

EDGE AI FOR REAL-TIME PHYSIOLOGICAL STATE DETECTION: A DUAL-USE ARCHITECTURE FOR ENHANCING COMBATANT RESILIENCE

Claudiu VESA, Ghiță BÂRSAN, Bogdan Andrei MĂNESCU

“Nicolae Bălcescu” Land Forces Academy, Sibiu, Romania
vesa.claudiu@armyacademy.ro, gbarsan@armyacademy.ro,
manescu.bogdan@armyacademy.ro

Abstract: Contemporary military operations require rapid, decentralised decision-making based on reliable combatant telemetry, yet cloud-dependent monitoring systems remain vulnerable to electronic warfare and spectrum denial. This paper presents ArmyAI, a proof-of-concept dual-use Edge AI architecture for offline multi-label physiological state detection on commercial off-the-shelf (COTS) microcontrollers. The system is built around a low-power neural processing platform and encodes sensor telemetry as a compact image representation processed by a multi-label Convolutional Neural Network with independent per-state outputs, trained on a combined synthetic and small pilot dataset. The model achieves strong multi-label classification performance on a held-out test set, and its quantised form is compact enough to reside entirely within the constrained flash and activation memory of an embedded NPU. The intended deployment is a local-area network of wearables linked by a low-power LoRa LAN, with planned reporting to an exercise-control team during training events; both are design-stage and have not been implemented or measured on hardware in the present prototype. We discuss the architecture, methodology, results, limitations, and the alignment of the approach with European dual-use funding mechanisms.

Keywords: dual-use technology, Edge AI, wearable physiological monitoring, multi-label classification, STM32N6

1. Introduction

The modern operational environment is defined by its unpredictability, often described in terms of the VUCA (Volatility, Uncertainty, Complexity, Ambiguity) [1] and BANI (Brittleness, Anxiety, Nonlinearity, Incomprehensibility) paradigms. Under these conditions, the physical and cognitive load placed on military combatants, first responders, and heavy industrial workers regularly exceeds nominal limits, and timely awareness of an individual's physiological capacity is a key enabler of sustained force effectiveness and casualty prevention [2]. A growing body of work argues that accessible, commercial edge-computing technology can be adapted to enhance both military readiness and

civilian resilience [3], and that physiological monitoring on the wearer is a central building block of such systems.

Cloud-dependent monitoring architectures are nevertheless poorly suited to contested environments: they expose continuous radio links to electronic warfare, jamming and spectrum denial, and they introduce remote points of failure that defeat the purpose of decentralised situational awareness. Edge AI — inference performed on the wearable device itself — is therefore an attractive alternative, but it is constrained by tight memory, power, and latency budgets that have historically limited the complexity of usable models. Within this gap, two practical questions remain open for small-team military and dual-use applications. First,

how can multi-label physiological state classification be expressed within the operating envelope of a single COTS microcontroller with an embedded neural accelerator. Second, how can such a wearable be integrated into a small dismounted team without re-introducing the cloud and Wi-Fi dependencies the design was meant to avoid.

This paper presents ArmyAI, a proof-of-concept Edge AI architecture that addresses both questions on the STM32N6570-DK platform and its Neural-ART NPU [4]. A NodeMCU board is used as a low-cost helper interface to physiological sensors over PC; its Wi-Fi radio is disabled and the board acts only as a sensor aggregator that forwards an ASCII payload over USART2 to the STM32N6 application core. On the STM32N6 the payload is encoded as a 16×128 grayscale image and classified by an unpooled multi-label Convolutional Neural Network with eight independent sigmoid outputs, with results rendered on a local TouchGFX dashboard inside a ThreadX RTOS [3]. The intended deployment is a Local Area Network of 8–12 wearables linked by a low-power LoRa LAN; alert-level traffic only is shared between nodes, and future iterations are planned to forward summary state to an exercise-control team during training events. A hardware-backed secure boot chain and memory-region isolation [5, 6] provide the underlying execution foundation.

The contributions of this paper are: (i) a complete COTS-based offline architecture for wearable multi-label physiological state detection on STM32N6 hardware; (ii) a corrected accuracy figure of 93% on a held-out test split, replacing an earlier figure obtained on the training distribution; (iii) a description of how the wearable integrates into a small team via a LoRa LAN and how it is planned to report to an exercise-control element in the future; and (iv) an explicit discussion of limitations, including the role of fixed-threshold pre-labelling, the prototype use of NodeMCU as a helper board, and the absence of measured

hardware-level power and latency figures. The remainder of the paper is organised as follows. Section 2 reviews related work in Edge AI, TinyML, wearable physiological monitoring, and multi-label classification. Section 3 describes the system architecture. Section 4 details the methods, including the dataset and model. Section 5 presents results. Section 6 discusses limitations and operational relevance. Section 7 concludes.

2. Related Work

2.1. Edge AI and TinyML on Constrained Hardware

Surveys of TinyML and on-device deep learning highlight three pressure points: model footprint relative to available RAM, inference latency on fixed-frequency cores, and energy per inference on battery-operated nodes [7, 8, 9]. Empirical work on activity-recognition wearables further shows that simpler architectures (e.g. MLPs) can match deeper CNNs in accuracy while consuming an order of magnitude less memory and energy [10]. Quantisation to int8, structured pruning, and neural-accelerator co-design are now standard, with vendor toolchains such as ST Edge AI Core , and dedicated NPUs such as the STM32N6 Neural-ART [5] bringing convolutional workloads into the milliwatt envelope of wearables. Laboratory benchmarks, however, do not always translate to operational use, motivating explicit reporting of both model-side (parameters, MACC, memory) and device-side (current draw, latency) metrics.

2.2. Wearable Physiological Monitoring and Multi-Label Classification

Wearable physiological monitoring has a long lineage in civilian sports science and military human-performance research [11, 12]. Classical pipelines couple multi-modal sensing — heart rate, skin temperature, electrodermal activity, accelerometry — with feature engineering and shallow classifiers such as SVMs [13, 14]. Recent work moves toward end-to-end deep models on the same modalities: wrist-PPG fed into hybrid CNNs for stress recognition [15], multi-modal CNNs combining time- and

frequency-domain features [16], and int8-quantised TinyML stress classifiers on wearable MCUs [17]. A 2021–2025 systematic review confirms multimodal fusion with hybrid deep models as the de facto state of the art, while highlighting persistent validation and generalisation gaps [18]. A recurring methodological observation is that physiological stress is rarely a single mutually exclusive label — heat strain, fatigue, dehydration, and cognitive load co-occur — which motivates multi-label formulations with per-label sigmoid outputs, evaluated by F1, precision, recall and Hamming loss rather than top-1 accuracy [19, 20]. Ethical and governance considerations — consent, retention, bias — are increasingly first-class concerns in this literature [21].

2.3. Dual-Use Framing and the Vulnerability of Cloud-Dependent Architectures

Conventional IoT deployments stream raw telemetry to centralised cloud back-ends, exposing the system to electronic warfare, interception, and spectrum denial. Civilian-developed edge architectures — hardened by removing the radio dependency altogether — are a defensible substrate for both civilian welfare and security objectives, and are conceptually aligned with the dual-use eligibility criteria of EU programmes such as Horizon Europe and the European Defence Fund. ArmyAI is positioned within this framing: the same architecture supports civilian heavy-industry monitoring and military situational awareness without proprietary defence-classified components [21].

3. System Architecture and Hardware Foundation

The ArmyAI end-to-end data flow on a single wearable operates without dependency on external Wi-Fi or cellular

networks. Inter-wearable and rear-link communication is constrained to a dedicated low-power LoRa LAN, described in section 3.3. The architecture is intentionally separated into four planes: a sensor-acquisition plane, an inference plane, a presentation plane, and a tactical-communication plane.

3.1. Hardware Ingestion and Telemetry Piping

At the sensor periphery, a NodeMCU (ESP8266) board serves as a low-cost helper interface to physiological and environmental sensors over I²C. Its on-board Wi-Fi radio is intentionally disabled in firmware: the board is retained as a sensor aggregator and serial bridge only, not as a wireless transceiver. Aggregated readings are framed as an ASCII payload and forwarded over a wired serial link to the STM32N6 application core. In the present prototype, the SHT31D temperature/humidity sensor is the only physical channel populated; the remaining six non-baseline states are populated by synthetic data only (section 4.1). Wiring additional channels — pulse oximetry, accelerometry, EDA — is identified as future work in section 7. The use of NodeMCU in this role is a prototype convenience; the limitations and the planned migration to a lower-power MCU are discussed in section 6.

3.2. Secure Processing Loop

The STM32N6 parses this payload on its application core [4]. The user-interface framework runs on a ~60 Hz refresh, and the application core converts the incoming sensor arrays into a 16×128 grayscale buffer using fixed thresholds. We note explicitly that this thresholding step is a deterministic preprocessing layer, not a learned inference step; its role in the overall pipeline and its consequences for the interpretation of the CNN results are revisited in section 4 and section 6.

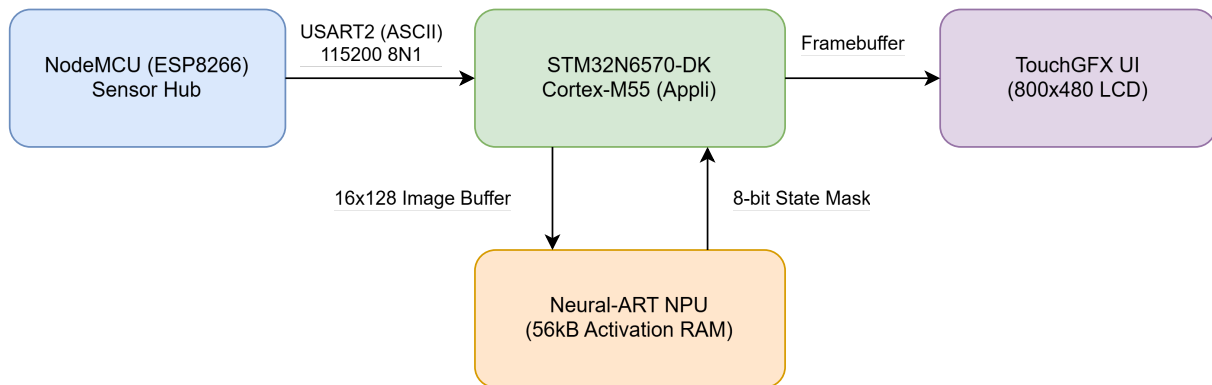


Figure 1: System Architecture – end-to-end data flow from sensor hub to TouchGFX display via the Neural-ART NPU

The Neural-ART NPU receives a pointer to this buffer and performs int8 inference in parallel with the Cortex-M55. The output is an eight-element logit vector, converted to eight independent binary flags by a per-element sigmoid threshold (see section 4.3). Flags are rendered on the on-board 800×480 LCD via TouchGFX, and forwarded to the tactical-communication plane described next.

3.3. Tactical Communications over a LoRa LAN

ArmyAI's intended deployment is a Local Area Network of 8–12 wearables — the typical size of a dismounted small unit — connected by a low-power LoRa LAN rather than by Wi-Fi or cellular links. The LoRa LAN described below is a design-stage topology and has not been measured on hardware in the present prototype. LoRa was selected for three reasons. First, its long range at low transmit power fits the energy budget of a wearable battery. Second, the LAN carries only alert-level events (the eight flags plus a wearer identifier and timestamp), not raw telemetry, keeping airtime and detectability low. Third, the radio parameters can be tuned to the operational regulatory environment in which the team is exercised. The use of LoRa for short-range, infrastructure-agnostic tactical links is consistent with recent work on multi-hop and peer-to-peer wearable configurations [22, 23].

Within the LAN, one node is configured as a

team aggregator that receives the alert frames from the other 7–11 wearables and renders the team picture on its own TouchGFX dashboard. The aggregator role is software-defined and can be reassigned. No node in the LAN forwards traffic outside the LoRa segment by default; the architecture intentionally avoids any direct bridge to public networks.

3.4. Data Logging and Future Exercise-Control Reporting

Each wearable maintains a local log of (i) the eight sigmoid flags per inference cycle and (ii) a down sampled summary of the raw sensor stream. The log is written to on-board non-volatile storage with a rolling-buffer policy sized for a full exercise duration. Logs are not transmitted in real time; they are retrieved post-mission through a wired USB or serial docking interface, providing an air-gapped path for post-action analysis and decoupling the wearable's operational radio profile from its analytical value.

The exercise-control reporting channel described next is explicitly out of scope for the present proof of concept and is design-stage only; it is described here to make the deployment topology explicit for future work. A future iteration of the platform is planned to extend the LoRa LAN with an exercise-control reporting channel: a single LoRa gateway in the rear area subscribes to the alert frames produced inside the team LAN and aggregates them into a near-real-time picture for the exercise-control team

during training events. This reporting channel is read-only from the wearables' perspective and alert-only in payload.

3.5. Threat Model and Security Boundary

The architecture targets a constrained threat model rather than claiming general resilience. The wearable is assumed to operate in environments subject to jamming, passive RF interception, and adversarial spoofing of the LoRa channel. In scope: (i) confidentiality and integrity of LoRa alert frames, addressed by per-team symmetric keying and a monotonic frame counter for anti-replay; (ii) integrity of the on-device firmware, addressed by a hardware-backed secure boot chain and memory-region isolation [5, 6]; and (iii) integrity of the post-mission log, addressed by signed log records before docking transfer. Out of scope for the present prototype: physical capture of an unlocked device, side-channel attacks on the NPU, supply-chain compromise of the STMicroelectronics toolchain, and adversarial-ML attacks against the deployed CNN. These are noted as future-work targets in section 6.

4. Methods

This section describes the dataset used to develop the prototype, the encoding that turns sensor readings into the 16×128 input image, the CNN architecture and training procedure, and the two baselines (SVM and Softmax-CNN) against which it was compared. We make explicit which steps are deterministic rule-based preprocessing and which are learned, so that the role of the neural network in the pipeline is not over-claimed.

4.1. Dataset and Labels

Because the present prototype targets a proof of concept rather than a deployed clinical system, the dataset used for development was generated under controlled conditions and is composed of two parts: (i) a synthetic component, in which each of the eight target states is sampled from a parameterised distribution covering its expected sensor range, and combinations of states are drawn from a uniform distribution over the 2^8

possible label vectors; and (ii) a small pilot component collected from a single instrumented volunteer wearing the sensor harness during short, scripted exertion sequences (warm-up, mild exertion, peak exertion, recovery). The pilot component is used exclusively as a sanity-check held-out set; it is not used for training. Sample counts, sampling protocol, and the precise mapping from raw sensor values to the eight states are reported in full in Appendix A and in the supporting code repository.

Each example carries an eight-element binary label vector, one bit per target state: high body temperature, dehydration, elevated heart rate, muscular fatigue, environmental heat load, low-oxygen / altitude stress, cognitive overload, and baseline normal. The labels are not mutually exclusive, which motivates the multi-label formulation used in section 4.3.

We deliberately flag a methodological limitation here rather than in the conclusion: the synthetic component is generated from the same threshold-based logic that the firmware later applies as preprocessing (section 3.2). Training a CNN on labels produced this way verifies that the network can recover the encoding, but it does not by itself demonstrate physiological inference from raw signals. The accuracy figure reported in section 5 must be read against this caveat, and section 6 (Discussion) returns to it explicitly. The single-volunteer pilot data was collected from an author of this paper; no third-party human subjects were involved, and the use is bounded to the sanity-check role described above. A multi-subject IRB-approved study is identified as the principal data-side future work in section 7.

4.2. Sensor-to-Image Encoding

Sensor values are mapped to a 16-pixel by 128-pixel grayscale image partitioned into eight 16×16 column blocks, one per target state (columns 0–15 = state 0, columns 112–127 = state 7). Within each block, the pixel intensity encodes the level at which the corresponding state's threshold is exceeded. This encoding preserves spatial separation between states and is what motivated the

unpooled architecture in section 4.3. Because the thresholds used to generate the synthetic labels are identical to those used by the firmware preprocessing layer (section 3.2), the CNN's task in this prototype is to recover a known encoding, not to perform physiological inference from raw signals. The implication for the result in section 5.1 is revisited in section 6.1.

4.3. Model Architecture and Training

The CNN is implemented in PyTorch [24] as three convolutional blocks with ReLU activations and no pooling layers, followed by a fully-connected head producing eight logits. Eliminating global pooling preserves the spatial coordinates of each state's 16×16 block through to classification. The eight logits are passed through an element-wise sigmoid to produce eight independent probabilities, trained with a binary cross-entropy loss applied independently per output.

The model was trained on a GPU workstation, with the dataset split 70/15/15 into training, validation, and held-out test, stratified so each of the eight labels appears in each split. To avoid the geometry-corrupting issue identified during the Softmax-CNN attempt, no resize, crop, or rotation augmentations were applied; only additive Gaussian noise on the pixel intensities was used.

4.4. Baselines

We compared the proposed unpooled multi-label CNN against two baselines drawn from earlier iterations of the prototype. Each baseline is described together with the reason it was discarded; the quantitative comparison is reported in section 5.

4.4.1. SVM Baseline

An initial baseline used a one-vs-rest Support Vector Machine over a flattened version of the input. It has very low computational overhead but no spatial inductive bias, and on combined-stress examples (e.g. heat plus muscular fatigue) the per-label classifiers showed an elevated false-positive rate. The SVM is retained as a baseline rather than as the deployed model.

4.4.2. Softmax-CNN Baseline

A second baseline used a CNN with a Softmax head over every possible combination of the seven non-baseline states plus an exclusive “normal” class. It plateaued at approximately 52% validation accuracy for three reasons: global pooling before classification destroyed the spatial separation between state blocks; the Softmax head forced overlapping multi-state combinations to compete for the same probability mass rather than co-activating; and standard image-augmentation pipelines distorted the deliberately structured input geometry. These failures motivated the unpooled, multi-label design adopted in section 4.3.

5. Results

We report two classes of results: classification performance against the two baselines (section 5.1), and an embedded-deployment profile on the STM32N6570-DK (section 5.2). All numbers are obtained on the held-out test split described in section 4.3; hardware values are taken from the STM32N6 datasheet at the configuration used by the prototype and are explicitly labelled as datasheet-typical rather than measured (a measured-bench study is identified as future work in section 7).

5.1. Classification Performance

On the held-out test split (15% of examples, drawn from the synthetic component only the pilot component is reserved as an independent sanity check), the proposed unpooled multi-label CNN achieves 93% subset accuracy (exact-match on the eight-bit label vector). The SVM baseline reaches 71% subset accuracy on the same split, dominated by false positives on combined-stress examples; the Softmax-CNN baseline reaches 52%, consistent with the architectural causes identified in section 4.4.2. We emphasise two points. First, subset accuracy is a coarse multi-label metric: per-label precision, recall, F1, and Hamming loss are the right next instruments and are reported as future work (section 7) once the pilot component is expanded. If the

conference page budget allows, per-label precision, recall, F1, and Hamming loss on the same test split will be added in a final version; in the present submission only subset accuracy is reported. Second, the 93% figure is the recoverability of a known deterministic encoding; it is not a measurement of physiological-state inference from raw signals, and we return to this distinction in section 6. We note for transparency that an earlier internal evaluation of this network reported approximately 100% accuracy; that figure was obtained on the training distribution and is superseded by the 93% held-out figure

reported above.

5.2. Embedded Deployment Profile

After quantisation to int8 with STEdgeAI Core v4 [6], the deployed model occupies 1.125 MB of constant weights in external flash and 56.0 kB of runtime activations in on-chip NPU memory. The network requires approximately 294,700 parameters and 1.54 million multiply-accumulate operations per inference. The companion Figure 2 illustrates the pipeline from the 16×128 monochrome input through the three Conv2D blocks to the eight independent sigmoid outputs.

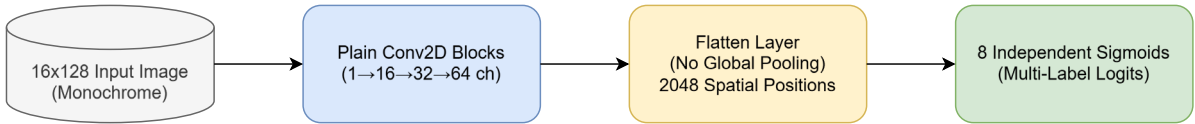


Figure 2: AI Pipeline Architecture 16×128 monochrome input through Conv2D blocks to 8 independent sigmoid outputs

Datasheet-typical operating points for the platform indicate single-digit millisecond inference for an int8 graph of this size, and a wearable-class power envelope in the active inference state with deeper sleep modes available between cycles. These are reference values, not bench-measured values

for the present prototype; in-system measurement of inference latency, current draw, end-to-end sensor-to-display latency, and estimated battery life is identified as an explicit future-work item in section 7.

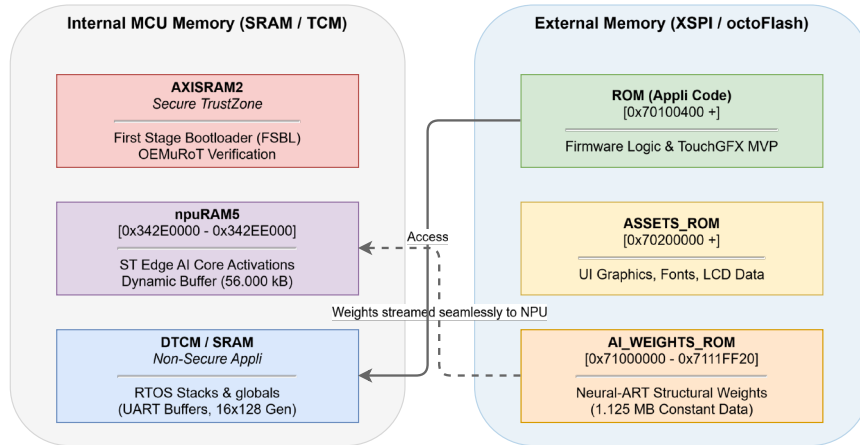


Figure 3: Memory Map Architecture – internal MCU memory (SRAM/TCM) and external memory (XSPI/octoFlash) region

6. Discussion

6.1. Interpreting the Accuracy Figure

The 93% subset accuracy reported in section 5.1 is a measurement of one specific question: given a 16×128 image produced by

the deterministic threshold encoder of section 3.2, can the CNN recover the eight-bit label vector that the encoder represents? On that question the network performs well, and substantially better than the SVM and

Softmax-CNN baselines drawn from earlier iterations of the prototype. It is not a measurement of physiological-state inference from raw signals, because the encoder collapses raw signals into discrete states before the network sees them. Reading the network as a physiological classifier in the present configuration would conflate two different claims; we deliberately separate them. We note that comparable Edge-AI biomedical studies — for example, the int8-quantised ECG anomaly detector of (92.3% accuracy at 0.024 mW on a Raspberry Pi/Arduino-class device) and the WESAD-trained TinyML PPG stress classifier of [17] — operate on raw or lightly filtered signals rather than on a deterministic encoder, which we consider the appropriate target for ArmyAI's next iteration.

A direct consequence is the path forward for the algorithm. The deterministic encoder is appropriate for the proof-of-concept role of the present platform — it gives a small, repeatable, fully-defined target that lets us verify the end-to-end firmware integration, the NPU deployment, and the TouchGFX UI on real hardware. To make the contribution a physiological one, the encoder must be replaced (or bypassed) with a learned front-end operating on continuous sensor streams, trained on a real human-subject dataset rather than a parameterised synthetic one. We treat that as the principal scientific extension of this work rather than a tuning detail.

6.2. Hardware and System Choices

The use of an ESP8266-based NodeMCU as the peripheral sensor hub deserves explicit comment, because the choice is counter-intuitive in a system that intentionally has no operational Wi-Fi link. In the present prototype the module is used purely as an I²C-to-UART helper: the Wi-Fi radio is held disabled in firmware, the module runs from the wearable's regulated rail rather than an independent supply, and only the sensor-reading and USART code paths are exercised. The reason to retain it for the prototype is availability — the NodeMCU exposes I²C, can be flashed from the same

toolchain as the rest of the platform, and is cheap enough to be replicated across a team of 8–12 wearables without procurement friction. The reason to replace it in any future revision is energy: a dedicated low-power MCU (for example an STM32L0/L4-class part, or a sensor-side coprocessor on the same N6 family) would remove the standby cost of an unused Wi-Fi radio. A natural replacement is an STM32L0-class MCU in a single-package SiP with the I²C sensor bus, removing the ~70 mW standby cost of the unused ESP8266 Wi-Fi radio. We list this swap as a hardware-side future-work item.

6.3. Limitations and Threats to Validity

The principal limitations of the present study are: (i) the dataset is dominated by a synthetic component generated from the same thresholds the firmware applies as preprocessing, and the pilot human-subject component is small and used only as a sanity check; (ii) the reported classification figure is subset accuracy on that synthetic test split, and the multi-label metrics that the literature treats as standard (per-label precision, recall, F1, Hamming loss) are not yet reported; (iii) the hardware values reported in section 5.2 are datasheet-typical operating points for the STM32N6570-DK, not in-system bench measurements of the prototype under inference load; (iv) the LoRa LAN and the exercise-control reporting channel described in section 3.3 and section 3.4 are design-stage architectural choices and not validated radio measurements in the operational band; (v) the threat model in section 3.5 is intentionally narrow, and explicitly excludes adversarial-ML attacks on the deployed CNN, side-channel attacks on the NPU, supply-chain compromise of the toolchain, and physical capture of an unlocked device. Generalisation of any claim made here to operational deployment requires the bench, field, and human-subject studies listed in section 7.

7. Conclusion and Future Work

This paper has presented a proof-of-concept Edge AI architecture, ArmyAI, for offline physiological-alert generation on

STM32N6-class hardware. We described the LoRa-LAN topology over which a team of 8–12 wearables exchanges alert-level events without external network dependencies, the local data-logging plane that retains telemetry for post-mission analysis, and the planned exercise-control reporting channel that will surface team state to an exercise-control team in future training events. We described the encoding that turns sensor readings into a 16×128 image, the unpooled multi-label CNN trained against it, and the deployment of that network to the Neural-ART NPU at 1.125 MB of weights and 56.0 kB of activations. On the held-out test split the network reaches 93% subset accuracy against 71% for the SVM baseline and 52% for the Softmax-CNN baseline drawn from earlier iterations of the prototype. We have been explicit about what this figure does and does not measure (section 6.1), about the limitations of the dataset and of the prototype hardware envelope (section 6.3), and about the threats that the architecture is

and is not designed to address (section 3.5). The principal next steps are: (a) collecting a real human-subject dataset and replacing the threshold-based encoder with a learned front-end that consumes raw sensor streams, so that the network is asked the physiological question rather than the recoverability question; (b) reporting per-label precision, recall, F1, and Hamming loss on that dataset in place of subset accuracy; (c) bench-measuring inference latency, idle and active current draw, end-to-end sensor-to-display latency, and battery life on the prototype under representative load; (d) implementing and field-testing the LoRa LAN at team scale and the exercise-control reporting channel; and (e) replacing the NodeMCU sensor hub with a low-power MCU appropriate for an extended-mission wearable. The platform described here is intended as the foundation on which those measurements can be made, not as a finished operational system.

References

- [1] N. Bennett and G. J. Lemoine, “What VUCA Really Means for You,” *Harvard Business Review*, vol. 92, no. 1/2, p. 27, 2014.
- [2] R. Siegfried, “NATO Modelling & Simulation Group Contributions to Implementing NATO S&T Strategy,” in *CA2X2 Forum 2020 Papers Collection*, NATO Modelling & Simulation Centre of Excellence, 2020.
- [3] K. E. Friedl, “Military Applications of Soldier Physiological Monitoring,” *Journal of Science and Medicine in Sport*, vol. 21, no. 11, pp. 1147–1153, 2018, doi: 10.1016/j.jsams.2018.06.004.
- [4] STMicroelectronics, “STM32N6 Series Data Brief and Reference Manuals,” 2024. [Online]. Available: <https://www.st.com>
- [5] STMicroelectronics, “Secure Programming Using STM32CubeProgrammer.” [Online]. Available: https://www.st.com/resource/en/user_manual/dm00403500.pdf.
- [6] STMicroelectronics, “BootRom for STM32N6.” [Online]. Available: https://wiki.st.com/stm32mcu/wiki/Security:BootRom_for_STM32N6.
- [7] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. Sebastopol, CA, USA: O’Reilly Media, 2019.
- [8] C. Banbury, V. J. Reddi, M. Lam et al., “MLPerf Tiny Benchmark,” in *Proc. NeurIPS Datasets and Benchmarks Track*, 2021, doi: 10.48550/arXiv.2106.07597.

- [9] T.-D. Nguyen, T.-V. Nguyen, T.-P. Nguyen, and T.-T. Pham, "TinyML Applications in Wearable Devices: A Systematic Review and Research Directions," in *Advances in Information and Communication Technology (ICTA 2025)*, Lecture Notes in Networks and Systems, vol. 1831. Cham, Switzerland: Springer, 2026, pp. 194–203, doi: 10.1007/978-3-032-18316-3_22.
- [10] E. Lattanzi, M. Donati, and V. Freschi, "Exploring Artificial Neural Networks Efficiency in Tiny Wearable Devices for Human Activity Recognition," *Sensors*, vol. 22, no. 7, Art. no. 2637, 2022, doi: 10.3390/s22072637.
- [11] R. David, J. Duke, A. Jain et al., "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems," *Proc. Mach. Learn. Syst. (MLSys)*, vol. 3, pp. 800–811, 2021.
- [12] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020, doi: 10.1109/JPROC.2020.2976475.
- [13] U. B. Baloglu, M. Talo, O. Yildirim et al., "Classification of Myocardial Infarction with Multi-Lead ECG Signals and Deep CNN," *Pattern Recognition Letters*, vol. 122, pp. 23–30, 2019, doi: 10.1016/j.patrec.2019.02.016.
- [14] P. Bobade and M. Vani, "Stress Detection with Machine Learning and Deep Learning Using Multimodal Physiological Data," in *Proc. 2nd Int. Conf. on Inventive Research in Computing Applications (ICIRCA)*, IEEE, 2020, pp. 51–57, doi: 10.1109/ICIRCA48905.2020.9183244.
- [15] N. Rashid, L. Chen, M. Dautta, A. Jimenez, P. Tseng, and M. A. Al Faruque, "Feature Augmented Hybrid CNN for Stress Recognition Using Wrist-Based Photoplethysmography Sensor," in *Proc. 43rd Annual Int. Conf. of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 2021, doi: 10.1109/EMBC46164.2021.9630576.
- [16] J.-Z. Xiang, Q.-Y. Wang, Z.-B. Fang, J. A. Esquivel, and Z.-X. Su, "A Multi-Modal Deep Learning Approach for Stress Detection Using Physiological Signals: Integrating Time and Frequency Domain Features," *Frontiers in Physiology*, vol. 16, Art. no. 1584299, 2025, doi: 10.3389/fphys.2025.1584299.
- [17] P. Ganesan, Y. R. Thota, H. Shehata, and T. Nikoubin, "TinyML Based Stress Detection Utilizing PPG Signals: A Lightweight Approach for Smart Wearable Devices," in *Proc. Great Lakes Symp. on VLSI (GLSVLSI '25)*, ACM, 2025, pp. 941–946, doi: 10.1145/3716368.3735274.
- [18] Pannavira and A. Hermawan, "Multimodal Wearable-Based Stress Detection Using Machine Learning: A Systematic Review of Validation Protocols and Generalization Gaps (2021–2025)," *Ultimatics: Jurnal Teknik Informatika*, vol. 17, no. 2, pp. 203–213, 2025, doi: 10.31937/ti.v17i2.4488.
- [19] B. Bent, B. A. Goldstein, W. A. Kibbe, and J. P. Dunn, "Investigating Sources of Inaccuracy in Wearable Optical Heart Rate Sensors," *npj Digital Medicine*, vol. 3, Art. no. 18, 2020, doi: 10.1038/s41746-020-0226-6.
- [20] M. J. Buller, W. J. Tharion, S. N. Cheuvront et al., "Estimation of Human Core Temperature from Sequential Heart Rate Observations," *Physiological Measurement*, vol. 34, no. 7, pp. 781–798, 2013, doi: 10.1088/0967-3334/34/7/781.
- [21] M. G. Hanna, L. Pantanowitz, B. Jackson, O. Palmer, S. Visweswaran, J. Pantanowitz, M. Deebajah, and H. H. Rashidi, "Ethical and Bias Considerations in Artificial Intelligence/Machine Learning," *Modern Pathology*, vol. 38, no. 3, Art. no. 100686, 2025, doi: 10.1016/j.modpat.2024.100686.
- [22] J. R. Cotrim and J. H. Kleinschmidt, "LoRaWAN Mesh Networks: A Review and Classification of Multihop Communication," *Sensors*, vol. 20, no. 15, Art. no. 4273, 2020, doi: 10.3390/s20154273.

- [23] A. K. Agrawal, S. Das, S. De, and J. Kumar, “Modeling and Link Budget Feasibility Analysis of Secure LoRa-Based Peer-to-Peer Communication for Short-Range Tactical Networks,” arXiv preprint arXiv:2602.23924, 2026, doi: 10.48550/arXiv.2602.23924.
- [24] A. Paszke, S. Gross, F. Massa et al., “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in Advances in Neural Information Processing Systems (NeurIPS 32), 2019, pp. 8024–8035.