mode. This labeling scheme is central to the study: although the downstream models are trained without supervision, the embedded labels provide ground truth for quantitatively evaluating how well the learned representations separate operating conditions and isolate injected anomalies (Section 3.3).

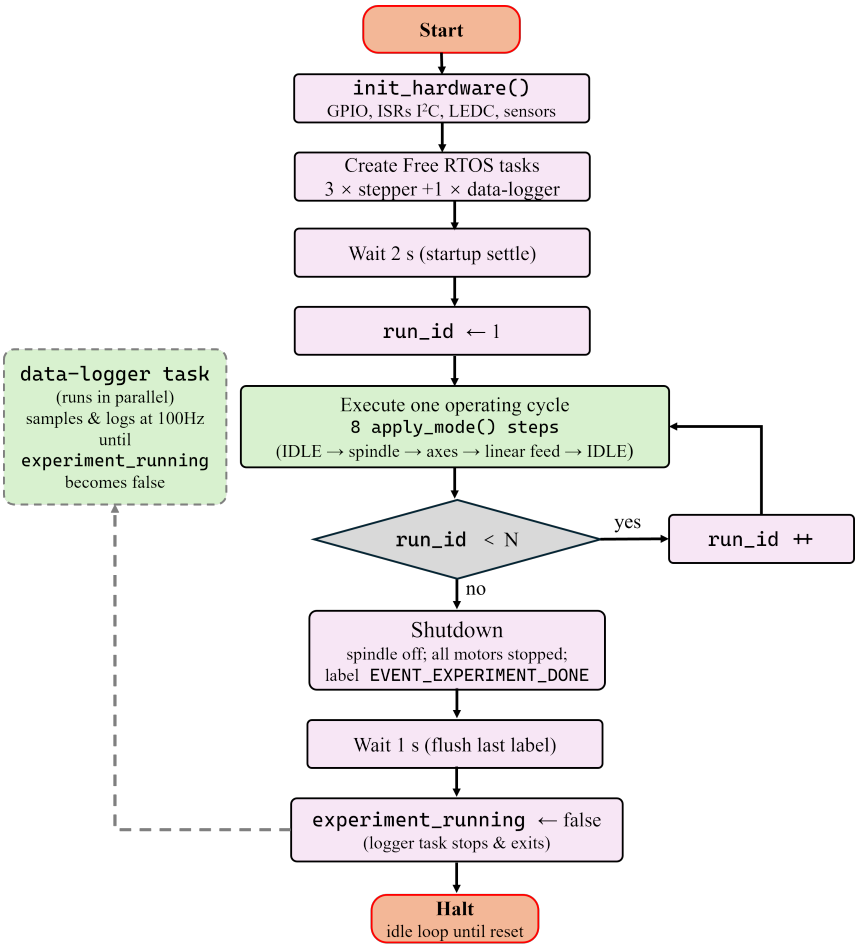

Figure 3.6: Control flow of the main firmware task: hardware initialization, task creation, the repeated operating-cycle loop guarded by the run counter, and the shutdown sequence that signals the logger task to stop.

An explicit anomaly label field is reserved in the schema and held at a null value throughout the nominal sequence described above. Anomalies are introduced in a separate acquisition campaign by commanding motion outside the nominal sequence—for example, driving an axis toward an out-of-range coordinate or forcing simultaneous spindle feed under a stall condition—and tagging the affected samples through this field. Because the nominal and anomalous runs share the same schema and sampling pipeline, anomaly-free data can be used for unsupervised training while the labeled anomalous runs serve exclusively for evaluation.

3.2.6 Data Transport to the Edge Inference Node

In the current implementation the labeled CSV stream is emitted over the serial console and captured on a host for offline processing, which yields the dataset analyzed in this thesis. The target deployment, used to evaluate real-time feasibility, instead transports the same stream over the 100 Mbit s^{-1} Ethernet interface of the ESP32-P4 board: the acquisition node and both Raspberry Pi nodes share a single switched Ethernet segment, allowing the record stream to be delivered to the Raspberry Pi 5

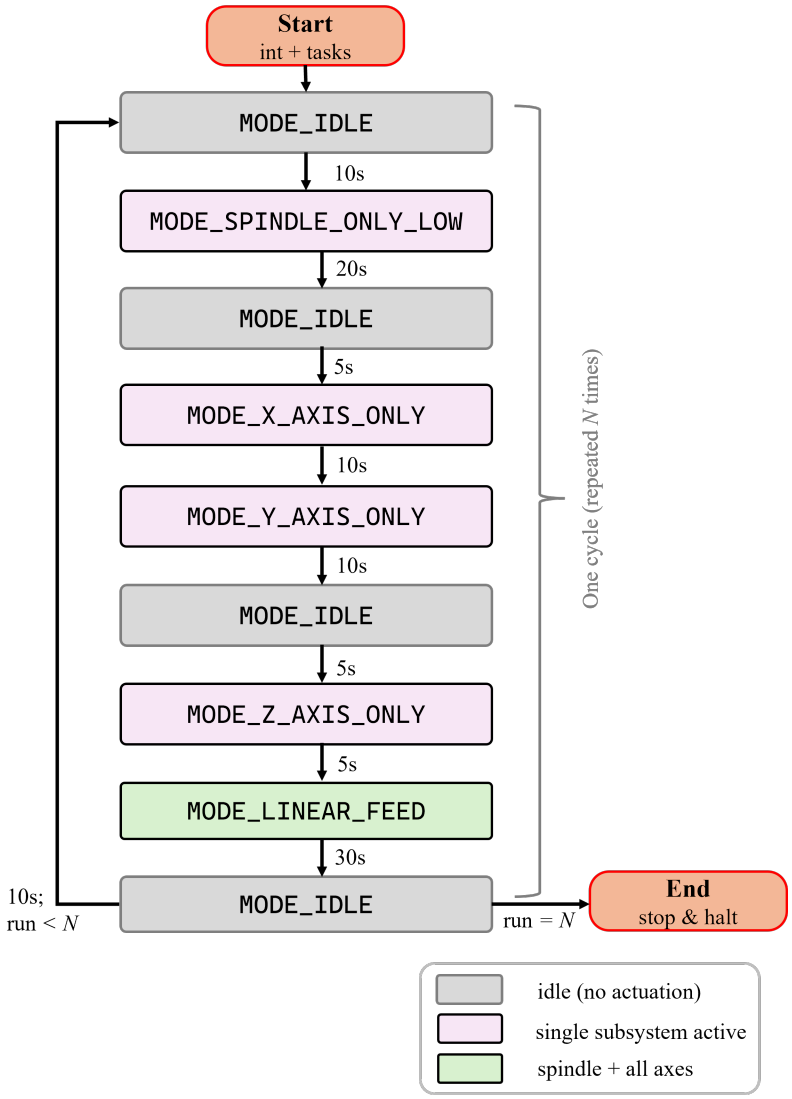

Figure 3.7: Operating-mode state machine. Each cycle steps through the eight modes with the durations shown; the sequence repeats N times before the experiment terminates.

inference node with the Raspberry Pi 3 acting as the network gateway. Because the record format is identical in both cases, models developed on the captured dataset transfer directly to the streaming deployment.

3.3 CNC Data Preprocess

The CNC dataset is built from CSV logs of a CNC milling test bench instrumented by the hardware part of the project. Each log row contains a timestamp, a `run_id`, a `machine_mode` label, and a set of sensor channels. We use the 8 pure sensor-feedback channels and deliberately exclude the control inputs (`spindle_pwm_command`, position commands, sensor counts, `axis_moving`) so that the learned representation reflects what the machine did, not what it was told to do; the channel groups are

summarized in Table 4.2.

Table 3.4: Channels of the CNC dataset.

Group	Channels
Spindle	`spindle_current`, `fg_frequency`
Linear vibration	`acceleration_x`, `acceleration_y`, `acceleration_z`
Angular vibration	`gyro_x`, `gyro_y`, `gyro_z`

3.3.1 Windowing

Each run is split into contiguous pure-mode segments (boundaries placed wherever `machine_mode` changes), and windows of length $T = 256$ with stride 128 are extracted only from segments that fit entirely within a single mode. The window label is the mode of its segment. Segments shorter than the window length are skipped. This procedure guarantees that no window straddles a mode boundary — windows are clean exemplars of a single mode — and the per-run `run_id` is recorded alongside each window so that the downstream geometry metrics can recover the trajectory through embedding space within each run.

3.3.2 Splits

Runs 1–7 form the training set, runs 8–10 form the test set. This is a run-disjoint split: no window in the test set comes from the same physical production run as any window in the training set, which is the right protocol when the eventual deployment scenario is "fit on past production days, run on future days". The resulting dataset has 425 training windows and 173 test windows, each of shape $(256, 8)$.

3.3.3 Normalization

The loaders apply train-only z-normalization. For the multivariate datasets (Mechatronic and UEA) we use per-channel `StandardScaler` fitted on the training set, because the 8 channels have very different physical units and scales.

3.4 Encoders

All four encoders expose the same interface,

```
encode(data: ndarray[N, T, C]) -> ndarray[N, D]
```

and consume the same (N, T, C) windows, so that the rest of the pipeline is encoder-agnostic.

3.4.1 Summary-statistic baseline

For each window, we take the per-channel `mean`, `std`, `min`, `max` and concatenate. With $C = 8$ channels this yields $D = 32$. Zero learned parameters, zero training.

3.4.2 FFT baseline

For each channel we take $|\text{rfft}(x)|$ and keep the first $n_{\text{coef}} = 8$ low-frequency bins (DC included). Concatenated across channels this yields $D = 8 \times 8 = 64$. Zero learned parameters.

3.4.3 TS2Vec

We use the upstream TS2Vec encoder [22] (TCN, depth 10, hidden 64) and run a small hyper-parameter sweep on the CNC training set across (iters, batch, repr_dims) as shown in Table 3.5. Each configuration is repeated with seeds $\{0, 1, 42\}$. At inference, the encoder is called with `encoding_window='full_series'` to produce one vector per window.

Table 3.5: TS2Vec hyper-parameter sweep on CNC.

Config	Iters	Batch	`repr_dims`
Baseline	600	8	320
A (more iter)	3 000	8	320
B (smaller D)	3 000	8	128
C (large batch)	3 000	32	128

3.4.4 MOMENT

We use the public `AutonLab/MOMENT-1-{small,base,large}` checkpoints [23] from Hugging Face, loaded with `task_name='embedding'` and used zero-shot — i.e. no fine-tuning on CNC. MOMENT expects fixed-length univariate inputs of length 512 per channel, so each $(T = 256, C = 8)$ window is (a) transposed to (C, T), (b) left-padded with zeros from $T = 256$ to $T = 512$, with the pad positions marked in `input_mask`, and (c) batched across the channel axis. The embedding dimension is $D = 512$ (small), 768 (base), or 1024 (large).

Three compression configurations are evaluated, all targeting CPU edge deployment:

- **FP32.** Original precision, used as the reference operating point. Inference is on GPU during training-time evaluation and on CPU during the edge benchmark.
- **INT8 dynamic (naive).** `torch.ao.quantization.quantize_dynamic(model, {nn.Linear}, dtype=qint8)`, CPU only, with the quantization engine chosen by host architecture (FBGEMM on x86, QNNPACK on ARM). Every `nn.Linear` layer of the encoder—including the patch value embedding, the T5 self-attention projections Q, K, V, O, and the FFN linears—is quantized.
- **INT8 dynamic (FFN-only).** The same dynamic-quantization call applied only to the linears inside the T5 `DenseReluDense` feed-forward blocks (`wi_0`, `wi_1` and `wo`); the attention projections and the patch embedding are left in FP32. This is motivated by the observation that dot-product attention and the input projection are the precision-sensitive paths in the network, while

the FFN linears tolerate INT8 well. We evaluate both variants because the difference between them is one of the central deployment findings of the thesis (Section 4.4.3).

3.5 Evaluation protocols

3.5.1 SVM linear probe

We follow the protocol from TS2Vec and MOMENT: we extract embeddings on the train and test splits, then fit an RBF-SVM. We report accuracy and AUPRC on the test set. The same evaluation function is used for every encoder.

3.5.2 Clustering

We run KMeans ($k = 6$, the number of ground-truth modes) on the test embeddings of each method and report the Adjusted Rand Index (ARI) against the ground-truth mode labels, plus three internal cluster-quality measures — Silhouette, Davies–Bouldin, and Calinski–Harabasz — that do not see the labels and therefore measure pure geometric separability.

3.5.3 Mode-aware geometry

This is the part of the evaluation pipeline that is specific to the mode-change problem. Let $\mathbf{Z} = \{\mathbf{z}_n\}_{n=1}^{N} \subset \mathbb{R}^D$ be the encoded test embeddings, $y_n \in \{1, \ldots, K\}$ their mode labels, $S_k = \{n : y_n = k\}$ the index set of mode k, r_n the production run from which window n originates, and

$$\mathbf{c}_k \;=\; \frac{1}{|S_k|} \sum_{n \in S_k} \mathbf{z}_n$$

the global centroid of mode k. We compute six metrics.

Mode Separability Index (MSI). A direct analogue of the Fisher discriminant ratio: mean pairwise distance between mode centroids divided by mean within-mode dispersion,

$$\mathrm{MSI} \;=\; \frac{\dfrac{2}{K(K-1)} \displaystyle\sum_{k<k'} \|\mathbf{c}_k - \mathbf{c}_{k'}\|}{\dfrac{1}{K} \displaystyle\sum_{k=1}^{K} \dfrac{1}{|S_k|} \sum_{n \in S_k} \|\mathbf{z}_n - \mathbf{c}_k\|}.$$

Higher is better.

PCA compactness. Let $\mathbf{u}_n \in \mathbb{R}^2$ be the projection of $\mathbf{z}_n$ onto the first two principal components fitted on $\mathbf{Z}$, and let $A_k = \mathrm{Area}(\mathrm{Hull}(\{\mathbf{u}_n : n \in S_k\}))$ be the

convex-hull area of mode k in the PCA plane (modes with $|S_k| < 3$ contribute 0). Then

$$\text{PCA compactness} \;=\; \frac{1}{K}\sum_{k=1}^{K} A_k.$$

Lower is better; this metric penalises encoders that separate modes by distance but spread each mode out in absolute embedding-space units.

Centroid stability — bootstrap. For each mode k we draw $B = 10$ bootstrap subsamples $\mathcal{B}_k^{(b)} \subset S_k$ of size $\lfloor 0.8|S_k| \rfloor$ without replacement, compute the bootstrap centroid $\mathbf{c}_k^{(b)} = |\mathcal{B}_k^{(b)}|^{-1} \sum_{n \in \mathcal{B}_k^{(b)}} \mathbf{z}_n$, and report

$$\text{Stab}_{\text{boot}} \;=\; \frac{1}{K}\sum_{k=1}^{K} \frac{1}{D}\sum_{d=1}^{D} \text{std}_b\Big(c_{k,d}^{(b)}\Big)\,.$$

Lower is better.

Centroid stability — per-run (true). The metric that actually predicts run-to-run reproducibility for downstream anomaly detection. Let $\mathbf{c}_k^{(r)}$ be the centroid of mode k computed from windows of run r only (modes with fewer than two contributing runs are skipped). Then

$$\text{Stab}_{\text{true}} \;=\; \frac{1}{K}\sum_{k=1}^{K} \frac{1}{D}\sum_{d=1}^{D} \text{std}_r\Big(c_{k,d}^{(r)}\Big)\,.$$

Only computable when `run_id` is available, which it is for the CNC test set (three test runs).

Loop consistency. Within each run we split the sequence of test windows into maximal contiguous segments of identical mode label. Let $\mathcal{S}_{k,r} = \{T_{k,r}^{(1)}, T_{k,r}^{(2)}, \dots\}$ be the segments of mode k in run r, where each $T_{k,r}^{(i)} \subset \mathbb{R}^2$ is the trajectory of that segment in the 2D PCA space of $\mathbf{Z}$. The metric is the mean dynamic-time-warping distance between all pairs of same-mode segments within the same run,

$$\text{Loop} \;=\; \underset{r,\,k,\,i<j}{\text{mean}}\; \text{DTW}\Big(T_{k,r}^{(i)},\, T_{k,r}^{(j)}\Big)\,,$$

with DTW the standard ℓ_2 dynamic-time-warping cost. Lower is better.

Transition sharpness. Within each run, every index i at which $y_i \neq y_{i-1}$ is a mode boundary with old label $k_{\text{old}} = y_{i-1}$ and new label $k_{\text{new}} = y_i$. For each such boundary we count

$$\tau_i \;=\; \min\{j - i + 1 \;:\; j \geq i,\; \|\mathbf{z}_j - \mathbf{c}_{k_{\text{new}}}\| \;<\; \|\mathbf{z}_j - \mathbf{c}_{k_{\text{old}}}\|\}\,,$$

i.e. the number of windows after the boundary needed for the embedding to cross the midpoint between the two global centroids. Transition sharpness is the mean of τ_i over all boundaries in all runs; lower is better.

3.5.4 Edge-deployment benchmark

For each encoder we build the encoder on CPU only and force PyTorch to a single thread so that the reported latency does not depend on a particular many-core workstation. The benchmark records three quantities per configuration. Parameter count is taken from `sum(p.numel() for p in model.parameters())`. On-disk size is the serialized size of the `state_dict` written to an in-memory buffer with `torch.save`, which approximates what an OTA update would push to the edge node. Single-window inference latency is measured at `batch_size` $= 1$ with 5 warm-up calls followed by 50 timed calls, and we report mean, standard deviation, p50 and p95 in milliseconds. For MOMENT, the same benchmark is repeated for $(\text{variant}, \text{dtype}) \in \{\text{small}, \text{base}, \text{large}\} \times \{\text{fp32}, \text{int8\_dynamic}, \text{int8\_dynamic\_ffn}\}$ to map the full compression sweep onto the same Pareto.

4

Results

This chapter reports the experimental results in four steps. Section 4.1 characterises the acquired dataset and validates that the test bench produces clean, repeatable, mode-discriminative data. Section 4.2 establishes a prior ranking of the four encoders on 20 public UCR and UEA datasets, using the SVM linear probe together with mean-rank statistics for clustering and geometry. Section 4.3 carries the same four encoders to the CNC test set and shows that the accuracy ranking inverts while the geometric ranking is preserved. Section 4.4 then addresses the two open questions that arise from that inversion: a TS2Vec hyperparameter sweep that tests whether its CNC underperformance is a tuning artefact, and a MOMENT size-and-quantization sweep that maps the leading encoder onto the CPU latency budget of the bench.

4.1 Dataset and Bench Validation

The bench was run through its full programmed sequence, yielding ten complete operating cycles. A single acquisition produced 94 728 recorded samples in total; of these, 94 636 belong to the ten complete cycles and are used for analysis, while a short, one-second shutdown record at the end of the run (92 samples) is excluded. The principal properties of the dataset are summarised in Table 4.1, and the recorded fields are described in Table 4.2.

Table 4.1: Overview of the acquired dataset.

Property	**Value**
Total samples	94 728
Recording duration	1053.4 s (≈17.6 min)
Operating cycles (runs)	10
Nominal sampling rate	100 Hz
Realised sampling rate	90.9 Hz (median 11 ms period)
Operating modes	6
Recorded channels	25 fields (11 sensor-feedback)
Spindle (FG) pulses/rev	6
Encoder slots/rev (per axis)	20
IMU range (accel / gyro)	± 2 g / $\pm 250\,^{\circ}\,\mathrm{s}^{-1}$ (defaults)

Table 4.2: Recorded fields, grouped by role. Inertial channels are raw signed 16-bit counts at the sensor's default ranges; spindle current is in amperes and FG frequency in hertz.

Field(s)	Role	Description / unit
`timestamp`	timing	milliseconds since boot
`run_id`, `sample_index`	timing	cycle index and per-record counter
`machine_mode`	label	current operating mode
`event_label`, `transition_label`	label	single-sample phase markers
`anomaly_label`	label	anomaly tag (null in nominal runs)
`spindle_pwm_command`	command	commanded spindle duty (%)
`x/y/z_position_command`	command	commanded half-step count per axis
`x/y/z_axis_moving`	command	per-axis motion flag (0/1)
`spindle_current`	sensor	spindle supply current (A)
`fg_frequency`	sensor	spindle FG pulse rate (Hz)
`x/y/z_sensor_count`	sensor	photo-interrupter pulse count per axis
`acceleration_x/y/z`	sensor	3-axis acceleration (raw counts, $\pm 2\,\mathrm{g}$)
`gyro_x/y/z`	sensor	3-axis angular rate (raw counts, $\pm 250\,^\circ\,\mathrm{s}^{-1}$)

Timing. Although the logger targets a 100 Hz sampling rate, the realised rate over the full acquisition is 90.9 Hz, with a median inter-sample interval of 11 ms rather than the nominal 10 ms. The difference is consistent with the per-cycle overhead of the two blocking I^2C transactions (inertial unit and current monitor) and the formatted output performed on every logger iteration. The rate is nonetheless stable across the run, so the sampling interval is effectively constant for the purposes of windowed analysis.

Repeatability across cycles. Figure 4.1 shows all ten cycles over the full acquisition, with each operating mode drawn as a distinct coloured background band. The same sequence of modes repeats cleanly ten times: The spindle channels (current and FG frequency) switch between a near-zero idle level and two distinct active levels, and the commanded axis positions increase in a regular staircase as each cycle advances the feed axes. This periodic structure, with no missing or corrupted segments, confirms that the acquisition pipeline runs reliably over the full $\approx$1050 s recording.

Mode-discriminative signal content. Figure 4.2 expands a single representative cycle, with the shaded bands identifying each operating mode, and shows that every mode has a distinct multimodal signature. When the spindle is energised (the `SPINDLE_ONLY_LOW` and `LINEAR_FEED` phases) the supply current and FG frequency rise sharply and the accelerometer and gyroscope show markedly increased broadband activity, whereas the stepper-only phases move the commanded axis positions without engaging the spindle and produce comparatively little vibration. These qualitative observations are quantified in Table 4.3, which reports per-mode statistics over the whole dataset. The spindle-active modes are clearly separated from the

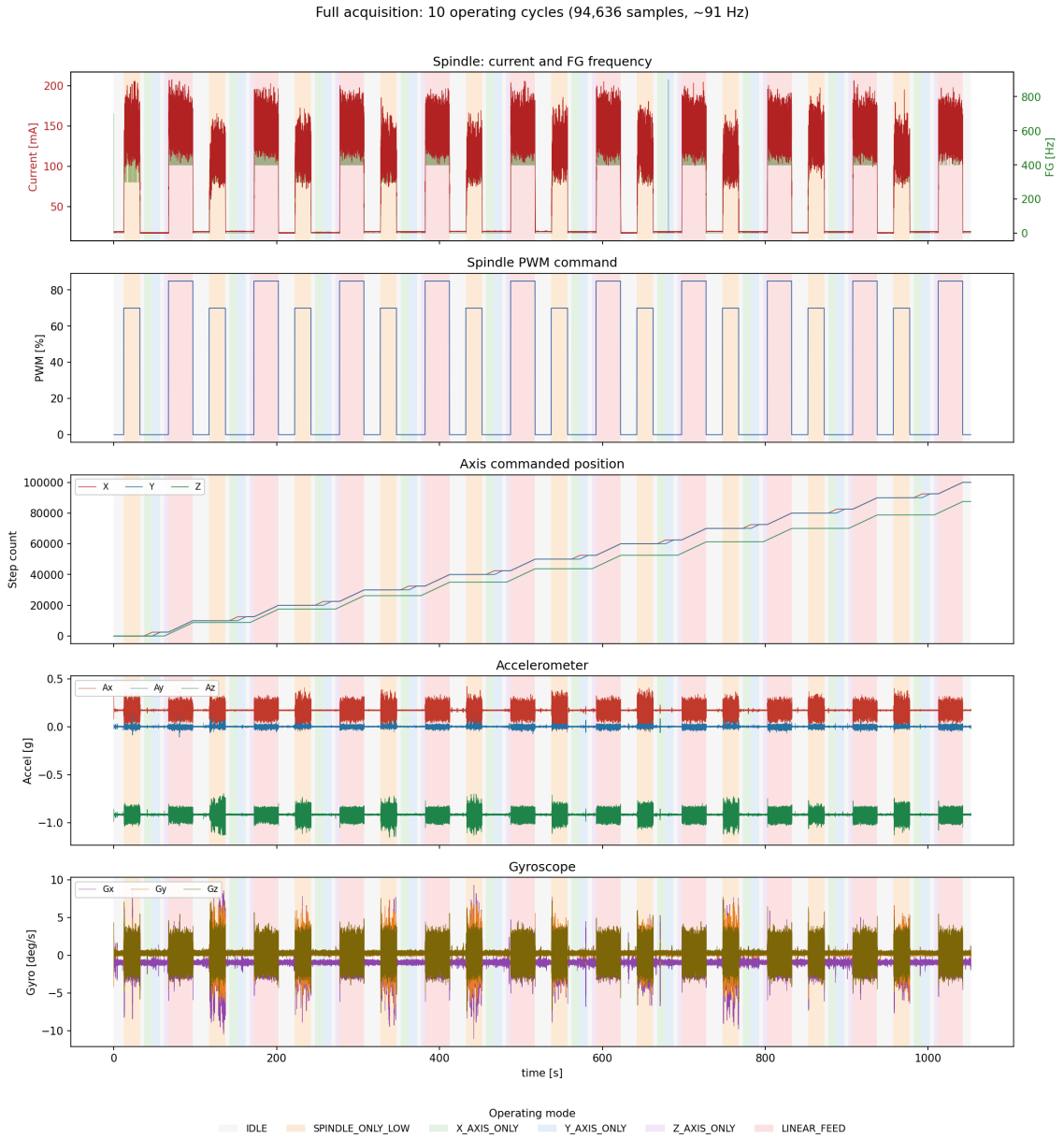

Figure 4.1: Full acquisition across the ten operating cycles. Each operating mode is shown as a coloured background band (see legend); the programmed sequence repeats cleanly, and the staircase in the commanded axis positions reflects the cumulative advance of the feed axes over successive cycles. Spindle current is shown in milliamperes and FG frequency in hertz; accelerometer and gyroscope channels are converted to g and $^\circ\,\mathrm{s}^{-1}$ from the sensor's raw counts.

idle and stepper-only modes: mean spindle current rises from about 19 mA at idle to 116 mA and 143 mA in the two spindle-active modes, FG frequency rises from essentially zero to roughly 390 Hz and 480 Hz, and the gravity-removed vibration RMS increases from a few milli-g to several tens of milli-g. The two spindle-active modes are themselves distinguishable through their different current and FG levels, reflecting the two commanded duty levels. Taken together, these results show that the bench produces synchronized, low-noise, mode-discriminative multimodal data, which is the prerequisite for the representation-learning and anomaly-detection experiments that follow.

Table 4.3: Per-mode signal statistics over the full dataset. Current and FG frequency are given as mean ± standard deviation; vibration is the RMS of the (gravity-removed) acceleration magnitude and angular rate is the RMS gyroscope magnitude. Spindle-active modes are clearly separated from idle and stepper-only modes in both current and vibration.

Mode	n	**Current** (mA)	**FG** (Hz)	**Vib.** (mg)	**Gyro** ($^\circ\,s^{-1}$)
IDLE	28 964	19.4 ± 6.3	4.0 ± 35.9	8.0	1.2
SPINDLE_ONLY_LOW	16 955	115.6 ± 24.9	393.2 ± 82.5	72.6	3.9
X_AXIS_ONLY	9116	18.9 ± 0.6	0.0 ± 0.0	4.0	1.1
Y_AXIS_ONLY	9115	18.9 ± 0.6	0.1 ± 9.4	4.0	1.1
Z_AXIS_ONLY	4552	18.9 ± 0.6	0.0 ± 0.0	3.7	1.1
LINEAR_FEED	26 026	143.3 ± 21.3	480.3 ± 63.0	48.2	2.7

4.2 Evaluation on UCR and UEA datasets

4.2.1 SVM linear-probe accuracy

Table 4.4 reports mean SVM linear-probe accuracy. TS2Vec is the strongest encoder overall (0.8417), edging out MOMENT on UCR by 1.4 pp and beating it on UEA by 7.7 pp. Summary is uniformly the weakest, with the Summary-to-FFT gap (8–12 pp by group) consistently smaller than the FFT-to-TS2Vec gap (6–15 pp), so a hand-crafted descriptor remains a useful lower bound. The UEA group is the one place where the two learned encoders separate: MOMENT (0.6566) actually drops below FFT (0.6717) there, while TS2Vec (0.7334) keeps a clear margin. The split is consistent with MOMENT being trained on univariate sequences and the UEA suite being the multivariate one, an asymmetry that surfaces again in the geometry tables below.

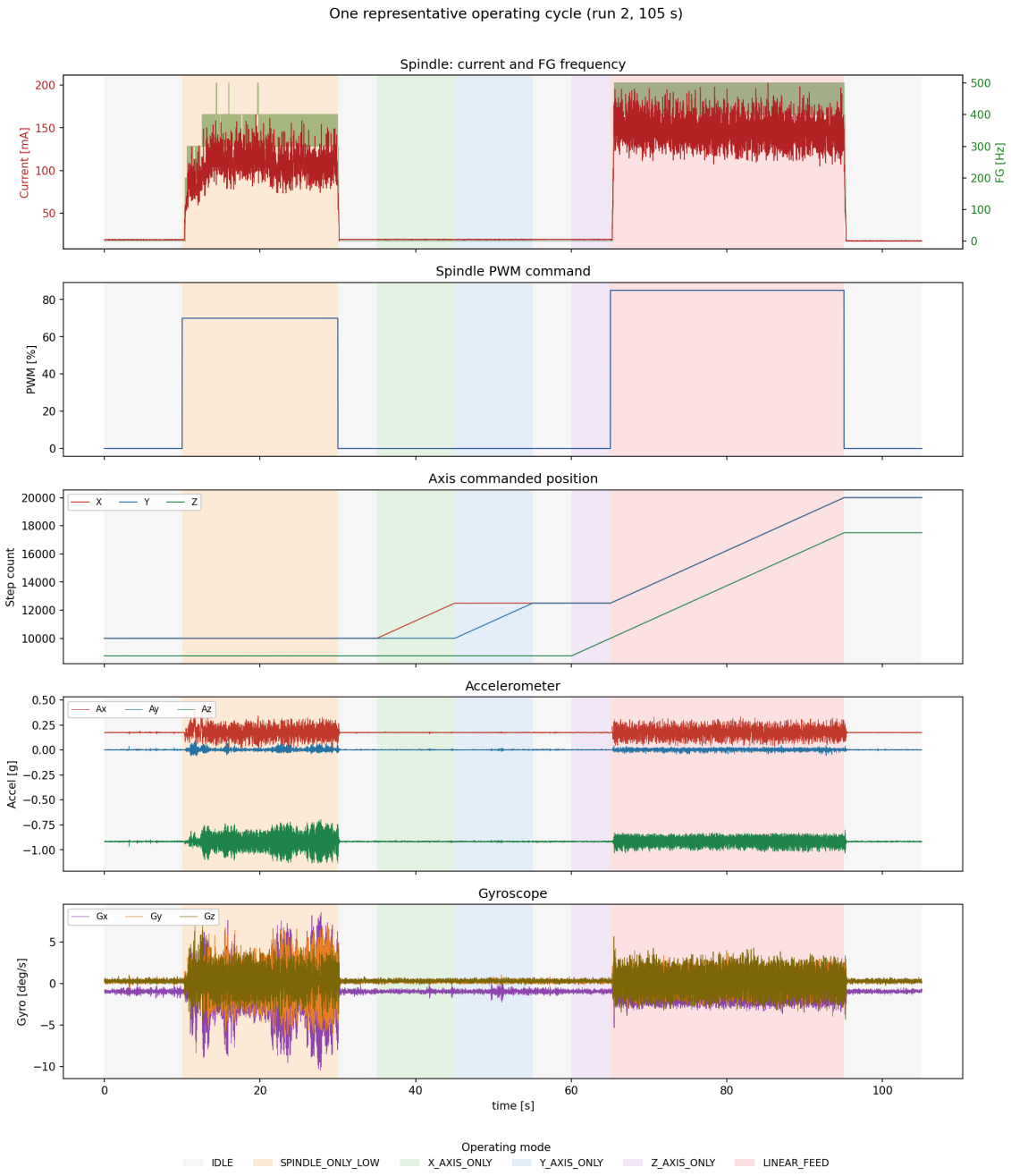

Figure 4.2: One representative operating cycle (run 2). The colored background bands identify each operating mode (see legend). Every mode has a distinct signature: the spindle-active phases raise current, FG frequency and vibration, while the stepper-only phases advance the commanded axis positions with little spindle activity.

Table 4.4: Mean SVM linear-probe accuracy over UCR + UEA datasets ($n = 10$ each).

Method	UCR ($n = 10$)	UEA ($n = 10$)	Overall ($n = 20$)
Summary	0.6838	0.5900	0.6369
FFT	0.7994	0.6717	0.7356
MOMENT	0.9357	0.6566	0.7961
TS2Vec	**0.9499**	**0.7334**	**0.8417**

4.2.2 Clustering and geometry across benchmarks

We extend the cross-benchmark evaluation beyond SVM accuracy with five unsupervised metrics: ARI, silhouette, MSI, PCA compactness, and bootstrap centroid stability. These metrics live on very different absolute scales across datasets. PCA hull area, for example, is in the squared scale of the embedding coordinates and routinely varies by four orders of magnitude between, e.g., `Coffee` (32 train windows, $D = 320$) and `UWaveGestureLibrary` (896 train windows, $D = 1024$). A raw arithmetic mean across datasets would therefore be dominated by a handful of large-scale datasets and would not be comparable between the UCR and UEA groups. We follow the standard practice in the time-series-classification literature [24] and report per-dataset rank statistics: within each (dataset, metric) cell the four encoders are ranked 1–4 (1 = best), and Table 4.5 reports the mean rank over the 20 datasets.

Table 4.5: Mean rank (lower = better) over 20 UCR + UEA datasets, per metric. Ranks are computed per (dataset, metric) and averaged.

Method	ARI ↑	Silhouette ↑	MSI ↑	PCA cpt ↓	Stab boot ↓
FFT	2.70	2.45	2.65	4.00	4.00
Summary	2.75	**1.55**	2.45	2.40	3.00
TS2Vec	**2.10**	2.80	**2.20**	2.40	2.00
MOMENT	2.45	3.20	2.70	**1.20**	**1.00**

Table 4.6 reports the head-to-head record of MOMENT against TS2Vec across the 20 datasets, since these two are the contenders for "best learned encoder".

Table 4.6: Head-to-head W/T/L: MOMENT vs. TS2Vec across the 20 UCR + UEA datasets.

Metric	UCR ($n = 10$)	UEA ($n = 10$)	Overall ($n = 20$)
SVM acc ↑	3 / 2 / 5	2 / 1 / 7	5 / 3 / 12
ARI ↑	3 / 1 / 6	3 / 0 / 7	6 / 1 / 13
Silhouette ↑	1 / 0 / 9	6 / 0 / 4	7 / 0 / 13
MSI ↑	4 / 0 / 6	4 / 0 / 6	8 / 0 / 12
PCA cpt ↓	9 / 0 / 1	9 / 0 / 1	**18 / 0 / 2**
Stab boot ↓	10 / 0 / 0	10 / 0 / 0	**20 / 0 / 0**

Reading the two tables together, two findings dominate and two caveats follow. The first finding is that MOMENT wins centroid stability on 20/20 datasets and PCA compactness on 18/20, with mean ranks 1.00 and 1.20 respectively. The result is the same on the UCR and UEA subgroups (mean rank 1.00 / 1.20 on both), so this geometric advantage holds across both benchmark regimes. The second finding is that the label-aware clustering (ARI) ranking inverts the geometric one: TS2Vec is the strongest ARI encoder (overall mean rank 2.10) and beats MOMENT head-to-head 13–6 across the 20 datasets. The encoder that produces the tightest, most stable clusters is therefore not the encoder whose unsupervised partition best matches the labels, so geometric quality and label-aware quality should be measured separately.

Two caveats temper the table. Silhouette disagrees between the UCR and UEA subgroups: Summary wins overall (mean rank 1.55), but on UCR the ranking is Summary > FFT > TS2Vec > MOMENT (a near-inversion of the ARI ranking), whereas on UEA the four methods are within 1.3 ranks of each other. Silhouette mostly reports how spread out within-cluster points are in the embedding's natural scale, which the low-dimensional handcrafted features score well on by construction. MSI is close to tied across methods: mean ranks span only 2.20–2.70 overall and the head-to-head MOMENT vs. TS2Vec record is 8/0/12, so we treat MSI as uninformative on this benchmark suite and rely on compactness and stability instead.

The split is consistent: TS2Vec is better at label-aware separability (SVM acc, ARI), MOMENT is better at geometric tightness and stability of the embedding space itself (PCA compactness, centroid stability).

4.2.3 Takeaway for the CNC case study

Two observations frame the rest of this chapter. First, on standard benchmarks TS2Vec is the strongest encoder by SVM accuracy and MOMENT is a close second, with FFT and Summary trailing by a wide margin. Second, MOMENT's only consistent advantage is the geometric stability of its embedding space across runs. The CNC milling-spindle case study that follows in Section 4.3 reverses both findings: MOMENT becomes the strongest encoder on SVM accuracy, even the parameter-free Summary baseline outperforms TS2Vec, and the run-to-run stability advantage of MOMENT becomes the decisive factor for downstream anomaly detection.

4.3 CNC milling-spindle case study

This section repeats the four-encoder evaluation of Section 4.2 on the CNC bench data of Section 4.1, under SVM accuracy, KMeans clustering, the six mode-aware geometry metrics, and 2D PCA / t-SNE views. The CNC test set is constructed from the bench log of Section 4.1 by the run-disjoint windowing of Section 3.3: windows of length $T = 256$ with stride 128 are taken only from segments lying fully inside a single operating mode, runs 1–7 form the training set and runs 8–10 form the test set, yielding 425 training and 173 test windows of shape $(256, 8)$ over 6 operating modes. All metrics in this section are computed on the 173 test windows

unless otherwise stated.

4.3.1 Classification accuracy and AUPRC

Table 4.7 reports SVM linear-probe accuracy and AUPRC on the CNC test set. The fp32 ranking is the reverse of the cross-benchmark result: MOMENT-large attains the highest accuracy (0.8497) and AUPRC (0.7492), with a margin of 10.4 pp over TS2Vec. Scaling MOMENT down (large → base → small) costs 6 pp accuracy and almost 16 pp AUPRC. Even the parameter-free Summary baseline (0.8439 / 0.6918) outperforms TS2Vec (0.7457 / 0.4598) on this dataset, a reversal we examine in the Discussion chapter. The effect of post-training int8 dynamic quantization on MOMENT is reported separately in Section 4.4.3, alongside the latency and disk-footprint measurements that motivate it. Whether the TS2Vec underperformance is robust to the choice of training hyperparameters is checked separately in Section 4.4.1.

Table 4.7: CNC test set: SVM linear-probe accuracy and AUPRC for the four encoders.

Encoder	D	SVM acc	AUPRC
FFT	64	0.7341	0.5158
Summary	32	0.8439	0.6918
TS2Vec (baseline, seed 42)	320	0.7457	0.4598
MOMENT-small (fp32)	512	0.7919	0.5868
MOMENT-base (fp32)	768	0.8208	0.6733
MOMENT-large (fp32)	1024	**0.8497**	**0.7492**

4.3.2 Clustering

Table 4.8 gives the KMeans clustering metrics on CNC test embeddings ($k = 6$). MOMENT attains the highest ARI (0.601), meaning that unsupervised partitioning of its embedding space recovers the ground-truth modes more faithfully than any other encoder. Summary attains the highest silhouette (0.698) and the highest Calinski–Harabasz (583), reflecting tight per-mode clusters in hand-crafted statistic space, but its ARI of 0.433 says those tight clusters are only partially aligned with the actual modes. FFT attains the smallest Davies–Bouldin (0.754), narrowly below Summary, but its ARI sits 10 pp below MOMENT. The same split that appeared on UCR and UEA is visible: MOMENT wins on the label-aware metric (ARI), while the internal-geometry metrics (silhouette, DB, CH) reward the lower-dimensional encoders whose clusters are tight in their own coordinates regardless of whether those clusters track the labels.

Table 4.8: KMeans clustering ($k = 6$) on CNC test embeddings.

Encoder	ARI ↑	Silhouette ↑	Davies–Bouldin ↓	Calinski–Harabasz ↑
FFT	0.5010	0.5785	0.7544	161.9
Summary	0.4330	**0.6984**	**0.7473**	**582.9**
TS2Vec	0.4539	0.5487	1.0044	338.0
MOMENT-large	**0.6008**	0.4113	1.1427	257.8

4.3.3 Mode-aware geometry

Table 4.9 gives the six mode-aware geometry metrics defined in Section 3.5.3. The pattern is sharper than SVM accuracy and clustering suggest, and it splits along two axes: structural quality of the embedding space, where MOMENT wins, and absolute margin and response speed, where the simpler encoders win.

On compactness, MOMENT's mode clusters reach a hull area of 0.32 in the 2D PCA plane, roughly 2.5× tighter than TS2Vec (0.79), an order of magnitude tighter than Summary (5.75), and nearly four orders below FFT (2803). The same advantage carries over to centroid stability across runs: the bootstrap proxy and the true per-run estimate agree directionally, and on the true metric MOMENT (0.0078) is approximately 3× more stable than TS2Vec (0.0256) and 11× more stable than Summary (0.0862), while FFT (0.835) is at least an order of magnitude worse than any learned encoder, and two orders worse than MOMENT. MOMENT's pre-training therefore produces an embedding space in which the same mode lands at close to the same place across independent production runs. Loop consistency follows the same ranking: MOMENT's intra-run trace of repeated modes is close to identical (DTW 0.95 in PCA space), TS2Vec is close behind (3.38), Summary noticeably wider (8.72), and FFT degenerate (57.31).

The two metrics that reward absolute margin and quick response go the other way. Summary attains the highest MSI (4.21), followed by TS2Vec (3.87); MOMENT sits at 2.11, slightly below FFT, so its modes are tightly packed but separated by a smaller absolute margin. On transition sharpness, all encoders react within 1.5–3.2 windows of a mode change, with TS2Vec sharpest (1.46), then FFT (1.54) and Summary (2.04); MOMENT is slowest (3.17), reflecting the temporal smoothing of its long-context transformer backbone.

Table 4.9: Mode-aware geometry on CNC test embeddings. Best per column in **bold**. "Stab boot" is the bootstrap proxy of centroid stability; "Stab true" is the run-disjoint estimate over the three independent production runs.

Encoder	MSI ↑	PCA cpt ↓	Stab boot ↓	Stab true ↓	Loop ↓	Trans ↓
FFT	2.162	2803.07	0.2596	0.8352	57.31	1.542
Summary	**4.210**	5.75	0.0213	0.0862	8.72	2.042
TS2Vec	3.872	0.79	0.0054	0.0256	3.38	**1.458**
MOMENT-large	2.111	**0.32**	**0.0014**	**0.0078**	**0.95**	3.167

4.3.4 Visualization

Figure 4.3 and Figure 4.4 project the CNC test embeddings of all four encoders into 2D via PCA and t-SNE. The 2D views tell a more nuanced story than the geometry table alone. Summary, TS2Vec, and MOMENT all cleanly separate the three high-energy modes — `IDLE`, `SPINDLE_ONLY_LOW`, and `LINEAR_FEED` — into distinct clusters. The three single-axis modes (`X / Y / Z_AXIS_ONLY`), however, overlap heavily with each other and with `IDLE` in 2D under every encoder, including MOMENT. This is not a contradiction with Section 4.3.3: the geometry metrics are computed in the full 1024-dimensional (MOMENT) embedding space, whereas the first two PCA components capture only a fraction of the total variance — the axis modes are distinguishable along the remaining directions, which is exactly the dimensionality the SVM probe of Section 4.3.1 has access to. The 2D projections should therefore be read as evidence that the major energy regimes separate cleanly, not as a substitute for the full-space geometry numbers. FFT fails on the lower 2D bar as well: only `LINEAR_FEED` and `SPINDLE_ONLY_LOW` form identifiable clusters, while `IDLE` and all three axis modes collapse into a single undifferentiated blob, consistent with FFT's lowest mean rank on PCA compactness in Table 4.5.

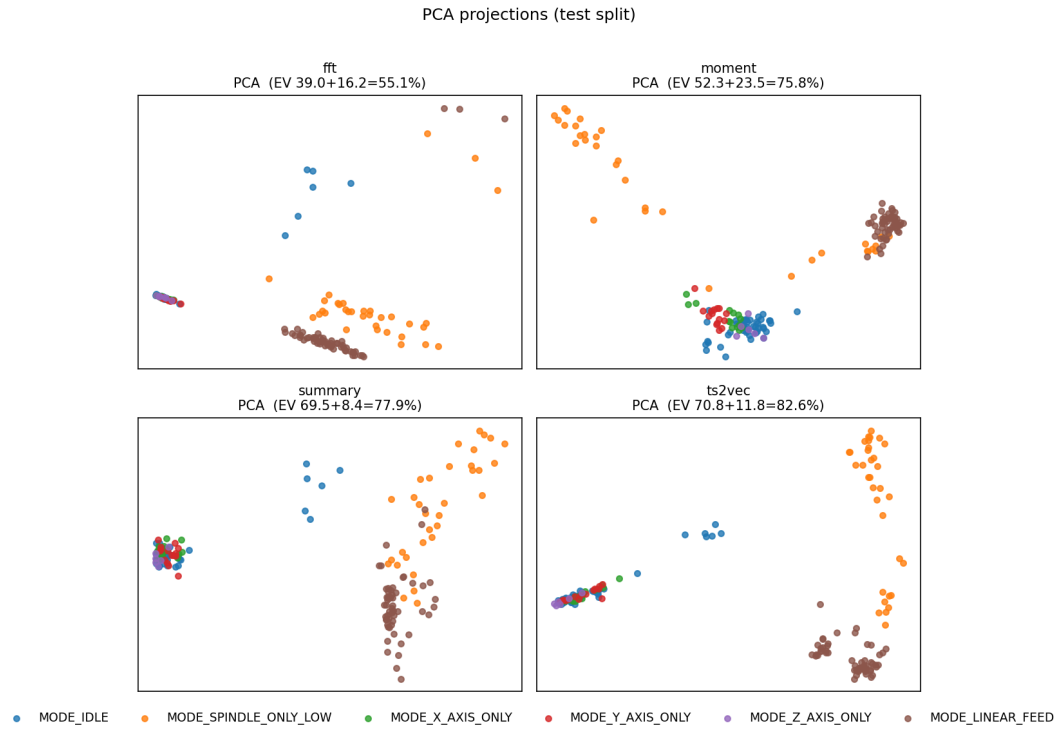

Figure 4.3: PCA projection of CNC test embeddings, four encoders.

4.3.5 Takeaway from the CNC case study

The four cross-encoder views on CNC give a consistent picture and confirm the prediction of Section 4.2.3. The standard-benchmark ranking inverts on accuracy: MOMENT-large is now the strongest encoder, and the parameter-free Summary baseline outperforms TS2Vec by approximately 10 pp at more than two orders of

Figure 4.4: t-SNE projection of CNC test embeddings, four encoders.

magnitude lower cost (Table 4.7). The same inversion is visible on label-aware clustering: MOMENT attains the highest ARI on CNC (Table 4.8), whereas TS2Vec held that position across UCR and UEA. MOMENT's geometric advantage from Section 4.2.2, on the other hand, carries over to CNC and is decisive: its embedding space is several times tighter than TS2Vec and an order of magnitude tighter than Summary on PCA compactness and three to eleven times more stable across runs than the other encoders on run-disjoint centroid stability (Table 4.9). Geometric stability is the property that downstream within-mode anomaly detection actually needs — a baseline that drifts across production runs makes within-mode novelty detection unusable, no matter how well it classifies the modes themselves — so MOMENT remains the candidate of interest on this bench.

Two open questions follow, both deferred to Section 4.4. First, is the TS2Vec underperformance of Section 4.3.1 a hyperparameter accident, or a genuine property of the encoder on this dataset? Second, can MOMENT — which runs at multi-second latency at full precision — be made cheap enough to deploy at edge without losing the geometric advantage that motivated its selection?

4.4 Method-specific analysis and edge deployment

The cross-encoder comparisons of Sections 4.2 and 4.3 measured only quality. This section addresses the two open questions left by Section 4.3.5. Section 4.4.1 checks the robustness of the TS2Vec accuracy result of Section 4.3.1 to a hyperparameter sweep. Sections 4.4.2 and 4.4.3 then add the cost dimension: a four-encoder latency

Pareto, followed by a MOMENT size-and-quantization sweep that asks whether the leading encoder can be brought into the per-window budget of the bench.

4.4.1 TS2Vec hyperparameter robustness

To check whether the TS2Vec result in Table 4.7 is brittle to the choice of training hyperparameters, we sweep three knobs (training iterations, batch size, and representation dimension) and rerun each configuration on three seeds. Table 4.10 reports the mean and standard deviation. Pushing the model longer (configuration A), shrinking the representation (B), and increasing the batch size (C) all help: configuration C reaches 0.7649 ± 0.0027, 2.7 pp above the baseline. Variance is also smallest at C, indicating the larger-batch + smaller-D regime is the more stable operating point on CNC. None of the sweep configurations close the gap to MOMENT-large fp32 (0.8497) or even to Summary (0.8439), so the conclusion of Section 4.3.1 is robust: the canonical TS2Vec representation chosen for clustering and geometry analysis (the seed-42 run with $D = 320$) is representative of what this encoder can do on this dataset.[1]

Table 4.10: TS2Vec hyperparameter sweep on CNC, mean ± standard deviation of test accuracy over three seeds (0, 1, 42).

Config	Iters	Batch	`repr_dims`	Test accuracy
Baseline	600	8	320	0.7380 ± 0.0027
A (more iter)	3000	8	320	0.7534 ± 0.0119
B (smaller D)	3000	8	128	0.7630 ± 0.0170
C (large batch)	3000	32	128	$\mathbf{0.7649 \pm 0.0027}$

4.4.2 Edge-deployment latency Pareto

Table 4.11 adds the cost dimension to the cross-encoder comparison: CPU batch-1 inference latency, on-disk size, and parameter count for the same four encoders, measured on a single-thread reference CPU ($n = 50$ runs per row). The latency spread is five orders of magnitude: FFT at 0.018 ms, Summary at 0.080 ms, TS2Vec at 12 ms, and MOMENT-large fp32 at ≈ 2.4 s. Summary and FFT are essentially free, and TS2Vec is two orders of magnitude cheaper than MOMENT-large at full precision. MOMENT at the operating point that wins on the geometric metrics (Section 4.3.3) is the expensive end of the spectrum, and Section 4.4.3 asks whether it can be brought into a deployable budget without giving up that geometric advantage.

[1] The TS2Vec rows reported in Tables 4.7, 4.8 and 4.9 all use the seed-42 dedicated run with $D = 320$ rather than a sweep configuration; downstream clustering and geometry analyses were not re-run for each sweep cell.

Table 4.11: CNC CPU batch-1 inference benchmark (single thread, $n = 50$).

Encoder	D	Params	Disk (MB)	Latency mean (ms)
FFT	64	0	0	**0.018**
Summary	32	0	0	0.080
TS2Vec	320	0.64 M	2.6	11.96
MOMENT-large fp32	1024	341.2 M	1321.4	2382.8

Figure 4.5 places the operating points of the four encoders on an accuracy–latency plane. The vertical dashed line marks the per-window budget (≈ 1.4 s) dictated by the acquisition rate; everything to its left fits inside the real-time interval on the reference CPU. TS2Vec sits below and to the right of Summary and is therefore dominated on both accuracy and latency on this dataset. FTT has lower accuracy and latency than Summary, while large int8_dynamic_ffn is the only point that beats Summary on accuracy, at the cost of sitting just over the budget line.

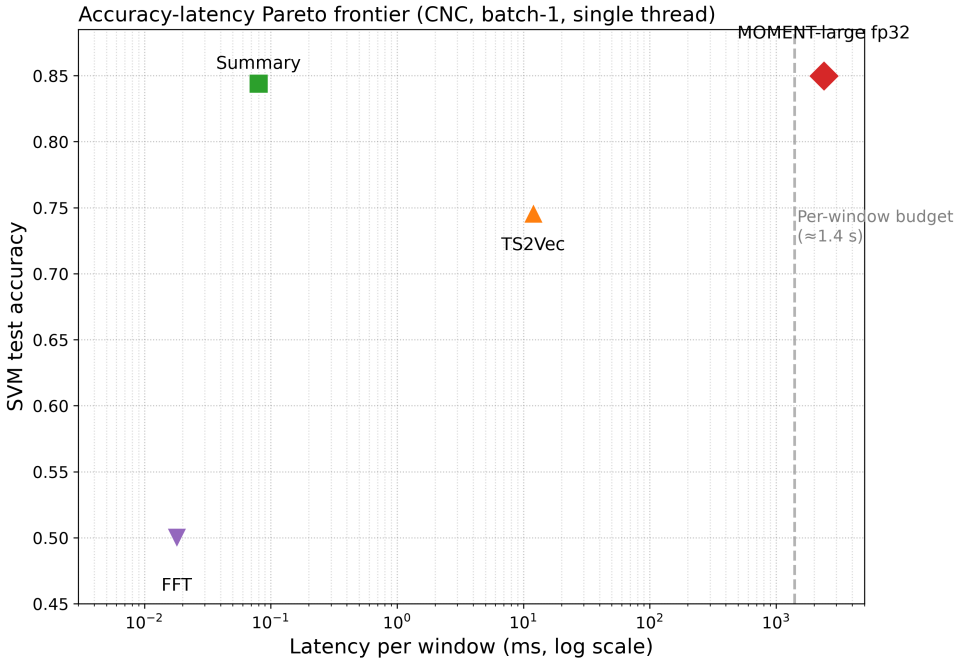

Figure 4.5: Accuracy–latency Pareto frontier for the four encoder operating points on CNC.

4.4.3 MOMENT size and quantization sweep

Table 4.12 sweeps MOMENT across three model sizes and three post-training quantization regimes. Three findings stand out. First, scaling MOMENT down delivers a near-linear latency reduction at modest accuracy cost: small fp32 reaches 0.792 accuracy at 155 ms (vs. 0.850 at 2383 ms for large fp32), a 15× speedup for a 6-pp accuracy hit. Second, naive int8 dynamic quantization is unusable on CNC: every size collapses to chance accuracy (≈ 0.50) and AUPRC near 0.27, even though

the quantized model itself runs and the disk footprint shrinks by 1.6× to 3× depending on size. Third, FFN-only int8 quantization (`int8_dynamic_ffn`, keeping attention in fp32) preserves accuracy at all three sizes: small 0.7861 vs. 0.7919 fp32 (−0.6 pp), base 0.8266 vs. 0.8208 (+0.6 pp), large 0.8555 vs. 0.8497 (+0.6 pp), at a 1.25–1.55× latency reduction and 1.33–1.82× disk-footprint reduction. FFN-only int8 is therefore the recommended deployment regime when a MOMENT-scale encoder is required.

Table 4.12: MOMENT size and quantization sweep on CNC. Naive int8_dynamic quantizes every linear layer; int8_dynamic_ffn quantizes only the FFN linears and keeps attention in fp32.

Size	Dtype	Params	Disk (MB)	Latency mean (ms)	Acc	AUPRC
small	fp32	35.3 M	144.6	154.98	0.7919	0.5868
small	int8_dynamic	16.5 M	90.6	103.74	0.5029	0.2909
small	int8_dynamic_ffn	22.8 M	108.6	123.88	0.7861	0.5743
base	fp32	109.6 M	432.9	614.91	0.8208	0.6733
base	int8_dynamic	24.7 M	190.0	354.78	0.5087	0.2611
base	int8_dynamic_ffn	53.0 M	271.0	469.54	0.8266	0.6654
large	fp32	341.2 M	1321.4	2382.77	0.8497	0.7492
large	int8_dynamic	33.0 M	439.5	1143.93	0.5029	0.2676
large	int8_dynamic_ffn	133.6 M	727.4	1535.24	**0.8555**	**0.7564**

Reading the latency column against the per-window budget of the bench: at the realised acquisition rate of 90.9 Hz (§4.1) and stride 128, a fresh window completes every ≈ 1.4 s. The cheapest accurate MOMENT configuration (small int8_dynamic_ffn, 124 ms) therefore runs in roughly 9% of the per-window budget on the reference CPU, the base configuration fits at ≈ 33%, and only large int8_dynamic_ffn (1535 ms) sits just over budget (≈ 109%) and would require batching across windows, further compression, or a faster deployment CPU to run continuously.

Four competitive points on the accuracy-versus-latency Pareto curve summarize the trade-off (Figure 4.6). Summary reaches 0.844 accuracy at 0.08 ms/window and zero on-disk cost, the cheapest competitive option by orders of magnitude, and is sufficient for tasks that need only mode classification. TS2Vec at the baseline configuration reaches 0.746 accuracy at 12 ms/window and 2.6 MB and is dominated by Summary on all three of accuracy, latency, and disk on this dataset. MOMENT-small int8_dynamic_ffn reaches 0.786 accuracy at 124 ms/window and 109 MB, the cheapest MOMENT configuration that fits comfortably inside the per-window budget; it trails Summary on accuracy by 5.8 pp, but provides the geometric stability across runs that Summary does not (Section 4.3.3). MOMENT-large int8_dynamic_ffn reaches 0.856 accuracy at 1535 ms/window and 727 MB, the only configuration in the benchmark that beats Summary on accuracy (by 0.012), at the price of sitting just over the per-window budget.

The Discussion chapter returns to these trade-offs and to the deployment recommendation that follows from them.

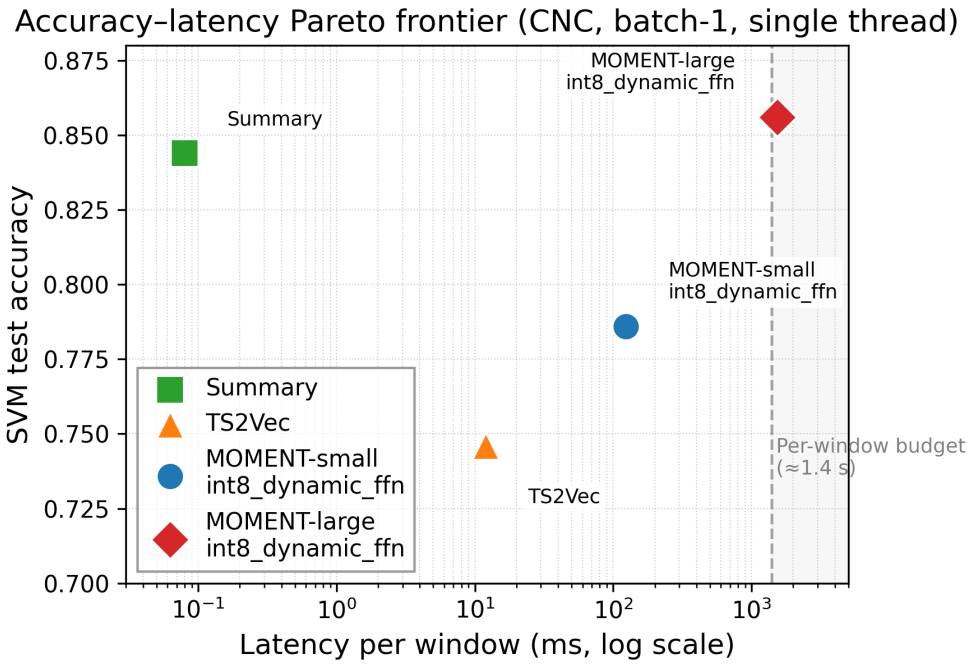

Figure 4.6: Accuracy–latency Pareto frontier for the four competitive configurations operating points on CNC.

5

Discussion

This chapter interprets the experimental findings of Chapter 4. The evaluation was structured along four complementary axes (Section 1.3): a supervised linear probe, unsupervised clustering, mode-aware geometry, and an edge-deployment benchmark. Each axis gives a different ranking of the four encoders, and the differences are themselves the most informative finding. Section 5.1 explains the split between geometric structure and label-aware separability and what it implies for downstream anomaly detection. Section 5.2 discusses the inversion between the cross-benchmark and CNC results, in particular why the parameter-free Summary baseline outperforms TS2Vec on this dataset. Section 5.3 analyses the post-training quantization results and the attention-versus-FFN sensitivity that makes naive INT8 unusable on this task. Section 5.4 returns to the real-time budget on the target inference node, and Section 5.5 states the limitations that bound the conclusions. Finally, Section 5.6 declares the AI-assisted tools used during the preparation of this thesis and defines the boundary between tool-assisted and author-controlled work.

5.1 Geometric structure versus label-aware separability

A consistent pattern in both the cross-benchmark sweep (Section 4.2.2) and the CNC case study (Section 4.3.3) is that no single encoder wins every metric. The encoder with the tightest, most stable embedding geometry is not the encoder whose unsupervised partition best matches the labels, and is not the encoder with the sharpest response to a mode boundary.

On the 20 UCR and UEA datasets, MOMENT attains rank 1 on bootstrap centroid stability for 20 out of 20 datasets and on PCA compactness for 18 out of 20, with mean ranks 1.00 and 1.20. The geometric advantage is consistent across the two benchmark groups. TS2Vec, on the other hand, has the best mean ARI (2.10) and the best mean SVM accuracy overall (0.8417 vs. 0.7961 for MOMENT). The geometric and the label-aware rankings therefore split cleanly: MOMENT produces a geometrically faithful embedding space, TS2Vec produces representations on which a learned head separates classes more accurately.

The CNC case study reproduces only the geometric half of this pattern. MOMENT-large is rank 1 on PCA compactness (hull area 0.32 vs. 0.79 for TS2Vec and 5.75 for Summary), bootstrap stability (0.0014 vs. 0.0054 and 0.0213), per-run true stability

(0.0078 vs. 0.0256 and 0.0862), and loop consistency (0.95 vs. 3.38 and 8.72). On the absolute-margin metric, MSI, Summary is highest at 4.21 and MOMENT trails at 2.11. On transition sharpness, TS2Vec reacts in 1.46 windows while MOMENT needs 3.17, reflecting the temporal smoothing of the long-context transformer backbone.

The methodological consequence is that geometric quality and classification accuracy should be reported separately because they answer different downstream questions. The supervised linear probe asks whether some boundary exists in the embedding space, which is the right question for mode classification on the inference node; the encoder with the most label-aware separability wins there. The per-run stability metric asks whether the centroid learned from one production run still describes the same mode on a future run, which is the right question for within-mode anomaly detection on top of those embeddings. The two tasks admit different answers and need different evidence, and a representation strong on one does not automatically transfer to the other.

5.2 Inversion between cross-benchmark and CNC, and the Summary baseline

The most counter-intuitive finding is that the SVM ranking on CNC inverts the cross-benchmark ranking. On the 20 UCR and UEA datasets, TS2Vec is the strongest encoder on average accuracy (0.8417) and MOMENT is second (0.7961). On CNC, MOMENT-large is the strongest (0.8497), even the parameter-free Summary baseline (0.8439) outperforms TS2Vec (0.7457), and the gap from MOMENT-large to TS2Vec is roughly ten percentage points. Three factors plausibly account for the reversal.

First, the CNC training set is small. After windowing with $T = 256$ and stride 128 on runs 1–7, only 425 training windows remain (Section 3.3). The hyper-parameter sweep in Table 4.10 shows that the strongest TS2Vec configuration on CNC reaches 0.7649 ± 0.0027, still 8.5 percentage points below MOMENT-large and 7.9 percentage points below Summary. Pushing more iterations, smaller representation dimension, or larger batches all help, but none of the configurations close the gap. The sweep variance is also smallest at the most regularised configuration (C), which is consistent with TS2Vec being capacity-limited by training-set size rather than by architecture on this dataset.

Second, several sensor channels on this bench encode the operating mode in their first two moments. The eight channels used for representation learning are spindle current, FG frequency, three-axis acceleration, and three-axis angular rate (Table 4.2). The per-mode statistics in Table 4.3 show that spindle current alone moves from 19 mA at idle to 116 mA in the spindle-only mode and 143 mA in linear feed, FG frequency moves from ≈ 0 to 480 Hz, and gravity-removed vibration RMS moves from a few milli-g to tens of milli-g. Per-channel mean and standard deviation already give an informative mode signature in this regime, so a hand-crafted statistical descriptor is not a weak baseline on this data, it is a near-optimal one.

Third, MOMENT enters the comparison with weights already trained on a large external corpus of public time series. The CNC training set never had to support that representation learning; only the linear probe on top of it is fit to the 425 windows. This is the regime in which transferred features tend to dominate, and it is consistent with the CNC outcome.

The reading is therefore not that TS2Vec is the wrong family of model, but that contrastive pre-training on 425 windows of single-bench data is the wrong recipe for this dataset. The cross-benchmark numbers show that the same encoder is the overall accuracy winner once the per-dataset training budget is larger and the channels are not already as mode-discriminative as they are here. A more useful TS2Vec on this kind of bench would require pre-training on weeks of acquired data rather than a single labeled campaign.

5.3 Post-training quantization and the attention-versus-FFN split

The post-training INT8 results in Table 4.12 contain the most actionable deployment finding. Naive dynamic INT8 quantization, applied to every `nn.Linear` module of the encoder, collapses accuracy at every model size: 0.5029, 0.5087, and 0.5029 for small, base, and large respectively, against 0.7919, 0.8208, and 0.8497 in FP32 — a drop of roughly 29 to 35 percentage points. AUPRC follows the same pattern, falling from 0.59–0.75 to 0.26–0.29. The quantized model still runs and the on-disk size shrinks by 1.6× to 3.0× depending on size, so the failure is silent: the latency and footprint numbers look like a success while the encoder no longer separates modes.

Restricting the same INT8 cast to the FFN linears (the T5 `DenseReluDense` `wi_0`, `wi_1`, and `wo` modules) and leaving the attention Q, K, V, O projections and the patch embedding in FP32 removes the collapse entirely. Accuracy moves to 0.7861, 0.8266, and 0.8555 at the three sizes, which is within a fraction of a percentage point of the FP32 reference and slightly above it at base and large. AUPRC tracks the same recovery. The latency reduction over FP32 is 1.25× for small, 1.31× for base, and 1.55× for large; the disk-size reduction is 1.33×, 1.60×, and 1.82× respectively. FFN-only INT8 is therefore the only INT8 regime that is both faster than FP32 and as accurate as FP32 on this dataset.

The methodological implication is that for transformer-based time-series encoders deployed under post-training INT8, the attention projections need to be treated as a precision-sensitive path. Dot-product attention is bilinear in Q and K and quadratic in the softmax, so quantization noise on those projections compounds non-linearly through the score matrix in a way that the row-wise dynamic scheme used by `quantize_dynamic` does not handle. The FFN is by contrast a per-token affine map followed by a non-linearity, and tolerates INT8 weights with FP32 activations. Splitting these two paths is a near-zero engineering cost and recovers the full FP32 accuracy at the deployment latency and footprint of the quantized model. Because the naive INT8 model still runs and still reports favourable latency and footprint, a

quantization study that did not also measure accuracy or AUPRC on the quantized model would have endorsed it; reporting cost without label-based metrics is what makes the failure silent.

5.4 Real-time feasibility on the target node

The edge benchmark in Table 4.11 and Table 4.12 maps a five-orders-of-magnitude latency spread onto the four encoders, and the practical question is which of those operating points fit inside the bench's per-window budget. FFT runs in 0.018 ms per window and Summary in 0.080 ms, both with zero learned parameters and zero on-disk footprint. TS2Vec at the baseline configuration runs in 11.96 ms per window with a 2.6 MB state dictionary. MOMENT-large FP32 runs in 2.38 s per window with a 1.32 GB state dictionary; MOMENT-small FP32 takes 0.155 s. The cheapest accurate MOMENT configuration, small with FFN-only INT8, runs in 0.124 s per window with 109 MB on disk.

Read against the budget cited in Section 4.4.3, the recommended MOMENT configurations are largely inside the per-window budget on the reference CPU. At the realized acquisition rate of 90.9 Hz and stride 128, a fresh window completes every $\approx$ 1.4 s: small int8_dynamic_ffn (0.124 s) uses about 9% of that interval, base int8_dynamic_ffn (0.470 s) uses $\approx$ 33%, and only large int8_dynamic_ffn (1.535 s) sits just over budget at $\approx$ 109%. Three paths address the remaining marginal case for large, and all three remain relevant if absolute latency on the deployment node turns out to be tighter than on the reference. The first is a deployment-CPU re-measurement on the Raspberry Pi 5 with the QNNPACK backend; the FBGEMM x86 numbers reported here are a proxy for the relative ordering of encoders only, and the absolute ARM latency may differ in either direction. The second is batching across windows, which amortizes the constant per-call overhead of the transformer forward pass over multiple windows at the price of a one-window detection delay. The third is a per-component quantization scheme that also handles attention, for example weight-only INT4 on the FFN combined with a static-scale INT8 for the attention projections; this is outside the scope of this thesis and is listed as future work.

The Summary baseline sits at the other extreme of the same Pareto curve. It reaches 0.8439 SVM accuracy on CNC at 0.080 ms per window and zero on-disk cost, so the headline accuracy gap to MOMENT-large FFN-only INT8 (0.8555) is 1.2 percentage points at roughly four orders of magnitude lower latency. On the label-aware view alone, Summary is the cheapest competitive option on this bench. The geometry metrics, however, place MOMENT roughly an order of magnitude tighter than Summary on these properties (0.0078 vs. 0.0862 on per-run centroid stability, 0.32 vs. 5.75 on PCA compactness; Table 4.9), and these are the properties that matter for a downstream anomaly detector that needs to use the same centroid across production days. The Pareto choice is therefore not between accuracy and latency in isolation, but between two operating points: a label-aware-only point (Summary), and a geometry-aware point that pays a latency cost (MOMENT FFN-only INT8) for an embedding whose centroids generalise across runs.

5.5 Limitations

Several limitations bound the scope of the conclusions above and define the natural follow-up work.

No anomaly evaluation yet. The dataset analyzed in this thesis was collected exclusively under the nominal operating-mode sequence, so the anomaly-label field defined in Section 3.2.5 is null throughout. The evaluation reported here therefore measures mode separability and the geometric properties of the embedding space, not the detection of injected faults. The mode-aware geometry metrics, particularly per-run centroid stability and loop consistency, are presented as proxies for downstream anomaly-detection viability rather than as a substitute for an end-to-end detector benchmark.

Single CPU reference. All latency measurements were taken on a single host CPU configured to one thread with the FBGEMM backend, so that the reported numbers are reproducible and do not reflect any particular many-core workstation. The deployment target is a Raspberry Pi 5 running the QNNPACK backend, which has a different microarchitecture, smaller caches, and different INT8 kernels. The relative ordering of encoders is unlikely to change, but absolute latency on the target may differ from the reported reference numbers and needs to be re-measured before any real-time claim is made on the deployed hardware.

Controlled bench rather than a production machine. The dataset was acquired on a purpose-built test bench operated through a deterministic mode sequence, so the per-mode signal statistics are cleaner than they would be on a production CNC. In particular, structure-borne noise from neighboring equipment, coolant pumps, tool changes, and operator interaction are absent. The transfer of the MOMENT FFN-only INT8 recipe to a real machine is plausible but is not demonstrated here.

No fine-tuning of MOMENT. MOMENT was used strictly zero-shot, with the public pre-trained checkpoints and no parameter update on CNC data. The accuracy and geometry numbers reported here are therefore a lower bound on what the same backbone could reach with light fine-tuning. The compression analysis in Section 5.3 would need to be repeated on the fine-tuned weights, since post-training quantization of a fine-tuned model can behave differently from the pre-trained checkpoint.

No INT4 or activation quantization. The compression sweep covered FP32 and two dynamic INT8 variants. Lower-bit weight-only schemes such as INT4, and static-scale activation quantization with calibration data, were not evaluated and are the natural next compression step once the attention sensitivity exposed by the naive INT8 result is taken into account.

5.6 Ethical Considerations

Generative AI tools, primarily Claude (Anthropic) and to a lesser extent ChatGPT (OpenAI), were used during the preparation of this thesis in a supporting capacity. Their use fell into two categories.

Writing and language support. Both tools were consulted for English grammar correction, sentence-level rephrasing, and prose clarity. All technical content, argumentation, and conclusions were formulated by the authors; the AI tools were used only to improve the readability of text that had already been drafted.

Code optimization. Python scripts for cleaning the raw PuTTY serial log, computing per-mode summary statistics, and generating the time-series visualizations in Chapter 4 were produced with AI assistance. Generative AI tools were used to assist with code optimization. All core research concepts, experimental designs, algorithmic architectures, and data interpretations remain the independent work of the authors. The authors reviewed all script outputs, cross-checked the computed statistics against independent manual inspection of the data, and verified that the figures faithfully represented the acquired dataset.

AI tools were not used to design the hardware platform, write or debug the embedded firmware, select or configure the machine-learning models, train or evaluate any encoder, interpret experimental results, or determine the conclusions of the thesis. All research decisions, experimental design choices, result interpretation, and final conclusions are the sole responsibility of the authors, who reviewed and approved every section of the document prior to submission.

6

Conclusion

This thesis evaluated compact time-series representations for unsupervised condition monitoring under edge-deployment constraints, on 20 public UCR and UEA datasets as a cross-benchmark reference and a purpose-built CNC milling-spindle test bench as the in-scope target. The evaluation was deliberately multi-view: a supervised linear probe, unsupervised clustering, six mode-aware geometry metrics, and a CPU edge-deployment benchmark, applied to four encoders that span a wide capacity range (per-channel summary statistics, FFT amplitude bins, the self-supervised TS2Vec encoder, and three sizes of the pre-trained MOMENT transformer under three precision regimes).

Four findings stand out. First, the ranking of encoders depends on the question being asked. On the 20 UCR and UEA datasets, TS2Vec is the strongest encoder by mean SVM accuracy (0.8417 overall) and MOMENT is the strongest by geometric stability (rank 1 on bootstrap centroid stability for all 20 datasets and on PCA compactness for 18 of 20). On the CNC test set, MOMENT-large reaches the highest accuracy (0.8497) and the tightest geometry (PCA compactness 0.32, per-run centroid stability 0.0078), while the parameter-free Summary baseline reaches 0.8439 accuracy and the sharpest absolute margin (MSI 4.21). Geometric stability and label-aware separability rank the encoders differently on the same data, and a single-metric summary therefore hides the property that determines whether the embedding generalizes across production runs.

Second, the Summary baseline is a near-optimal classifier on this bench but a poor representation for downstream within-mode anomaly detection. Per-channel mean and standard deviation already encode the operating mode through the spindle current, FG frequency, and vibration RMS, so a hand-crafted descriptor lands within 0.6 pp of MOMENT-large at four orders of magnitude lower latency. Its per-run centroid stability and PCA compactness are however roughly an order of magnitude worse than MOMENT (0.0862 vs. 0.0078 on stability, 5.75 vs. 0.32 on compactness), so Summary is the cheaper operating point only when run-disjoint geometric stability does not matter.

Third, post-training INT8 quantization is viable for the MOMENT backbone only when restricted to the FFN linears. Naive dynamic INT8, applied to every `nn.Linear` of the encoder, collapses test accuracy at every model size (down to ≈ 0.50, roughly 30 percentage points below FP32) while latency and disk size still report as wins, so a quantization audit that checks only deployment cost endorses a broken model. Restricting INT8 to the T5 `DenseReluDense` modules and leaving the attention

projections in FP32 preserves FP32 accuracy at all three sizes (small 0.7861, base 0.8266, large 0.8555) and gives a 1.25–1.55× latency reduction and a 1.33–1.82× disk-size reduction over FP32. The attention path is the precision-sensitive component, and any deployment recipe for transformer-based time-series encoders should treat it separately.

Fourth, real-time inference is feasible on the reference CPU for the recommended configurations up to base size. At the realized acquisition rate of 90.9 Hz and stride 128 (a fresh window completes every ≈ 1.4 s), MOMENT-small with FFN-only INT8 runs in 124 ms per window on a single FBGEMM thread — about 9% of the per-window budget — and the base variant runs in 470 ms (≈ 33%). Only MOMENT-large int8_dynamic_ffn (1535 ms, ≈ 109%) sits just over the budget on this reference and would need batching or further compression to run continuously. The Summary baseline runs three orders of magnitude faster than any MOMENT variant on the same reference but does not provide the geometric stability needed for run-disjoint anomaly detection.

The recommended deployment on the current evidence is therefore conditional. For monitoring tasks that need only mode classification on the inference node, the Summary baseline is sufficient and almost free. For tasks that need a stable, low-curvature embedding space across production days — the regime in which downstream within-mode anomaly detection becomes tractable — MOMENT-small with FFN-only dynamic INT8 is the cheapest variant that retains MOMENT's geometric advantage and runs at ≈ 9% of the per-window budget on the reference CPU. MOMENT-base int8_dynamic_ffn is the higher-accuracy fallback (+4 pp accuracy at ≈ 33% budget), and an on-target benchmark on the Raspberry Pi 5 is the natural next step before either configuration is committed.

Three concrete follow-up steps complete the loop from this representation study to a deployed monitor:

1. Acquire a labeled anomaly campaign on the bench, using the anomaly-label field reserved in the schema (Section 3.2.5), and evaluate within-mode anomaly detection on top of the mode-aware embeddings rather than only on their geometric proxies.
2. Re-measure the latency of the recommended FFN-only INT8 configurations on the actual Raspberry Pi 5 inference node with the QNNPACK backend, since the FBGEMM x86 reference is only a proxy for the deployment microarchitecture.
3. Extend the compression sweep beyond dynamic INT8. Weight-only INT4 on the FFN and a calibrated static-scale scheme for the attention projections are the two paths most likely to bring MOMENT-large inside the per-window budget on the reference CPU. Both would also shrink the on-disk footprint of the small and base configurations (109 and 271 MB at FFN-only INT8) for inference nodes with constrained storage, without sacrificing the geometric properties that motivated MOMENT's selection.

Bibliography

[1] ISO, *Condition monitoring and diagnostics of machines — General guidelines*, International Organization for Standardization Std. 17 359, 2018.

[2] IAEA, "Implementation strategies and tools for condition based maintenance at nuclear power plants," International Atomic Energy Agency, Tech. Rep. IAEA-TECDOC-1551, 2007.

[3] S. Kamm *et al.*, "A survey on machine learning based analysis of heterogeneous data in the industrial automation domain," *Computers in Industry*, vol. 149, p. 103928, 2023, complete author list before submission.

[4] E. Laftchiev and S. Sun, "Anomaly detection in discrete manufacturing systems using event sequences," *IFAC-PapersOnLine*, 2018, verify volume/issue/pages before submission.

[5] R. B. Randall and J. Antoni, "Rolling element bearing diagnostics—A tutorial," *Mechanical Systems and Signal Processing*, vol. 25, no. 2, pp. 485–520, 2011.

[6] M. Tiboni, C. Remino, R. Bussola, and C. Amici, "A review on vibration-based condition monitoring of rotating machinery," *Applied Sciences*, vol. 12, no. 3, p. 972, 2022.

[7] Z. Liu, A. Cichoń *et al.*, "Technology development and commercial applications of industrial fault diagnosis system: a review," *The International Journal of Advanced Manufacturing Technology*, vol. 118, pp. 3497–3529, 2021, verify volume/pages and full author list before submission.

[8] O. Surucu, S. A. Gadsden, and J. Yawney, "Condition monitoring using machine learning: A review of theory, applications, and recent advances," *Expert Systems with Applications*, vol. 221, p. 119738, 2023.

[9] O. Diallo *et al.*, "Identifying benchmarks for failure prediction in Industry 4.0," *Applied Sciences*, vol. 11, no. 6, p. 2555, 2021, complete author list before submission.

[10] B. Veloso, J. Gama, B. Malheiro, and J. Vinagre, "The MetroPT dataset for predictive maintenance," *Scientific Data*, vol. 9, p. 764, 2022.

[11] Z. Z. Darban, G. I. Webb, S. Pan, C. C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ACM Computing Surveys*, vol. 57, no. 1, p. 15, 2024.

[12] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "TS2Vec: Towards universal representation of time series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8980–8987.

[13] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski, "MOMENT: A family of open time-series foundation models," in *International Conference on Machine Learning (ICML)*, 2024, arXiv:2402.03885.

[14] M. Tan, M. A. Merrill, V. Gupta, T. Althoff, and T. Hartvigsen, "Are language models actually useful for time series forecasting?" in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, 2024, arXiv:2406.16964.

[15] R. Wu and E. J. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 3, pp. 2421–2429, 2023.

[16] L. Pinciroli, N. R. P. Vago *et al.*, "Predicting machine failures from multivariate time series: An industrial case study," *arXiv preprint arXiv:2401.16738*, 2024, complete author list before submission.

[17] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu, "Data-centric artificial intelligence: A survey," *ACM Computing Surveys*, vol. 57, no. 5, 2023, arXiv:2303.10158; verify volume/number/pages before submission.

[18] M. A. Merrill, M. Tan, V. Gupta, T. Hartvigsen, and T. Althoff, "Language models still struggle to zero-shot reason about time series," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 3512–3533.

[19] V. Shankar, "Edge AI: A comprehensive survey of technologies, applications, and challenges," in *2024 1st International Conference on Advanced Computing and Emerging Technologies (ACET)*. IEEE, 2024, pp. 1–6.

[20] N. N. Alajlan and D. M. Ibrahim, "TinyML: Enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications," *Micromachines*, vol. 13, no. 6, p. 851, 2022.

[21] L. Wulfert, J. Kühnel, L. Krupp, J. Viga, C. Wiede, P. Gembaczka, and A. Grabmaier, "AIfES: A next-generation edge AI framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 6, pp. 4519–4533, 2024.

[22] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "TS2Vec: Towards universal representation of time series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8980–8987.

[23] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski, "MOMENT: A family of open time-series foundation models," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[24] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

A

Appendix 1

This appendix lists the per-dataset values that underlie the aggregated cross-benchmark tables of Section 4.2. Table 4.4 (SVM linear-probe accuracy, group means) is expanded in Table A.1; the rank summary of Table 4.5 is expanded as one table per metric (Tables A.2–A.6). All values are computed on the test split of each dataset with the protocol of Sections 4.2.1 and 4.2.2. Best value per dataset is shown in bold.

Table A.1: SVM linear-probe accuracy on the test split of each UCR/UEA dataset. Group means at the bottom match Table 4.4.

Dataset	FFT	Summary	TS2Vec	MOMENT
UCR (univariate)				
ECG200	**0.8900**	0.7300	0.8800	**0.8900**
Chinatown	0.8397	0.8950	**0.9708**	0.9650
ItalyPowerDemand	0.8853	0.5948	**0.9572**	0.9417
SonyAIBORobotSurface1	0.8120	0.5923	**0.8835**	0.7438
GunPoint	0.9133	0.7533	0.9800	**0.9867**
Coffee	**1.0000**	0.6071	**1.0000**	**1.0000**
CBF	0.4756	0.5911	**1.0000**	0.9811
SyntheticControl	0.6167	0.3933	**0.9867**	**0.9867**
Plane	0.9619	0.7810	**0.9905**	0.9619
BeetleFly	0.6000	**0.9000**	0.8500	**0.9000**
UEA (multivariate)				
BasicMotions	**1.0000**	0.9500	0.9750	0.9750
ERing	0.5704	0.7667	0.8519	**0.9185**
Libras	0.7333	0.6500	**0.8389**	0.6167
RacketSports	0.8224	0.7697	**0.8618**	0.6908
AtrialFibrillation	**0.6000**	0.1333	0.1333	0.2000
NATOPS	0.8167	0.8444	**0.9167**	0.7222
HandMovementDirection	0.2703	0.2568	**0.3649**	0.2568
FingerMovements	**0.5400**	0.5000	0.5000	0.4900
UWaveGestureLibrary	0.5000	0.4219	**0.9031**	0.7250
PenDigits	0.8642	0.6069	**0.9889**	0.9708
UCR mean ($n = 10$)	0.7994	0.6838	**0.9499**	0.9357
UEA mean ($n = 10$)	0.6717	0.5900	**0.7334**	0.6566
Overall mean ($n = 20$)	0.7356	0.6369	**0.8417**	0.7961

Table A.2: KMeans clustering ARI (higher is better) on the test split of each UCR/UEA dataset.

Dataset	FFT	Summary	TS2Vec	MOMENT
UCR (univariate)				
ECG200	0.091	0.121	0.146	**0.257**
Chinatown	0.020	0.164	0.196	**0.839**
ItalyPowerDemand	0.015	**0.018**	0.011	0.000
SonyAIBORobotSurface1	0.008	0.147	**0.792**	0.401
GunPoint	**0.002**	0.002	−0.005	−0.005
Coffee	0.222	0.010	**0.857**	0.492
CBF	0.048	0.295	**0.806**	0.555
SyntheticControl	0.370	0.122	**0.976**	0.621
Plane	0.940	0.565	**0.977**	0.919
BeetleFly	0.113	**0.621**	−0.013	0.212
UEA (multivariate)				
BasicMotions	**0.932**	0.610	0.666	0.871
ERing	0.194	0.455	**0.726**	0.487
Libras	0.283	0.182	0.258	**0.289**
RacketSports	**0.254**	0.082	0.212	0.049
AtrialFibrillation	−0.057	**0.029**	−0.057	−0.009
NATOPS	**0.400**	0.397	0.340	0.149
HandMovementDirection	−0.017	**0.021**	0.016	−0.005
FingerMovements	0.005	**0.010**	−0.003	−0.007
UWaveGestureLibrary	0.250	0.107	**0.382**	0.332
PenDigits	0.246	0.158	**0.683**	0.433
UCR mean ($n = 10$)	0.183	0.206	**0.474**	0.429
UEA mean ($n = 10$)	0.249	0.205	**0.322**	0.259
Overall mean ($n = 20$)	0.216	0.206	**0.398**	0.344

Table A.3: Silhouette score (higher is better) on the test split of each UCR/UEA dataset.

Dataset	FFT	Summary	TS2Vec	MOMENT
UCR (univariate)				
ECG200	0.450	**0.475**	0.205	0.205
Chinatown	0.348	**0.603**	0.492	0.337
ItalyPowerDemand	0.560	**0.650**	0.504	0.404
SonyAIBORobotSurface1	0.232	**0.380**	0.179	0.129
GunPoint	0.647	**0.675**	0.616	0.500
Coffee	0.411	**0.473**	0.323	0.316
CBF	0.267	0.383	**0.395**	0.159
SyntheticControl	**0.353**	0.335	0.253	0.217
Plane	**0.651**	0.437	0.594	0.462
BeetleFly	0.256	**0.435**	0.118	0.168
UEA (multivariate)				
BasicMotions	0.307	0.532	**0.556**	0.276
ERing	0.120	**0.203**	0.176	0.179
Libras	0.254	**0.259**	0.187	0.177
RacketSports	0.153	**0.217**	0.138	0.154
AtrialFibrillation	0.232	**0.461**	0.454	0.174
NATOPS	0.211	**0.295**	0.216	0.101
HandMovementDirection	0.061	0.130	0.021	**0.272**
FingerMovements	0.393	0.406	0.196	**0.546**
UWaveGestureLibrary	**0.177**	0.171	0.110	0.124
PenDigits	0.179	0.234	0.280	**0.333**

Table A.4: Mode Separability Index (MSI, higher is better) on the test split of each UCR/UEA dataset.

Dataset	FFT	Summary	TS2Vec	MOMENT
UCR (univariate)				
ECG200	0.606	**0.728**	0.549	0.621
Chinatown	0.675	1.167	1.241	**1.491**
ItalyPowerDemand	0.647	0.627	**0.733**	0.687
SonyAIBORobotSurface1	0.736	**1.397**	1.004	0.711
GunPoint	0.417	0.421	0.541	**0.646**
Coffee	1.528	0.794	**1.622**	1.378
CBF	0.318	1.677	**1.902**	0.948
SyntheticControl	1.845	1.341	**1.967**	1.561
Plane	**7.491**	5.167	5.062	3.170
BeetleFly	0.626	**2.225**	0.809	0.980
UEA (multivariate)				
BasicMotions	2.343	3.480	**3.644**	1.863
ERing	0.843	1.520	1.326	**1.574**
Libras	1.481	1.374	1.177	**1.552**
RacketSports	0.706	0.747	**0.773**	0.510
AtrialFibrillation	**0.831**	0.794	0.619	0.781
NATOPS	1.479	**2.190**	1.280	0.694
HandMovementDirection	0.323	0.287	**0.349**	0.254
FingerMovements	**0.457**	0.296	0.254	0.069
UWaveGestureLibrary	**1.131**	0.780	0.924	1.117
PenDigits	1.242	1.321	**1.727**	1.492

Table A.5: PCA compactness (lower is better; 2D PCA hull area of mode clusters, in the squared scale of the embedding coordinates). Values span four orders of magnitude across datasets and encoders, so they are reported to 3 significant figures.

Dataset	FFT	Summary	TS2Vec	MOMENT
UCR (univariate)				
ECG200	1.07×10^{3}	2.02	2.47	**0.343**
Chinatown	72.2	0.489	1.72	**0.290**
ItalyPowerDemand	78.0	**1.11**	1.94	1.52
SonyAIBORobotSurface1	493	2.16	1.60	**0.732**
GunPoint	1.08×10^{3}	1.07	4.12	**0.366**
Coffee	138	0.0438	0.114	**0.0296**
CBF	903	1.52	1.01	**0.386**
SyntheticControl	184	0.474	**0.178**	0.370
Plane	67.1	0.0148	0.0207	**0.00502**
BeetleFly	2.49×10^{4}	0.131	0.692	**0.0641**
UEA (multivariate)				
BasicMotions	375	5.06	0.459	**0.126**
ERing	1.09×10^{3}	2.22	1.85	**0.0521**
Libras	618	3.04	3.19	**0.120**
RacketSports	456	38.4	4.42	**1.00**
AtrialFibrillation	1.86×10^{4}	17.4	7.02	**0.147**
NATOPS	5.20×10^{3}	28.2	4.56	**0.106**
HandMovementDirection	9.19×10^{4}	11.4	1.96	**0.620**
FingerMovements	6.33×10^{4}	239	20.1	**1.91**
UWaveGestureLibrary	2.43×10^{4}	3.09	5.44	**0.114**
PenDigits	16.1	**0.498**	1.10	1.90

Table A.6: Centroid stability under the bootstrap proxy (lower is better; mean across-replicate centroid displacement, in the squared scale of the embedding coordinates). MOMENT wins on 20/20 datasets.

Dataset	FFT	Summary	TS2Vec	MOMENT
UCR (univariate)				
ECG200	0.373	0.0152	0.00327	**0.00100**
Chinatown	0.0368	0.00605	0.00102	**0.000412**
ItalyPowerDemand	0.0193	0.00226	0.000685	**0.000444**
SonyAIBORobotSurface1	0.107	0.00365	0.00120	**0.000511**
GunPoint	0.276	0.00918	0.00255	**0.000541**
Coffee	0.387	0.00643	0.00176	**0.000443**
CBF	0.116	0.00385	0.00122	**0.000464**
SyntheticControl	0.235	0.00648	0.00284	**0.00157**
Plane	0.331	0.00317	0.00212	**0.000749**
BeetleFly	5.88	0.0111	0.00830	**0.00137**
UEA (multivariate)				
BasicMotions	0.541	0.0516	0.00800	**0.00192**
ERing	0.221	0.0109	0.00570	**0.000630**
Libras	0.504	0.0653	0.0137	**0.00227**
RacketSports	0.205	0.0517	0.00745	**0.00174**
AtrialFibrillation	6.30	0.207	0.0330	**0.00358**
NATOPS	0.333	0.0298	0.0102	**0.000762**
HandMovementDirection	3.83	0.0403	0.0120	**0.00180**
FingerMovements	0.292	0.0515	0.0107	**0.00172**
UWaveGestureLibrary	1.46	0.0159	0.00793	**0.000635**
PenDigits	0.0167	0.00136	0.00126	**0.000457**