ISSN: 2993-8643

Research Article

Engineering: Open Access

IEEE COINS 2023 Contest for In Sensor Machine Learning Computing

Andrey Korobitsyn[1], Danil Zherebtsov[2]*, Danilo Pau[3], Marco Bianco[4], Dmitriy Proshin[5]

[1]*Neuton.ai San Jose, CA, USA*

[2]*Principal AI engineer, Neuton.ai Dubai, UAE*

[3]*System Research and Applications STMicroelectronics Agrate Brianza, Italy*

[4]*MEMs Product Group STMicroelectronics Cornaredo, Italy*

[5]*Head of Machine Learning Neuton.ai Dubai, UAE*

***Corresponding Author**
Danil Zherebtsov, Principal AI engineer, Neuton.ai Dubai, UAE.

Submitted: 2023, Nov 01; **Accepted:** 2023, Dec 18; **Published:** 2023, Dec 20

Citation: Korobitsyn, A., Zherebtsov, D., Pau, D., Bianco, M., Proshin, D. (2023). IEEE COINS 2023 Contest for In Sensor Machine Learning Computing. *Eng OA, 1*(4), 236-242.

Abstract

The purpose of the investigation is to analyze the effect of fin length and position in terms of rotational angle on heat transmission and entropy generation. Different parameters such as Prandtl numbers, Hartmann numbers, Rayleigh numbers, and particle volume fractions are used to analyze nanofluid laminar flow behavior and temperature distribution. The fin has a significant impact on both the isotherm and the streamlines. Findings revealed that increasing the rotational angle of a spinning heat exchanger might result in more consistent temperature distribution along isotherms; larger fins, on the other hand, frequently provide greater heat dissipation due to increased surface area. Furthermore, when Rayleigh numbers increase, so does the temperature distribution between the fins and the surrounding fluid. The presence of a magnetic field affects fluid dynamics and contributes to the generation of entropy. Higher Prandtl numbers can result in the enhancement heat transfer phenomena and the generation of entropy.

Keywords: machine learning, intelligent sensor processing unit, neuton.ai, micro-controllers, neural network, digital signal processing, footprint, human activity recognition.

I. Introduction

TinyML1 community, started in 2019, quickly became a fast-growing field of machine learning (ML) technologies and applications. It includes hardware, algorithms, end to end workflows and software capable of performing on-device sensor data analytics at extremely low power, typically in the mW range and below, and hence enabling a variety of always-on use-cases and targeting battery operated devices.

More and more practical applications were developed till nowadays with extensive adoption of ML models for operation in resource constrained context, such as the one defined by the TinyML community. However, a hand-crafted approach to the development of these applications is challenged by the need to increase the level of automation, productivity, and interoperability.

Focused on providing ground-breaking contributions to these needs, the Neuton.ai team has developed a fully automated toolchain to devise proprietary neural network (NN) architectures designed specifically for TinyML applications. Such a toolchain provides the grounds for productive and widespread adoption of intelligence on ultra-tiny edge processors, starting from the availability of a dataset. Such a model featuring a limited number of parameters makes it possible to inference the automatically devised model directly on the sensor. This paper refers to the latter as integrating in its package ML computing capabilities limiting or without having to rely on the host processors (e.g., such as a micro-controller MCU or multi-processor unit MPU).

It's widely known that impressive developments had been achieved by the ML community with hand crafted NN models, however their extensive application in the upmost resource constrained devices is limited due to the nature of their underlying topologies and associated model sizes, specifically the fact, that an architecture of a conventional NN is predefined before the actual training process take place. Whether it will turn out to be accurate or not, its model footprint is already known at design time. This

requires several time-consuming iterations between model design and accuracy characterization before a deployable ML solution could be signed-off and embedded on a tiny device.

Quite differently, Neuton.ai workflow will automatically grow an ML topology starting from a single neuron. Through constant cross-validation, Neuton.ai tool automatically decides if additional nodes shall be added to the NN model during training. This allows the ML engineers to automatically build an optimal model with a single iteration (without adopting complex Neural Architecture Search (NAS) approaches) with each node providing upmost value for predicting the outcome.

The proposed Neuton.ai approach allows to train NNs without starting from a pre-cooked NN topology and thus without requiring either NAS or multi-year ML engineering know-how and experiences. Resulting models do not require compression and are on average an order of magnitude smaller in model size compared to hand-crafted NN models built with popular deep learning frameworks. Contrary to this traditional approach requires a machine learning engineer to define various neural network architectures, sets of hyperparameters and iterate over these settings with multiple runs. When a satisfactory result is achieved, final model undergoes compression procedures. After model compression additional validation is necessary to confirm that the model did not lose predictive power. Finally, the compressed model is ready to be deployed to the target edge device.

This significantly simplify the complexity to run experiments, increases the productivity, and make faster the time-to-market development of innovative solutions which enables developers of TinyML applications to: focus their best energy on creating break-through applications and quickly test their application hypothesis to develop production-grade models at an unprecedented pace. The advantage in Neuton.ai generated NN's footprint allows to inference the trained model directly using the very limited memory of single package sensor with built-in ML computing assets without having to utilize any resources of the host processor. This results in a significant energy efficiency (at µW level) because the host processor is woken up only if a more powerful task needs to be executed by leveraging its less limited assets in terms of memory and computational capabilities.

Examining the achieved results by the challenge organized at the 2023 IEEE COINS conference, this paper encourages the Tiny ML engineers to continue research in NN architectures, their applications and optimization of inference workloads to let advance development of intelligence at the edge.

The paper is organized as follows: section II defines the problem behind the IEEE COINS challenge set among teams; section III introduces the case study; section IV provides an overview of the target hardware being used; section V provides an overview of the target software being used; section VI describes the protocol for data acquisition; section VII lists the competing teams; section VIII reports the results achieved by each competing team; section IX reports the final scores and discusses them; section X concludes the paper summarizing the essence of challenge.

II. Problem Definition

The core of the contest was to embed an advanced machine learning (ML) model (e.g., a neural network, NN) on the tiniest possible computing hardware, with reference to the ML built-in capabilities into the inertial (single package) sensor chip.

To address this problem, STMicroelectronics (ST) in collaboration with Neuton.ai and IEEE COINS Conference chair, had organized a competition between applied teams and at the IEEE COINS 2023 conference.

This was meant to enable various engineers to compete among them in creating and proving that a ML model capable was able to recognize various human activities (HAR) by ingesting inertial sensor data and embedding the final NN model into the ST 6-axes inertia unit (IMU) sensor. The latter integrated an Intelligent Sensor Processing Unit (ISPU) with the following resource constraints:

1) Limited program memory (e.g. 32 KiB)
2) Limited data memory (e.g. 8 KiB)
3) Up to 10 MHz operating frequency

ST had provided sensor hardware and associated software while Neuton.ai provided access to its Automated Tiny ML creation and training Platform for data transformation and model training. Ultimately the teams were measured on if were able to take advantage of an end-to-end workflow that could make their operations more productive. If that could be proven by them, the main expectation of such a challenge would have been fulfilled at the sole benefit to the ML engineering community.

III. Case-Study Overview

The case study targeted the capability of recognizing various HAR by acquiring raw inertial sensor data and making them available to the inference running on the sensor.

The competing teams were required to identify several classes (known as activities). The number of them a single NN model had to recognize impacted the final teams' score. The higher the number of classes & classification accuracy was – the higher the score was.

Teams were required to collect enough data representing each activity. The number and variability of subjects taking part in data collection had to be enough for creating diverse samples out of training data statistic to test model capability to generalize.

The activities included Walking, Jumping, Exercises, Manual equipment operation, Daily routines, etc. Result assessment had been carried out by an ad-hoc committee set at IEEE COINS 2023 conference and including representatives from ST and Neuton.ai. Final score was based on the following criteria:

1. Number of recognized classes (*NCn*)
2. Hold-out balanced accuracy (*Ah*)
3. Sensor inference latency (*Ln*)
4. Sensor data RAM occupation (*Mdn*)

5. Sensor program RAM occupation (*Mpn*)
6. Realtime demo (*RT*)

The total score was defined in eq (1) as per the following criteria:

- Number of recognized classes (NCn)
- Neuton holdout balanced accuracy (Ah)
- ISPU inference latency (Ln)
- ISPU data RAM occupation (Mdn)
- ISPU program RAM occupation (Mpn)
- Realtime demo (RT)

Score = (200*NCn) + (100*Ah) + (80*Ln) + (100*Mdn) + (50*Mpn) + (100*RT) (eq. 1)

III. Hardware Overview

The hardware which was required to perform the challenge was composed of the following components:

- Nucleo STM32L476RG development board [1]
- X-NUCLEO-IKS01A3 sensor expansion board [2]
- STEVAL-MKI230KA adapter board of IMU sensor with embedded ISPU [3]

The recommended sensors settings were defined as follows:

- Accelerometer: 8 g
- Gyroscope: 2000 dps
- Sampling rate: 25-100 Hz

The competing teams were free to experiment with various sensor settings and were expected to choose the appropriate ones based on the types of activities they were training to recognize. Figure 1 and 2 depict the required sensor carry position to support the data acquisitions and inference.

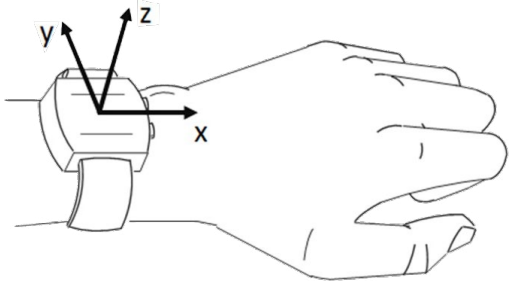

Figure 1: Sensor orientation

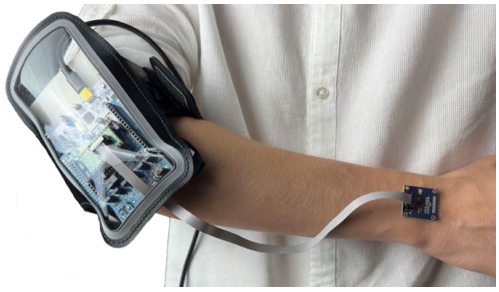

Figure 2: Example of sensor and development boards carry positions.

V. Software Required

To support the data collection X-CUBE-ISPU expansion software package for STM32Cube was used for flushing the STM32 MCU with ISPU Data Collection firmware [4]. Moreover the Unicleo-GUI graphical user interface was adopted for sensor setup and data collection [5].

Neuton.ai platform [6] was used for data transformation, extraction of features from raw sensor datasets as well as model training and code generation to be compiled onto ISPU. The ML process was fully automated; the user was required to set up appropriate configurations for feature extraction pipeline, the remaining was handled by automated model training engine. Exemplary snaphosts of the tool used in different phases are shown in figures 3, 4, 5.

VI. Data Collection Protocol

A. General recommendations

Data collection procedure required the highest amount of effort and time. If performed wrongly, it could result in poor model quality. The teams were required to make sure to collect all the data with caution and accuracy. The following sections provided all the necessary guidelines to be carefully followed.

B. Data volume

For the proof-of-concept it was enough to collect data from a single person: 2-5 minutes of signal data for each activity. For a more robust model that was meant to be able to generalize more on the general population (who's data had not been used for model training), it was required to collect data from several users: 3-5 people. A production grade model required from dozens to hundreds of unique users, depending on the classified activity and the number of classes.

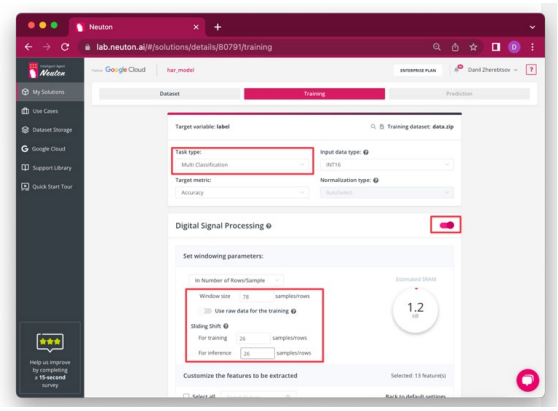

Figure 3: Signal processing options

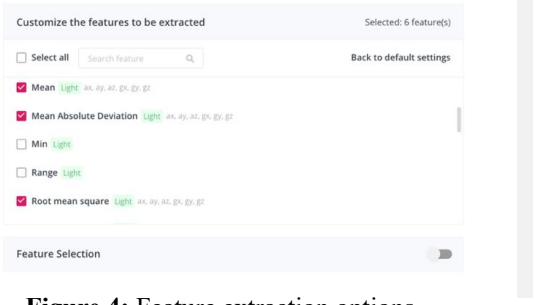

Figure 4: Feature extraction options

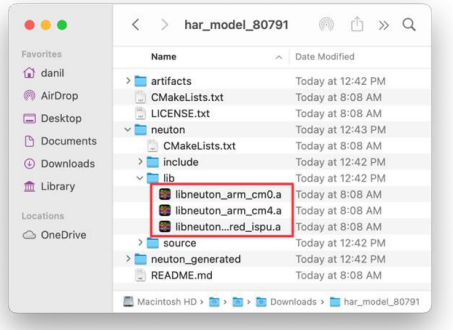

Figure 5: Final model archive example (downloaded from Neuton.ai after model was trained)

C. Data volume

For the proof-of-concept it was enough to collect data from a single person: 2-5 minutes of signal data for each activity. For a more robust model that was meant to be able to generalize more on the general population (who's data had not been used for model training), it was required to collect data from several users: 3-5 people. A production grade model required from dozens to hundreds of unique users, depending on the classified activity and the number of classes.

D. Data variability

It is advised to collect data for each activity in more than one session, preferably within two separate days, so that there will be slight differences in the collected data for each activity. When collecting data for same activity with multiple sessions, the teams had to make sure to label the files accordingly, so that the activity name and session number could later be extracted from the file name.

E. Classes balance

It was required to have a similar amount of training samples for each activity. For example, if a team had 5 activities and decided to collect 2 sessions for each activity 2 minutes each – the overall data then included 20 minutes of sensor readings with balanced number of each class.

F. Idle/unknown class

Regardless of the number of classes the teams were required to predict, the training data had to be examples of unknown activities. A NN inference, when the subject was doing anything else other than the classes trained, shall predict the 'unknown' class.

G. Applicable sensors

Typical human activities were best represented by accelerometer & gyroscope data. Even if no team anticipated the need of the gyro data, it was required to collect it anyway and discharge it later (if

not needed). This is because if the model trained on accelerometer-only data achieved poor results, the team had the option to include the gyro data for new training iterations. From this point of view, it was clear that data collection was the most time-consuming task of the challenge.

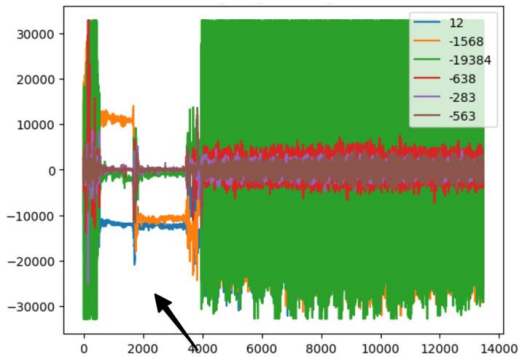

Figure 6: Example of data collection errors. X-axis – sensor readings sequential index, Y-axis – sensor readings values, legend colors represent three axis of accelerometer and three axis of gyroscope

A. Data Collection Errors Handling

Irrelevant sensor readings (for example when the team had started recording but did not start the actual activity yet) could corrupt the training data and were removed from the collected data afterwards. Fig 6 demonstrates an example where part of collected sensors data has a gap which could be reasoned by the subject pausing activities without stopping the data collection process.

VI. Teams' Overview

3 teams competed on the IEEE COINS 2023 contest and where from India, Tunisia, and Italy.

VII. Teams' Achievements

Predicted classes were 11	
At rest standing	Combing hair
At rest sitting	Table cleaning
Washing hands	Eating with spoon
Brushing teeth	Cleaning white board
Shaving	Typing
Writing	

Table 1: number of predicted classes and class names for team 1

Signal interpretation window: 128 samples
Model accuracy: Balanced accuracy: 0.896104
Model memory required:
Data RAM: 4,580 bytes;
Program RAM: 26,860 bytes.
Inference latency: 133 ms

Predicted classes were 24		
Shoveling	Handshake	Shaving with razor
Writing on board	Hand still	Washing face
Bolting a bolt	Jogging	Washing hands
Brushing hair	Using mouse/keyboard	Operating a wheelchair
Clapping	Punching a boxing bag	Wiping a table
Cutting vegetables	Roping	Wiping a board

Making dough	Cutting with saw	Writing on paper
Hammering nails	Using screwdriver	Unclassified class

Table 2: number of predicted classes and class names for team 2

Signal interpretation window: 40 samples
Model accuracy: Balanced accuracy: 0.814272
Model memory required:
Data RAM: 2,012 bytes,
Program RAM: 17,492 bytes.
Inference latency: 152 ms.

Predicted classes were 21		
Still	Chest fly	Dips
Walking	Pull ups	Triceps pushdown
Running	Lats pulldown	French press
Biking	Pulley	Shoulder warm-up
Rope jumping	Squats	Shoulder press
Push ups	Bicep curl	Shoulder lateral raises
Chest press	Hammer curl	Abs crunches

Table 3: number of predicted classes and class names for team 3

Signal interpretation window: 52 samples
Model accuracy: Balanced accuracy: 0.931678
Model memory required:
Data RAM: 1,840 bytes,
Program RAM: 12,532 bytes.
Inference latency: 21 ms.

The Team 2 and Team 3 models (as shown in table 2 and 3) were trained to predict more than 20 classes, and capable of achieving smaller memory footprint compared to the solution of Team 1 model (which was trained to predict 11 classes as reported in table 1).

IX. Final Scores and Discussions

As shown in table 4, Team 2 and Team 3 achieved very similar final scores. Team 2 won the challenge by a small margin of 7 points.

Teams' scores		
Team name	***Score calculation formulae***	***Total score***
Team 1	(200 * 0.43) + (100 * 0.9) + (80 * 0.0) + (100 * 0.45) + (50 * 0.19) + (100 * 0.4)	271
Team 2	(200 * 1.0) + (100 * 0.81) + (80 * 0.0) + (100 * 0.76) + (50 * 0.48) + (100 * 1.0)	**482**
Team 3	(200 * 0.87) + (100 * 0.93) + (80 * 0.47) + (100 * 0.78) + (50 * 0.64) + (100 * 0.6)	475

Table 4

Team 2 produced the most robust NN model which could accurately identify 24 classes of very diversified activities blended in a single NN model. It had been operating on the sensor built-in ML computing capabilities. The model had demonstrated high resilience to false positive predictions with consistent output of 'Unclassified' class prediction when the subject had been transitioning between activities.

X. Conclusions

This challenge was organized several months before the IEEE COINS 2023 conference in coordination with the conference chair, ST and Neuton.ai. It demonstrated that Tiny ML models were successfully deployed at the 'cutting-edge' level into the ISPU sensor. In addition, it revealed that the featured tools & technologies allow beginners engineers to device sophisticated

Tiny ML applications without having to write a single line of code. This opened a new perspective in terms of productivity for study and development of practical ML applications with previously unmatched power efficiency. Moreover, it has been proven that if more efforts are devoted into the creation phase of a project, its execution can be accelerated and guided by less expert engineers. This will help the process to widespread ML application at the deep edge with unprecedented productivity at a large scale.

Acknowledgments

Special thanks go to teams 1, 2 and 3 participating in the competition and a contribution to a very successful IEEE COINS 2023 contest. They were awarded with money prizes for their efforts offered by ST.

Debt of gratitude goes to the IEEE COINS 2023 conference chair Farshad Firouzi, Ph.D. [7] for his enthusiastic support in all competition's preparatory phases, the conference itself and during the award ceremony.

References

1. Nucleo L476RG: dev board: https://www.st.com/en/evaluation-tools/nucleo-l476rg.html
2. X-NUCLEO-IKS01A3: sensor expansion board: https://www.st.com/en/ecosystems/x-nucleo-iks01a3.html
3. STEVAL-MKI230KA: adapter board of IMU sensor with embedded ISPU: https://www.st.com/en/evaluation-tools/steval-mki230ka.html
4. X-CUBE-ISPU: expansion software package for STM32Cube: https://www.st.com/en/embedded-software/x-cube-ispu.html
5. Unicleo-GUI: graphical user interface: https://www.st.com/en/development-tools/unicleo-gui.html
6. Neuton.ai platform: neuton.ai
7. Farshad Firouzi, Ph.D.: https://www.linkedin.com/in/farshadfirouzi/