Energy-efficient neural networks with near-threshold processors and hardware accelerators

Jose Nunez-Yanez*, Neil Howard**

***University of Bristol, UK ****Sensata Technologies, UK**

Abstract

Hardware for energy-efficient AI has received significant attention over the last years with both start-ups and large corporations creating products that compete at different levels of performance and power consumption. The main objective of this hardware is to offer levels of efficiency and performance that cannot be obtained with general-purpose processors or graphics processing units. In parallel, innovative hardware techniques such as near- and sub-threshold voltage processing have been revisited, capitalizing on the low-power requirements of deploying AI at the network edge. In this paper, we evaluate recent developments in hardware for energy-efficient AI, focusing on inference in embedded systems at the network edge. We then explore a heterogeneous configuration that deploys a neural network that processes multiple independent inputs and deploys convolutional and LSTM (Long Short-Term Memory) layers. This heterogeneous configuration uses two devices with different performance/power characteristics connected with a feedback loop. It obtains energy reductions measured at 75% while simultaneously maintaining the level of inference accuracy.

1. Introduction

Over the last few years, novel hardware for deep-learning in AI from well-known companies and start-ups have entered the market with a focus on high energy-efficiency/performance and low cost. These devices generally focus on only supporting network inference and they analyse data produced near to the sensor locations under strict constraints of performance and power. On the other hand, GPUs remain dominant in the area of training, although some hardware is optimized for both training and inference such as the Graphcore Intelligence Processing Unit (IPU) [1]. These IPUs are massively-parallel, mixed-precision floating-point processors made available as PCIe cards. The cost and power of these devices mean that they are more suitable for cloud deployments (as done with Google TPUs) rather than at the network edge.

At the network edge, the power profile of many deep-learning devices is in the order of 1 Watt and typical examples include the EdgeTPU from Google, Neural Compute Stick (NCS2) from Intel and hardware from start-ups such as XMOS's Xcore.ai [2]. In terms of hardware features, these accelerators include dedicated hardware engines designed to accelerate compute-intense kernels such as matrix-matrix multiply, with systolic arrays being the preferred approach in many cases. Companies such as Eta Compute [3] and Ambiq [4] have targeted even lower power, at the milliwatt level, by creating microcontrollers that exploit novel voltage scaling, such as near- and sub-threshold computing. Both of these products are based on the ARM architecture. The rising popularity of RISC-V has also resulted in other novel architectures. One example is GAP8 [5] that, in addition to having 8 RISC-V processors for compute and another one for control, adds a neural network accelerator engine. The interest in this hardware has been facilitated thanks to standard frameworks that are able to train and create neural network configurations that can then be further optimized using hardware-specific tools. A framework that has grown in popularity for this type of resource-constrained AI at the edge is TensorFlow Lite [6] and the hardware used in this paper combines it with some vendor-specific steps to perform final optimizations.

In this research, we focus on energy efficiency and computing adaptivity at the network edge, with the aim of adapting energy resources to changes in the input data behaviour in real-time.

We investigate how to build a heterogeneous neural network using energy-efficient state-of-the-art hardware applied to human activity recognition as a case study. We consider commercially-available devices with different levels of power/performance based on near-threshold microcontrollers. These have a large range of performance points at different voltages. We also consider neural network accelerators as a more capable alternative, though at the cost of higher power. Without loss of generality, we will refer to these two device types as “little” and “big” since our methodology is inspired from the “big.LITTLE” [7] ARM technology that has been popular in the area of mobile computing. We consider that both “little” and “big” devices are physically collocated in an embedded system. The novel contributions of this work can be summarized as follows:

1. Evaluate the power, energy and performance of state-of-the-art ultra-low-power processors to measure the benefits of near-threshold computing processors as “little” devices.
2. Propose a methodology that splits a complex problem that includes convolutional and recurrent layers into simpler problems. Then train independent networks to solve these simpler problems.
3. Propose a practical deployment of this heterogeneous setup using state-of-the-art “big” edge neural accelerators with “little” near-threshold processors, evaluating the effects on inference accuracy compared with a homogeneous (“big”-only) solution.
4. Evaluate the energy and performance of the proposed heterogeneous hardware compared with using the “big” edge neural accelerator.
5. We have released the training/inference setup for the heterogenous components at https://github.com/eejlny/subthreshold_hetero_mcu_ncs

2. Background

Edge computing is seen as a solution to the latency and privacy issues associated with cloud computing but a major challenge is being able to run AI workloads on compute resources that are limited by power and capabilities [8]. The review conducted in [9] establishes the advantage of hybrid computing schemes between device, edge and cloud servers. In this paper we focus on device and edge AI formed by a heterogenous compute system that extends an edge neural network accelerator with a low-power subthreshold processor with limited compute capabilities.

Typically, these edge devices are specialized for inference and support low-precision arithmetic such as 8-bit integers. For example, Gyrfalcon’s matrix processing engine (MPE) uses processor-in-memory techniques to compute neural network models. The 2803 chip [10] delivers 24TOPS/W and consumes a minimum power of 700mW. Similarly, Google offers a low-cost and low-power version of the Google TPU called EdgeTPU [11] that can run dedicated convolutional neural networks with 8-bit precision. This is capable of 4TOPS while consuming 2W. The systolic array size is much smaller than a cloud TPU at around 64 x 64 mult-add cells, resulting in 4TOPS at 480MHz. The EdgeTPU is not capable of general network training which requires floating-point precision, although it can deploy transfer learning techniques [12]. FPGA vendors such as Xilinx have also focused on inference, with the Xilinx DPU [13] unit. This contains register configuration, data controller, and convolution computing modules optimized for the FPGA hardware resources. An encapsulated scheduler can assign tasks to multiple DPUs and tools are available, such as Dsight, to monitor performance. It uses a systolic array architecture for matrix multiplications and packs 8-bit operators into the device DSP blocks. A popular alternative on FPGAs is to design binarized accelerators that use single bits for weights and activations. This means that complex

multiplication operations are replaced with XNOR and popcount operations. This strategy is used in [14] that proposes an architecture that initially transfers the convolution operations into matrix multiplication by unfolding and duplicating inputs and rearranging the weights. The system uses a dataflow architecture to map full networks to a large Zynq device 7z100. The dataflow architecture uses a balanced pipeline, with buffers to smooth out the flow of computations, and achieves a very high throughput. The limitation is that the hardware complexity means that only small networks with few layers and filters can be mapped before hardware sharing is required. An alternative to reduce complexity without the extreme quantization used in binary networks is presented in [15]. This assigns variable bit widths to layers with a general reduction in precision for deeper layers. The PEs (processing elements) support bit-serial multiplication and the weight bits are sent serially with additional PEs used to compensate for the lower throughput of the bit-serial architecture, which is much simpler than bit-parallel multiplication hardware. The concept of using mixed-precision hardware is also explored in [16] which proposes a scheduling strategy to distribute real-time tasks associated with sensor data acquisition, inference and action on a heterogenous system that combines an FPGA and CPU. The FPGA inference accelerator supports precisions from 8 to 64 bits, resulting in different computation times that the scheduler needs to take into account.

Intel also offers FPGA-based solutions that can be targeted with their OpenVino [17] toolkit that has, as main components, the model optimizer and inference engine. OpenVino takes as input a trained network, using a framework such as TensorFlow/Keras, and optimizes it to target Intel hardware that can be Intel CPUs, GPUs, FPGAs, or other devices such as the Intel NCS2 based on the Myriad Vision Processing Unit (VPU). The VPU has 16 VLIW general-purpose programmable cores, optimized for vision processing workloads, and includes a neural compute engine to accelerate tensor matrix calculations. The VLIW cores also enable other algorithms, unrelated to deep learning, although the current OpenVINO toolset focuses just on supporting recurrent and convolutional neural networks. The VPU uses 16-bit floating-point arithmetic instead of integer arithmetic, but these more complex data types, compared with fixed point arithmetic, also imply that performance must be lower at the same silicon size.

The previously-presented architectures vary in their power, performance and capabilities, and heterogeneous solutions have been proposed combining them to improve on these individual aspects. In [18], JointDNN is proposed as a collaborative neural network for cloud and mobile systems that processes some layers on the mobile device and some layers in the cloud device. An optimization algorithm based on Integer Linear Programming considers communication costs, computation costs and energy requirements to schedule layers to the cloud and mobile device with some networks, such as autoencoders, mapping the first and the last layers to the mobile device. In JointDNN, all the layers are computed. In contrast our work targets heterogeneous edge systems with the aim of reducing the activation of the most complex parts of the network. Distributed computing has been explored in MoDNN [19] and DeepThings [20] that use fine-grained partitioning on lightweight end devices such as Raspberry Pis and Android smartphones. The run-time assigns tasks to independent processor nodes that are homogenous. This previous work focuses on systems with multiple CPUs and GPUs or collections of homogenous simple processors. In contrast, we focus on using the “big.LITTLE” concept, specifically applied to deep learning tasks.

Our approach is different from other related work (exploring “big” and “little” configurations for neural networks) such as [21] and [22] that construct a "little” network with the same inputs and outputs as the “big” network and uses the score metric from the "small” network to decide when to activate the “big” network. In our case, the “little” and “big” networks perform different tasks and information flows in both directions, in a feedback loop. The idea of using more than one network to achieve a classification task has been explored before, with concepts such as

networks in which multiple classifier heads of the same network are simultaneously trained on the same training data in order to improve generalization and robustness [23]. Also related is the concept of knowledge distillation, where a smaller network is trained by transferring knowledge from a pre-trained high-capacity model, achieving network compression and resulting in a smaller network that obtains better performance than one trained independently, using only labels [24]. Our framework is conceptually different since training of the “little” and “big” models is done independently, with different labels and with an information exchange between “big”-to-”little” and “little”-to-”big” at run-time, in order to obtain an overall inference accuracy that is as good as the “big” model, but with much lower energy costs.

Also related to our approach is the fast exiting presented in [25], which tries to make decisions closer to the edge so that the cloud device is not used, or commercial products such as Apple Siri. Siri employs two on-device networks to classify speech into one of 20 classes such as general speech, silence and wake-word. In Siri, the first DNN is smaller, with 5 layers with 32 neurons, and runs on a low-power, always-on processor and it triggers a second, more powerful, DNN (5 layers with 192 units) on the main processor. The objective of the first network is to detect if something has been said and the objective of the second is to perform the full classification. The solution presented in this paper is different because it has information flowing in both directions, adapting the model of the simple network which not only detects the presence of an activity, but also fully classifies the activity in most cases.

3. Low-power processor evaluation

Table 1 shows technical details of commercially available processors aimed at low-power applications. A selection of four shown in bold letters is used to perform the initial evaluation in this section. The table shows that a popular microarchitecture is the 32-bit ARM Cortex-M, with the armv7-M instruction set, that is used by three of the selected candidates, provided by Eta Compute, Ambiq and STM. Recently there has been significant interest in using open-source microarchitectures such as RISC-V which are royalty-free. Several companies have introduced devices based on the RISC-V microarchitecture. An example shown in Table 1 is GAP8 [26] which, in contrast with the other solutions presented in the table, is a multi-core architecture of PULP RI5CY cores, with one fabric controller and 8 computing cluster processors. The fabric controller acts as a master for control and communications with the cluster and it is responsible for memory allocation, execution start, output collection and, finally, memory deallocation. The GAP8 processor is also characterised by having an additional dedicated hardware engine to speed up the convolutions needed in neural network applications. This hardware convolution engine (HWCE) can generate a 5-by-5 convolution or a pair of 3-by-3 convolutions, with 16-bit operands, in a single cycle. A set of tools like NNtool and auto-tiler are available to facilitate the mapping of the network model to the processor cluster.

All the considered processors use voltage and frequency scaling techniques in order to optimize how energy is used, depending on performance requirements, but while the STM and GAP8 processors can be considered to use standard DVFS (Dynamic Voltage and Frequency Scaling), the Ambiq and Eta Compute parts deploy proprietary forms of AVS (Adaptive Voltage Scaling). AVS is characterized by using some form of sensing of the device operation in a feedback loop at run-time to better track the frequency supported at a given voltage. In contrast, DVFS uses fixed voltage and frequency pairs, predetermined at fabrication time [27]. The STM32L4 and Ambiq Apollo3 are both Cortex-M4 processors while the ECM3531 is a Cortex-M3. Cortex-M4 and M3 are both similar cores, with a 3-stage pipeline and Thumb-2 instruction set, but the Cortex-M4 adds a range of instructions and hardware, such as 16/32-bit MAC and 8- and 16-bit SIMD arithmetic, specifically optimized to

handle DSP algorithms. Both the M3 and M4 cores in Table 1 include floating-point hardware support. The Apollo3 also adds 4KB of data cache while the STM32L4 uses a special accelerator called “ART” to implement a small cache memory. The ECM3531 uses an additional “Coolflux” 16-bit DSP designed by NXP to complement the simpler Cortex M3 microarchitecture, although this DSP is not currently used by the neural network libraries. The main fundamental difference between the STM and the Ambiq/Eta Compute devices is the power optimization technology that, in the last two cases, uses “near-threshold” computing techniques, with a core supply voltage at 0.5V compared with the 1V used by the STM device. In principle “sub-threshold” computing, with voltage supplies around 0.3V, should be even more efficient. However, at such low voltages, there is a large increase in energy loss through leakage, due to the massive increase in switching time of the transistors. These means that, in practice, using near- rather than sub-threshold computing has been found to be more optimal from an energy point of view. Comparing the datasheet power values of the near-threshold Eta Compute and Ambiq Apollo3 with the traditional low-power microcontroller from STM, the power advantage is indicated to be around a factor of four. It is also clear that the neural model must be small since it needs to fit in a just a few KB of available memory which is also needed to allocate run-time data structures, program binaries etc.

Table 1. Power efficient MCU alternatives

Device	Architecture	Fabrication Process	Embedded Memory (flash/SRAM/ cache)	Low core Voltage	Coremark Power	Power scaling capabilities
STM32L4	**32-bit Cortex-M4F**	**90nm**	**1 MB/320 KB**	**1.05V**	**133 uA/MHz 80 MHz 114 uA/MHz 26 MHz**	**Voltage scaling between 1.08V at 80 MHz and 1.05V at 26 MHz.**
GAP8	**32-bit RISC-V multicore**	**55nm TSMC**	**0 KB/64KB+512KB/0 KB**	**1.0V**	**N/A**	**Voltage scaling between 1.0V at 90 MHz (cluster) 1.2V at 175 MHz (cluster)**
Apollo2	32-bit ARM Cortex-M4F	TSMC 40nm	1MB/256KB/ 16 KB	0.5V	14.8 uA/MHz 48 MHz	SPOT (Subthreshold Power Optimized Technology) with two main power levels: SPOT 24MHz/48MHz
Apollo3	**32-bit ARM Cortex-M4F**	**TSMC 40nm**	**1 MB/384 KB/16 KB**	**0.5V**	**10.3uA/MHz < 48 MHz 27 uA/MHz > 48 MHz**	**SPOT (Subthreshold Power Optimized Technology) with two main power levels: TurboSPOT 96MHz/ SPOT 48MHz**
Eta Compute ECM3531	**32-bit ARM Cortex-M3 with 16bit DSP coolflux**	**55nm ULP TSMC**	**512 KB/256 KB/0 KB**	**0.55V**	**13uA/MHz**	**CVFS (continuous voltage scaling) with multiple working points of frequency and voltage available.**
Eta Compute ECM3532	32-bit ARM Cortex-M3 with 16-bit “Coolflux” DSP	55nm ULP TSMC	512 KB/256 KB/0 KB	0.55V	5uA/MHz	CVFS (continuous voltage scaling) with multiple working points of frequency and voltage available.

To perform an initial evaluation of the performance, power and energy characteristics of the preselected 4 processor cores we have mapped an MNIST model with a topology as shown in Fig. 1.

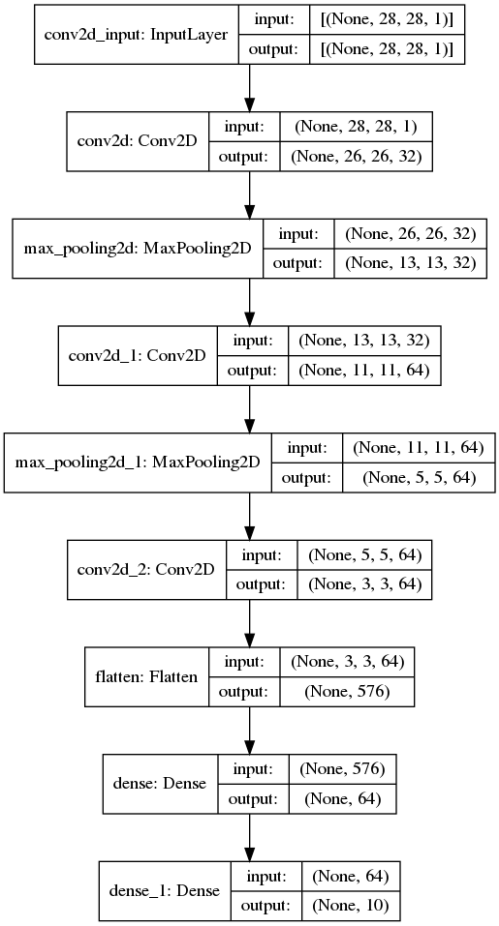

Figure 1. Convolutional neural network test topology

For the power measurements we have used a x-nucleo-LPM01A as a power source for all the Cortex-M class processors, measuring the current draw with a voltage supply at 3V. We have used a Picoscope oscilloscope to measure the voltage drop in the 1Ohm shunt resistor available in the GAP8 board. This enables us to calculate the current going into the processor, and hence also the power by multiplying with the processor voltage. Fig. 2 shows the initial analysis of power consumption on the tested devices. The figure shows that GAP8, Apollo3 and ECM3531 devices operate at similar power levels if we consider a GAP8 using only one core. The GAP8 device is optimized for only two voltage levels, corresponding to voltage levels 1.0V and 1.2V. Similarly, the Apollo3 device is optimized for frequencies of 48 MHz and 96 MHz and shows that power scaling slows down at a selected frequency of 24 MHz.

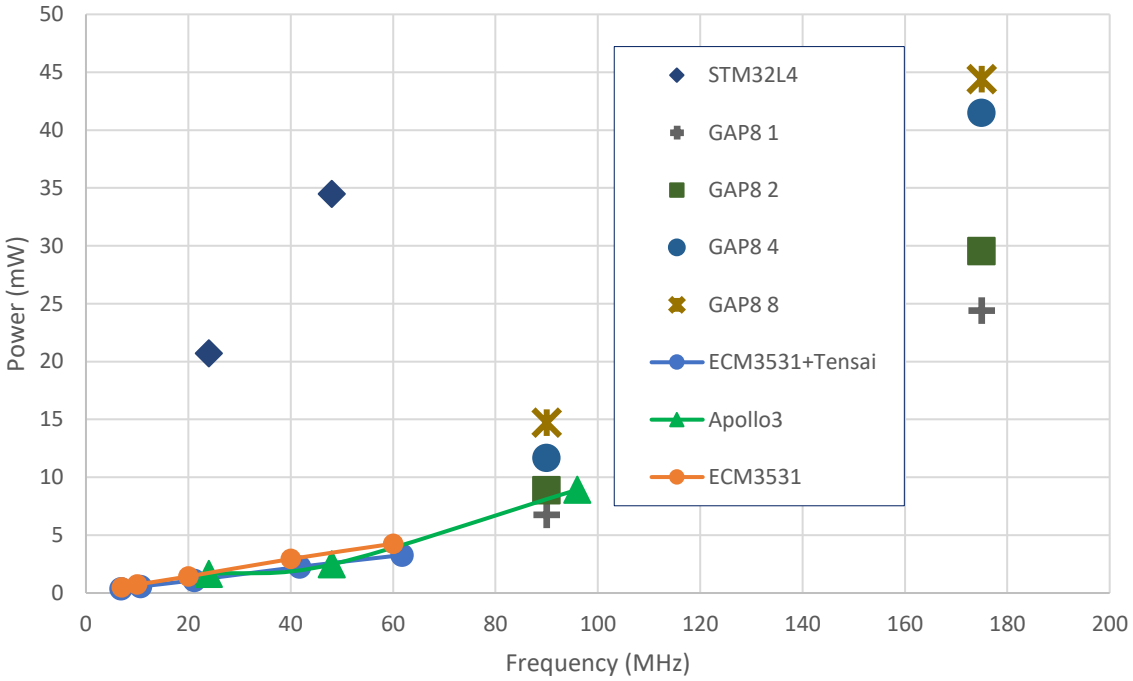

Figure 2. MCU power evaluation

Fig. 2 shows that the power efficiency of the STM device is significantly worse than the rest of the devices considered. Our application scenario will benefit from a large range of optimized frequency/voltage points so that it is possible to select the point that delivers enough performance depending on input data. This is illustrated in Fig. 3 that shows how, conceptually, we aim at adapting the working point of the device depending on the complexity of the inference task. For example, the “little” device samples the input data reacting to its own inference decisions that are used to estimate the amount of activity in the input sensor. This means that the inference rate increases if an increase in activity is detected, indicating the presence of data that will benefit from being observed at a higher resolution. The increase in inference rate means that less time is available to complete the request, so the voltage and frequency point of the “little” device increases. The “big” device intervenes when the “little” device detects an activity type change, assisting the “little” device in performing the classification. After a while, the processing rate decreases if the “little” device stops detecting activity changes. Conceptually, AVS with multiple frequency and voltage points can be deployed in this scenario so processing speed adapts to time and energy constraints [28].

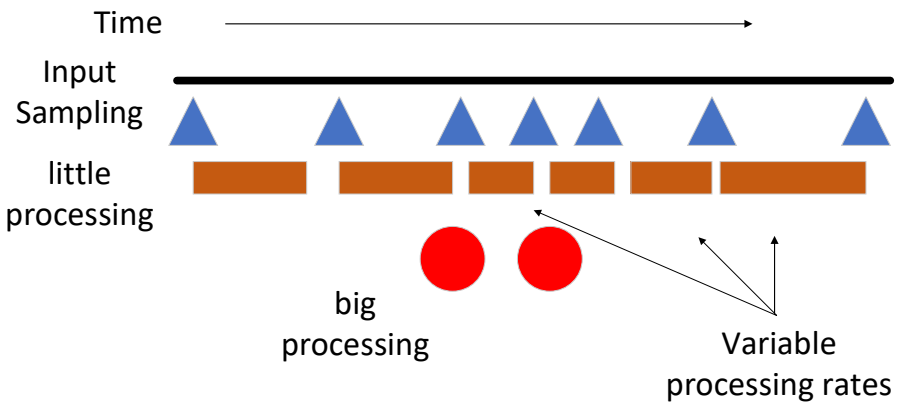

Figure 3. Adaptive heterogeneous processing

The Eta Compute ECM3531 offers this feature with internal voltage regulators that are optimized for frequency ranges between 7 MHz and 60 MHz. Figs 4 and 5 show the execution time for one MNIST inference and the corresponding energy usage. The Apollo3 and ECM3531 devices show similar execution time when the standard GCC compiler and CMSIS-NN library are used. Both devices show that the lowest power point is not the most energy efficient. This can be attributed to the increase of leakage at lower performance points which is significant challenge in low-voltage operation. The ECM3531 can be used with a proprietary optimizing compiler called Tensai, created by Eta Compute, and this results in significantly better execution times. In this scenario, the effect of voltage scaling on energy is less visible and this could be attributed to an increase of the effect of the embedded memory not having its supply voltage scaled. The GAP8 device shows the best execution time and this can be attributed largely to the usage of the hardware engine to accelerate the convolutional neural network kernels. The energy figure also shows the GAP8 device as the most energy efficient thanks to the reduction in execution time. Overall, the ECM3531 with the Tensai compiler offers a good balance between energy efficiency and low power with a large range of performance points available.

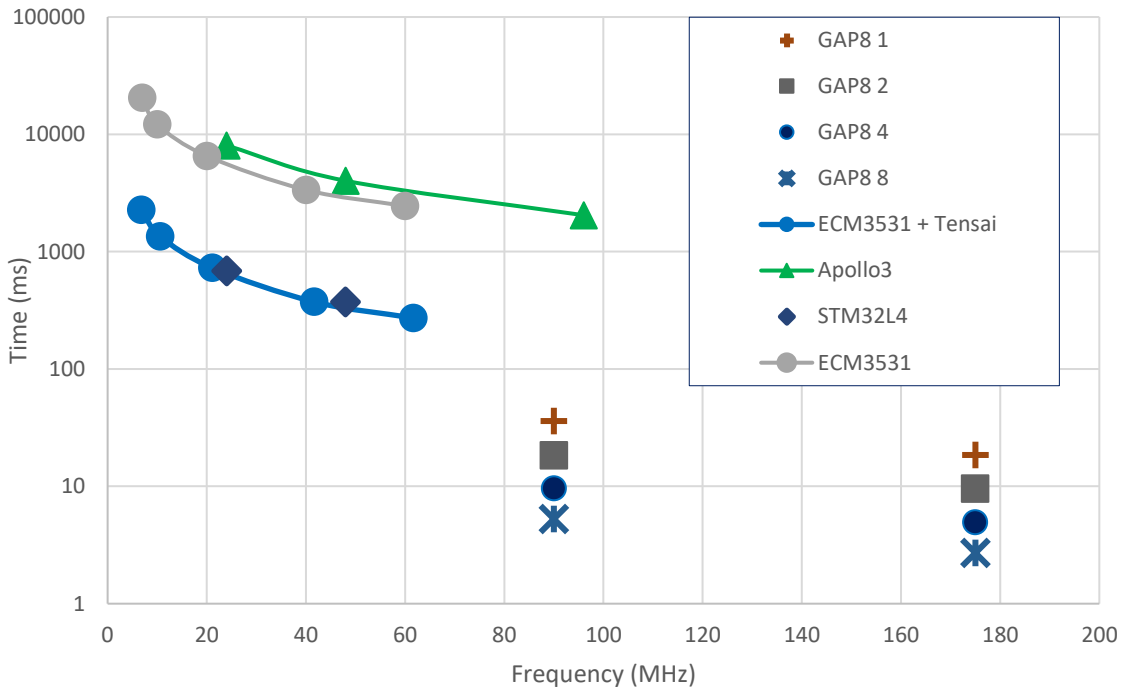

Figure 4. MCU performance evaluation

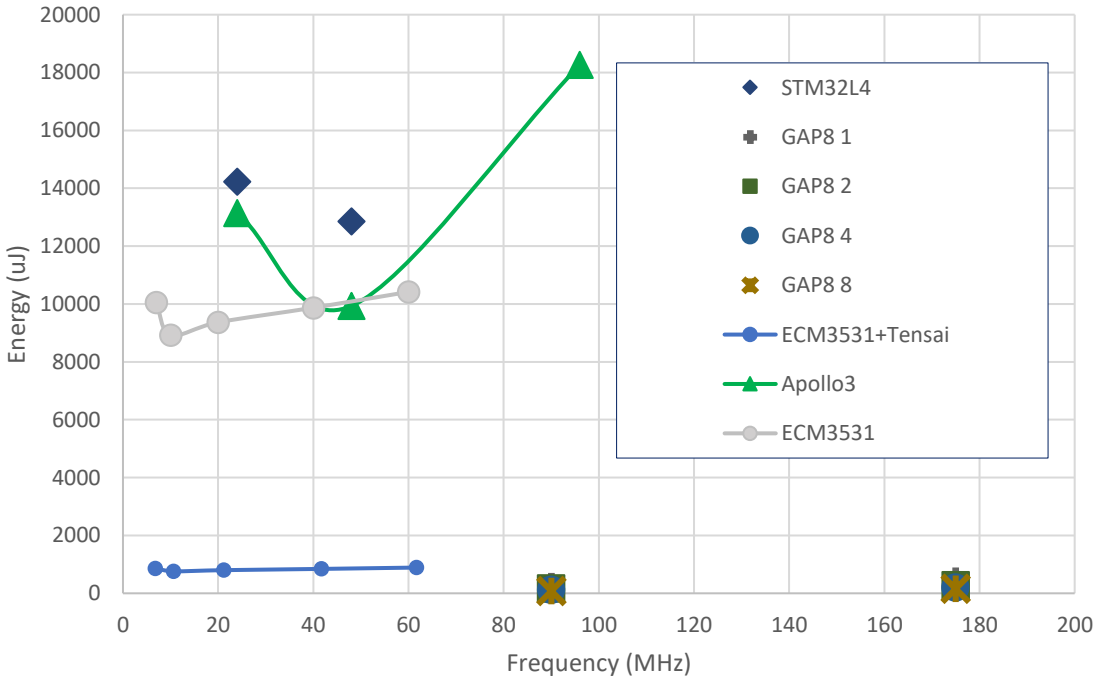

Figure 5. MCU energy evaluation

4. Heterogeneous Methodology

In section 2 we have seen that hardware solutions optimized for low-power inference are becoming popular and that these devices have variations in arithmetic precision, support for different neural layers such as recurrent layers (e.g. LSTM) and power and performance profiles optimized for different applications. Many frameworks to perform neural network training also exist (e.g. Torch, TensorFlow, MXnet, Theano or Caffe) but lately TensorFlow Lite has appeared as particularly relevant for resource-constrained devices such as mobile and edge devices. TensorFlow Lite is optimized for microcontrollers and also edge hardware accelerators. This single-entry point is useful to speed-up the development of the heterogeneous network since only the final optimization steps are customized for each device while the training step is performed only once and it is shared. The proposed collaborative network exploits the different capabilities, performance and power of these devices, creating a heterogeneous solution that improves the energy values. In this case study, we investigate a heterogeneous network mapped to two clearly distinct hardware devices. The first one is the NCS2 device from Intel, based on their Myriad series of visual processors, enhanced with dedicated execution units for neural network acceleration. Device power is estimated at around 1W with a throughput of 1TOPS with a base clock frequency of 700MHz, with 16-bit floating-point precision. The second device is a microcontroller based on a near-threshold voltage processor, operating at a minimum voltage of around 0.5V. From the initial analysis conducted in section 3, we have chosen near-threshold operating devices from Eta Compute and Ambiq as candidates for this second device.

Fig. 6 shows the proposed heterogeneous flow that uses Keras/TensorFlow as a common framework for model development. Keras is a popular high-level neural network API written in Python. It is very well integrated with TensorFlow to facilitate neural network development. The outputs of the training and evaluation stages are HDF5 Keras models. The models are then frozen to remove data and variables needed for training but not for model deployment. The resulting frozen models must then be optimized for each individual target independently. The NCS2 device uses the OpenVino toolset that also supports other Intel devices such as FPGAs, GPUs and CPUs. The NCS2 uses 16-bit floating-point precision and OpenVino replaces the 32-bit floating-point values used during training with 16-bit floating-point values. Inference can work well with even lower precision so no accuracy loss is apparent after this substitution. In this work, we have used OpenVino 2020.1.023 that also introduces support for LSTM layers. The outputs from OpenVino consist of a number of files that contain the network description and network parameters. On the other hand, the Eta Compute ECM3531 and Ambiq Apollo3 devices use neural network models generated by the TensorFlow Lite converter and can employ any precision since they are fully-fledged 32-bit ARM processors with floating-point support. In this research, we quantize all model parameters and input data to 8-bit precision in order to simplify the network and make it more suitable for microcontroller deployment. The model pruning and scaling step shown in Fig. 6 indicates our own automatic iterative step in which the original “big” model is simplified progressively for the MCU device, scaling and removing layers to satisfy the lower problem complexity that results from substituting the multiple-class problem with a two-class problem. The training and validation datasets are rearranged to a set of two-class problems for “little” model training. This problem reduction to two classes is an intrinsic part of the proposed methodology. It generates a number of models for the “little” device, equal to the number of original classes. For example, if the original model had a total of 10 classes then a total of 10 optimized “little” models are created. Pruning and scaling continues as long as accuracy is maintained to the same level as in the original classification problem. For example, if the original “big” model obtained an accuracy of 90%, then the optimization objective is that each of the individual simplified

networks achieves 90% or higher in their simplified problems in order to minimize possible accuracy degradation of the heterogeneous solution.

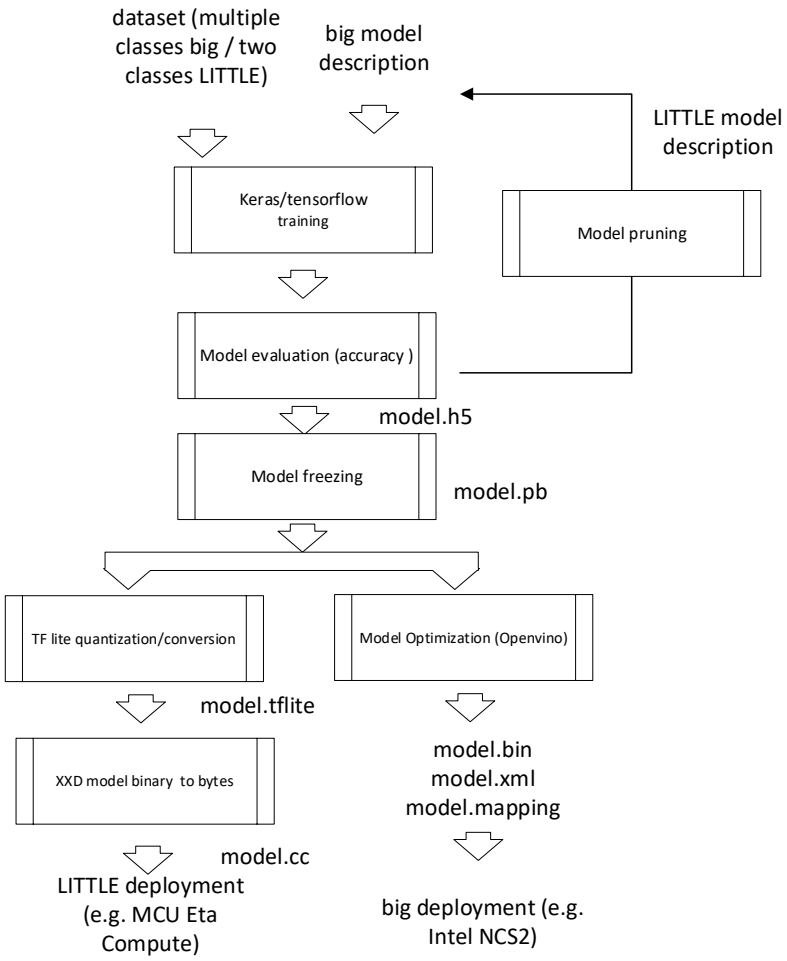

Figure 6. Model heterogeneous mapping/pruning/quantization

5. Model development

The use case application under consideration consists of an activity detection system using accelerometer and gyroscope sensors to classify user activity into 6 different classes: walking, walking upstairs, walking downstairs, sitting, lying down, and running. All data is normalized to the range [-1,1]. The accelerometer sensor produces triaxial acceleration (total acceleration) and the estimated body acceleration, while the gyroscope produces triaxial angular velocity [29]. The NCS2 "big" model consists of convolutional, max pooling, LSTM, concatenation and, finally, dense layers, as shown in Fig 7.a and originally proposed in [30]. LSTM layers are not very well supported in edge deep learning accelerators, with support either being on-going work or not possible due to the specialized pipelines of the accelerators. One of the motivations for choosing the NCS2 device is that the latest version of the OpenVino toolset supports LSTM and, consequently, it is possible to investigate its performance, energy impact and accuracy gains. The "little" model shown in Fig 7.b has been derived via pruning from the full model to obtain a highly simplified configuration in order to improve its suitability for microcontroller deployment. It uses a single sensor as input. In contrast, the "big" NCS2 device uses the input from the three sensors, which are processed individually before being

merged with a concatenate layer. The conv1d layer used in the “little” model has been simplified to only 8 filters, instead of the original 512.

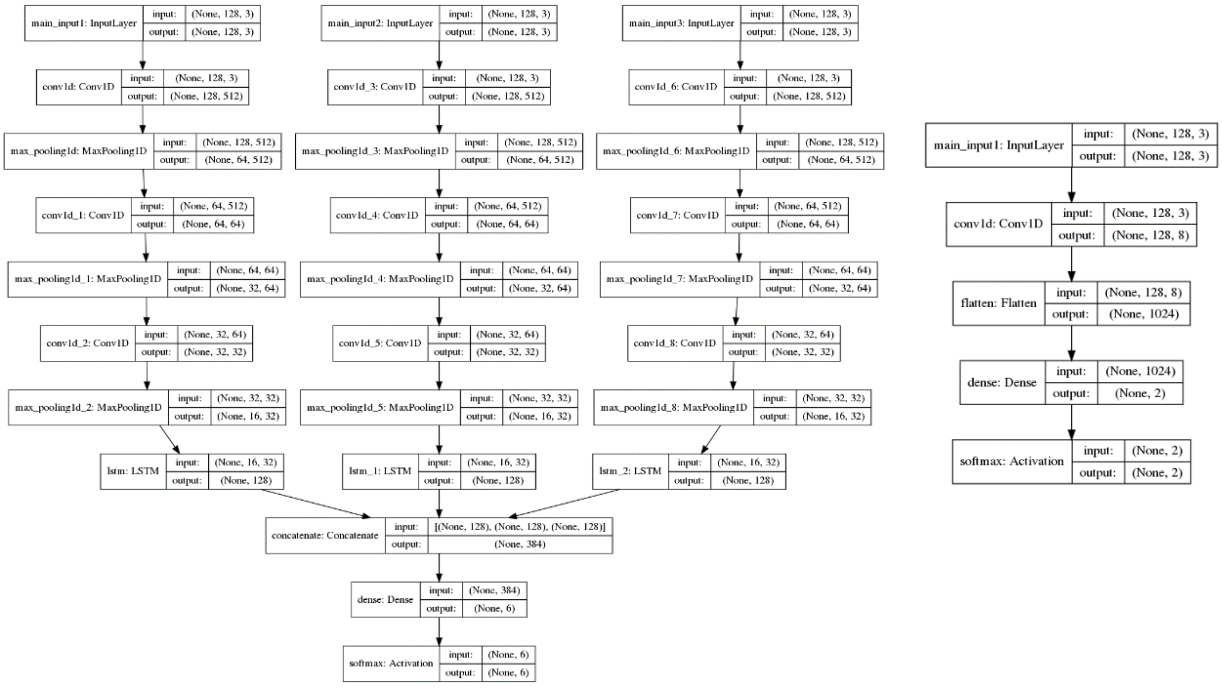

Figure 7. (a) “big” model *(b) “little” model*

Since a single input sensor is used by the “little” model, it is necessary to select which sensor to use. The “little” model needs to detect a change of the current activity to a new activity. This results in 6 possible changes and, to determine which of the available sensors offers better accuracy, each of these 6 configurations is trained with all three input sensors. Results are shown in Fig. 8. We can observe that the body_acc sensor is the most accurate for the three walking configurations and the total_acc sensor for the other three configurations: sitting, standing and laying. Overall, the “little” network obtains an accuracy of over 90%, following careful sensor selection (although it can degrade down to 82% if the wrong sensor is chosen, as seen in the walking upstairs case).

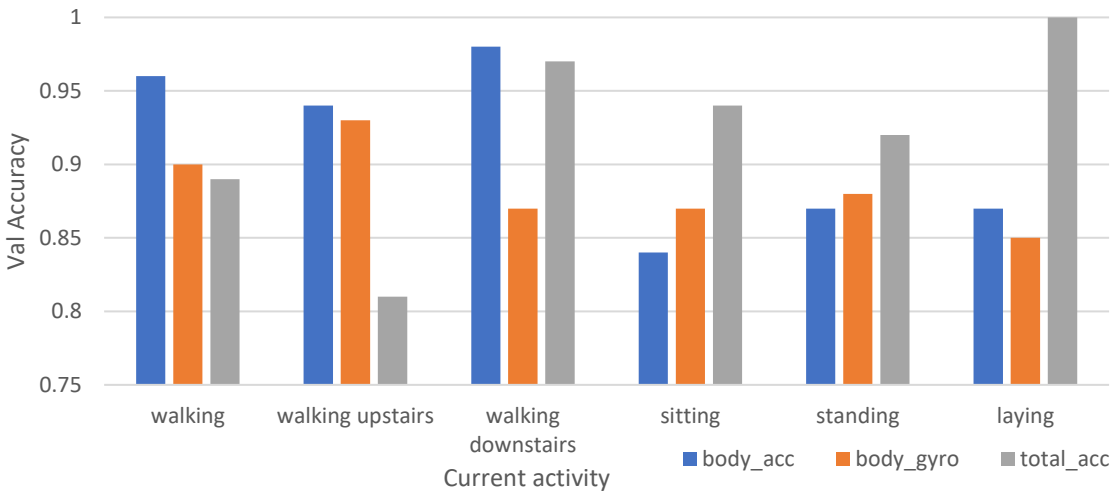

Figure 8. "little" model sensor accuracy evaluation

Fig 9.a shows, conceptually, the heterogeneous neural network composed of NCS2 and MCU devices. The NCS2 device uses a single model "A", shown in Fig 7.a, trained to classify the network activities, while the MCU device uses a single model topology "B", shown in Fig 7.b, with a total of 6 learned configurations. As indicated previously, each of these 6 configurations is specialized to detect one single activity. The original input training and evaluation data is used to train the full model, while 6 new data sets are derived to train the "little" networks. This means that each of these new data sets contains current_activity/other_activity, and current_activity can have six different values. The heterogeneous system operational mode is shown in Fig 9.b where, after initialization, the Eta Compute device computes continuously, while the NCS2 only computes when changes in the active activity are detected. The Eta Compute device communicates to the NCS2 when these changes are detected, while the NCS2 device communicates the new current activity, after inference, to the Eta Compute device. The Eta Compute device will switch its activity model to reflect the new current activity. After completing the flow of Fig 6, the resulting models are compiled, together with the control code, and the resulting binary occupies only 4KB. The full model for the NCS2 device occupies 660KB. Although it could be argued that the NCS2 model is still a small model, it would not be feasible to map it to the Eta Compute device due to flash memory constraints of 512KB.

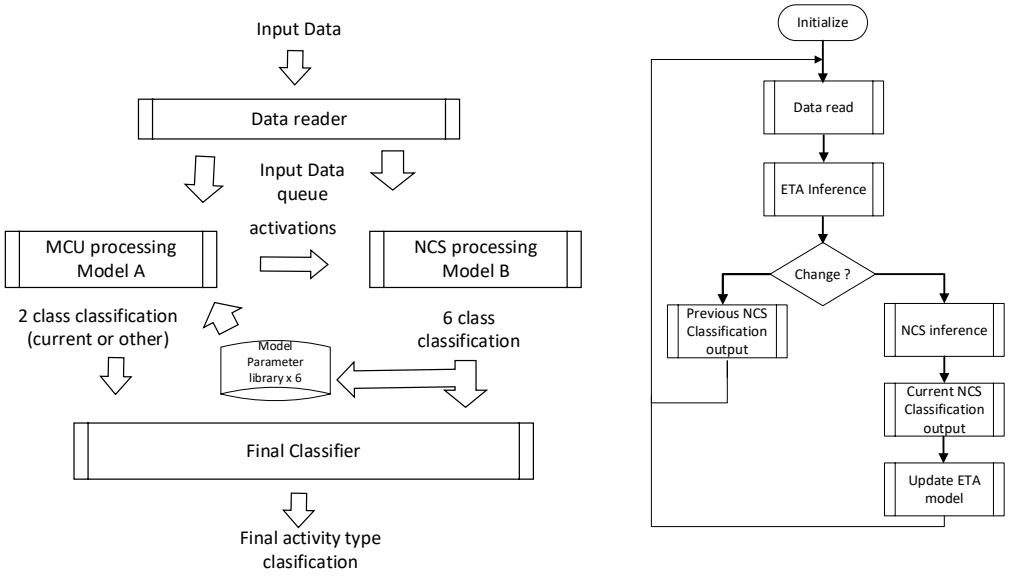

Figure 9. (a) Heterogeneous model *(b) Processing steps*

6. Heterogeneous inference accuracy

We have considered two variations of the full model: with or without LSTM layers. The effects of using these LSTM layers in model accuracy are shown in Fig. 10. In Fig. 10, the bars correspond to number of predictions and the points shown as rhombus symbols correspond to the measured accuracy. The rhombus in the figure for the LSTM NCS2 and CNN NCS2 cases, shown on the left, indicate that there is approximately 1% degradation in accuracy by removing the LSTM layers. The accuracy of these homogeneous configurations is in the range of 92% to 93%, depending on the presence of LSTM layers. The data set contains a total of 2946 data points and the corresponding grey bars in the figure show that, when the NCS2 is used exclusively, a total of 2946 inference invocations are performed, which corresponds to one invocation per input data sample, as expected. We have then evaluated the data set with the heterogeneous configuration that deploys both the “little” Eta Compute and the “big” NCS2 devices. The results, shown on the right of Fig.10, show that the accuracy of the heterogeneous configurations remains virtually unchanged. The bars now show a total of 2946 inference invocations in the Eta Compute device but only around 700 invocations of the NCS2 device. As indicated in the previous section, the NCS2 device is only used when the Eta Compute device predicts a change in the input class. The data for the model has been obtained with a sampling rate of 50Hz and, according to the data set documentation, each of the data samples correspond to 2.56 seconds of activity. Consequently, there are 50*2.56 = 128 samples per sensor and axis. There are three axes so a total of 128x3 values form an input into the neural network. There are about 25 samples in each activity window (an activity is replaced by another activity after 25 samples) in the data set, so each activity window corresponds to approximately 25*2.56 = 64 seconds. With these constraints, the neural network needs to be able to perform one classification every 2.56s in order to maintain real-time performance with the data supplied from the sensors. In a real deployment, a human activity such as walking, sitting or lying down could last for multiple minutes so, consequently, a larger proportion of decisions could be made on the low-power Eta Compute device, while

the NCS2 device remains in a deep sleep state. An additional observation is that the recurrent neural network theory indicates that LSTM layers are characterized by keeping a state memory of the previous inputs. Since, in the heterogeneous configuration, the “big” model sees only a fraction of the original inputs, this could mean that the LSTM functionality is degraded. However, in the analysis shown in Fig. 10, we see a similar classification accuracy in the heterogeneous case as in the homogenous case in which all the inputs are seen.

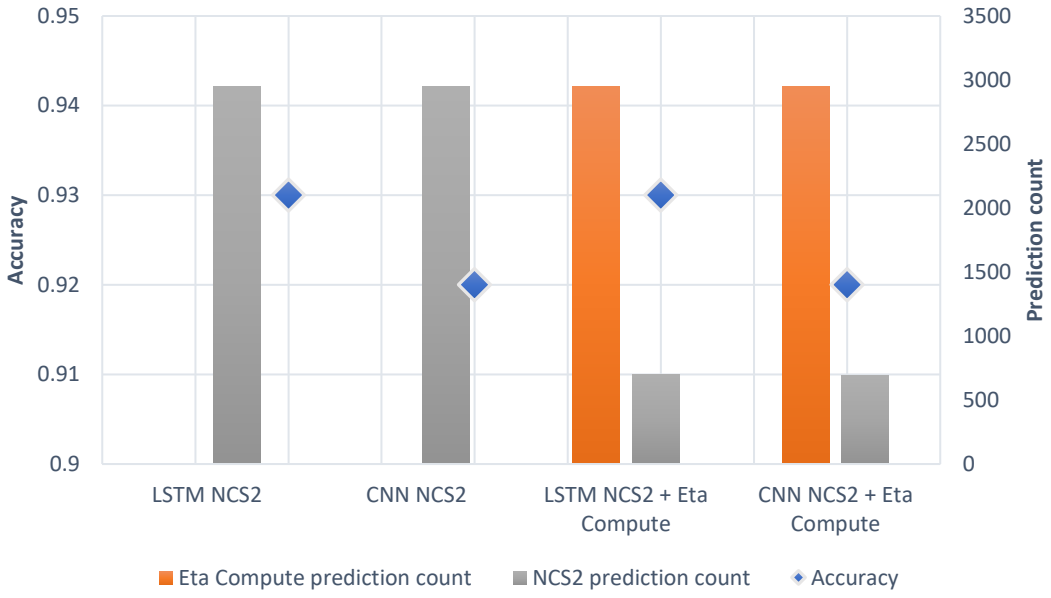

Figure 10. Heterogeneous vs homogenous model accuracy and activity

To gain further insights into heterogeneous system behaviour, Fig. 11 and Fig. 12 show the accuracy effects of the heterogeneous configuration with a window sample of 300 inputs. The truth values show the ideal behaviour (with the correct activity identified with a probability of 1.0), while the predictions at the top show the class activity predicted by the network and its probability. In a perfect system, both top and bottom figures should match perfectly. The “big” model shown in Fig 11.a, for both CNN and LSTM cases, correctly predicts the class with a probability of 1.0 for most of the samples. After approximately 100 samples, the model enters the category of “standing” that it finds difficult to differentiate from “sitting”. This is observable again at sample 270. If we compare this with the heterogeneous model shown in Fig 11.b, we can observe a similar behaviour after sample 100. Overall, the heterogeneous model shows a smoother output, although this is not necessarily better. It just means that the Eta Compute model is not detecting changes in the input and using the previous classification provided by the NCS2 model as its output. If this classification was incorrect then the error will propagate. The same experiments are repeated in Fig.12 for the case without LSTM layers. Overall, the results in terms of accuracy, shown in Fig 11 and Fig 12, confirm that, in this case study, the heterogeneous model does not degrade the accuracy compared with the homogenous model that uses a single device.

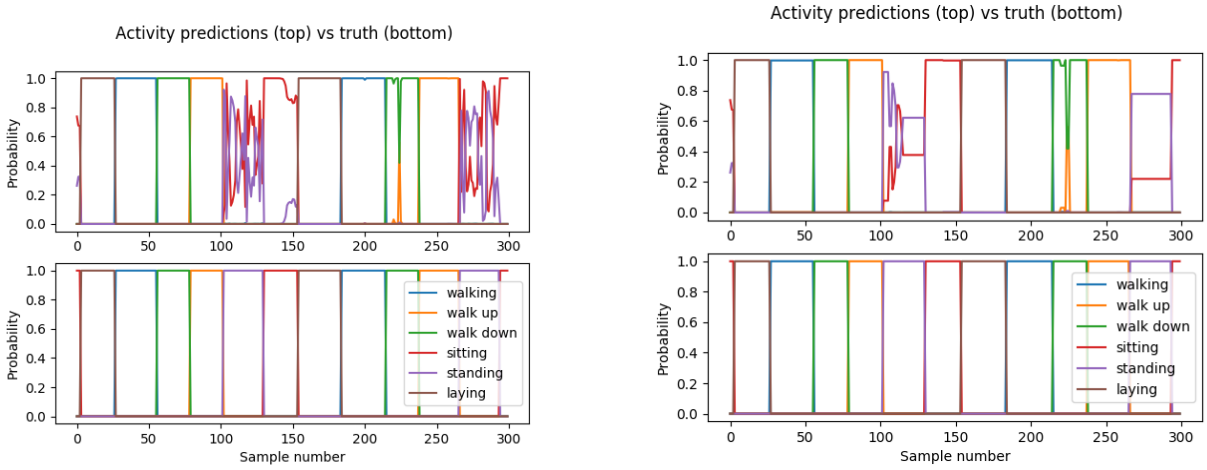

Figure 11. (a) homogeneous LSTM NCS2 vs (b) heterogeneous LSTM NCS2+Eta Compute

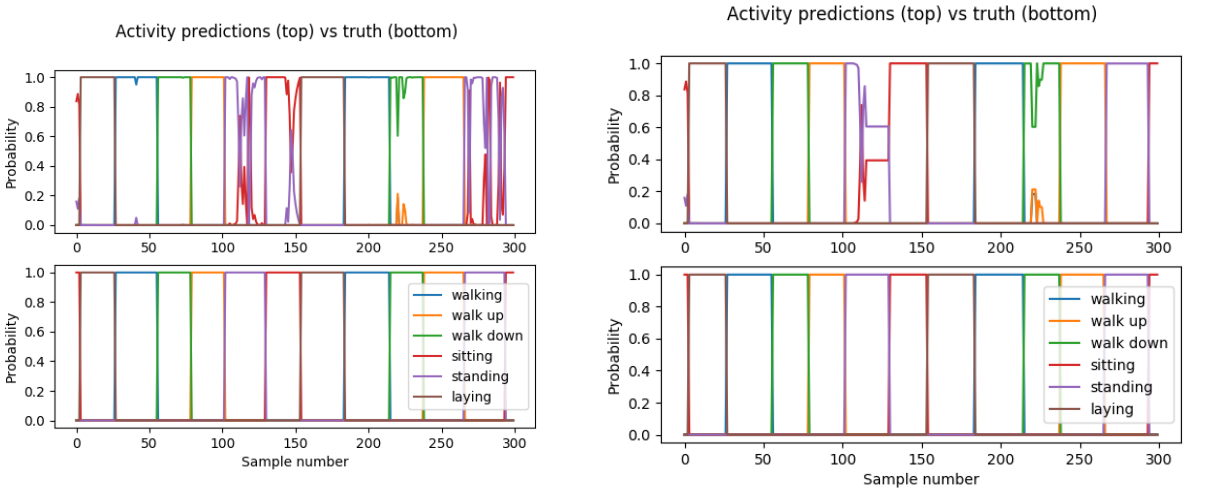

Figure 12. (a) homogeneous CNN NCS2 vs (b) heterogeneous CNN NCS2+Eta Compute

7. Heterogeneous system performance, power and energy

From the evaluation conducted in section 3, we have selected the Ambiq Apollo3 and Eta Compute ECM3531 for this final evaluation. We have mapped the “little” network presented in Fig 7.b to the Ambiq Apollo3 and Eta Compute ECM3531 devices. Compilation of the C models and host code uses GCC7.3.1 with CMSIS-NN support. CMSIS-NN is a collection of efficient neural network kernels developed by ARM in order to maximize the performance and minimize the memory footprint on Cortex-M processor cores. We also include the results

obtained with the optimizing compiler Tensai developed by Eta Compute. Fig. 13 compares the inference time of Apollo3 and ECM3531 devices using the “little” network. This shows a clear advantage for the ECM3531 when using the Tensai compiler. An important consideration is that the ECM3531 supports a larger range of frequencies compared with Apollo3. ECM3531 uses optimized frequency and voltage values between 7MHz and 62MHz. In contrast, the Apollo3 frequency selection is limited to two main levels of 48MHz and 96MHz (the latter called “boost”). We have added an additional point at 24MHz to extend this range.

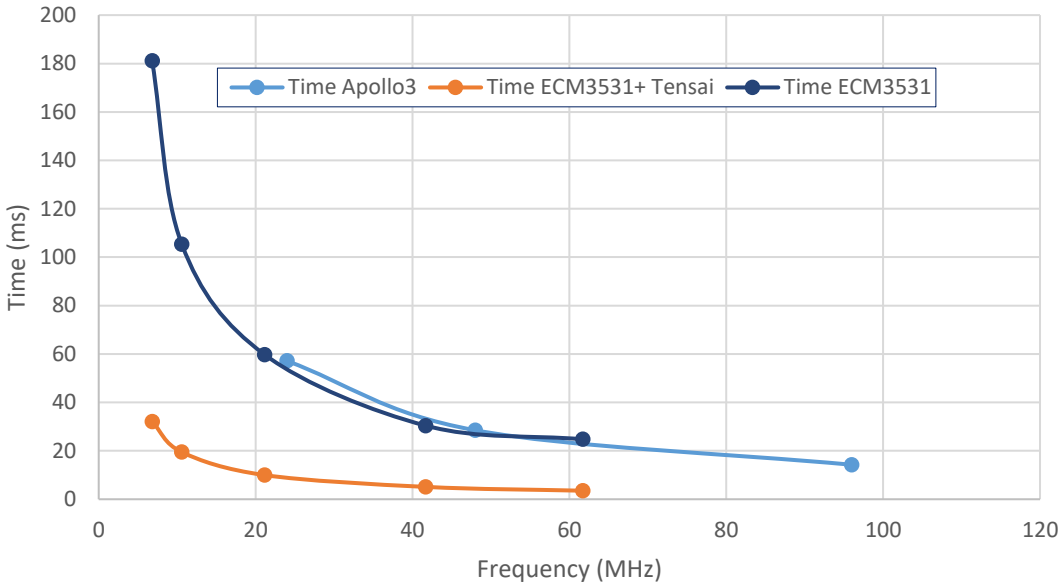

Figure 13. Inference time on near-threshold voltage MCUs

Next, we evaluate power and energy of single inferences in the “little” network, shown in Fig. 14. For the Apollo3 device, the lowest power below 1mW is achieved at the lowest frequency of 24MHz as expected, but it is also clear that energy efficiency is better at the nominal frequency of 48MHz. For the ECM3531, the lowest power is obtained at 7MHz, where the voltage regulators are configured to 600mV. When using the GCC compiler, the best energy efficiency is not achieved at the lowest power level. In contrast, when using the Tensai optimizing compiler, energy efficiency for the different frequency modes is largely unaffected. This could be explained due to the simple network that has a low operation density per byte, so energy is dominated by memory accesses. In essence, the optimizing compiler results in computation that is more memory bound and less compute bound. This means that if x operations are needed per byte brought from memory using Tensai, and y operations are needed per byte brought from memory in the case of the gcc compiler, then $y>>x$. The power consumed by the flash/SRAM memory devices present in the device does not scale with voltage but, since they are built into the device, they are being measured by the power board. In any case, we observe a low power figure at 7MHz of 0.3mW for the ECM3531, and better energy efficiency for all the frequency points than the Apollo3 device. At 7MHz the ECM3531 device can perform an inference with the “little” network every 32ms, much faster than the 2.56s required by the sensors in this sample case study. This means that, between processing samples, the processor can enter a deep sleep state in which we have measured a current of 12nA, equivalent to approximately 20nW. It is clear that with this time constraint, a much lower frequency, in the range of kHz, should be sufficient. However, the current software development kit available for the Eta Compute device does not support lower frequencies and the near-threshold logic will not operate at voltage levels below 0.5V. The low-performance requirements means that deeper sub-threshold operation could be very useful in this case but

the issues with transistor leakage need to be solved first. Alternatively, the sensor sampling rates could be increased in a real deployment to better match the performance capabilities of the platform and obtain finer granularity in the classification process. This could be particularly relevant in related applications such as machine health monitoring [31], where the state of the machine can change at much higher rate than human activity.

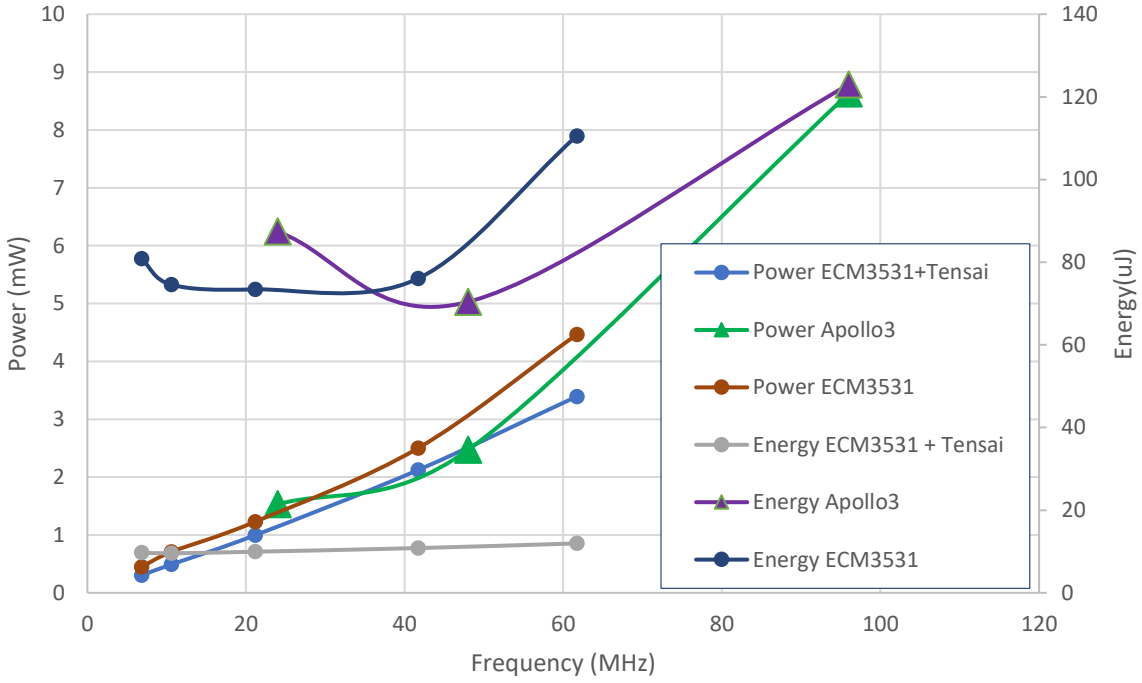

Figure 14. Power and energy in the near-threshold voltage MCUs

We have measured the NCS2 power using a power meter at approximately 1W when actively running computations. Fig. 15 compares the energy requirements of the heterogeneous solution using the ECM3531/NCS2 combination with the homogenous solution using only the NCS2. The energy usage results from performing a total of 2946 predictions over the whole data set with the ECM3531 running at its low frequency of 7 MHz and using the Tensai compiler. The configurations that include LSTM layers show higher energy costs. This is due to the increase of processing time in the NCS2 device once LSTM layers are enabled, which increases from 1.55 ms to 5.12 ms per inference. Overall, we observe that the energy cost of the heterogeneous solution is about one quarter of the homogenous solution with this sample data. The main reason is that, in this sample data, about ¾ of all inferences are done successfully in the “little” network and only ¼ require access to the NCS2 device.

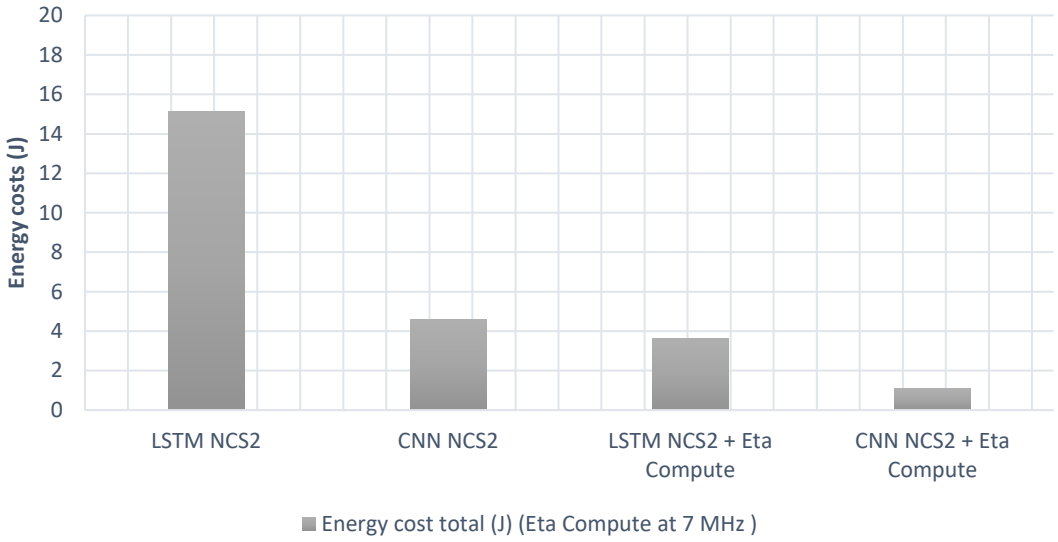

Figure 15. Energy consumption of homogenous vs heterogeneous inference solutions

8. Conclusions

Over the last few years, there have been significant advances in energy-efficient hardware, with the objective of deploying sophisticated AI applications near the edge, close to sensors, using resource-constrained devices. In this research we have evaluated a methodology based on TensorFlow Lite to create and deploy neural networks using energy-efficient state-of-the-art near-threshold processors that use sophisticated voltage scaling techniques, combined with dedicated neural accelerators. We have investigated a heterogeneous solution consisting of two devices, forming a “big”-“little” embedded system, that cooperate at solving a task by exchanging information between them. The “little” device is based on the ECM3531 Eta Compute Cortex-M3 MCU while the “big” device uses the NCS2 neural accelerator developed by Intel. In both cases, Keras and TensorFlow are used to train and generate the network parameters, with only the final optimization steps being specific to each of them, to perform tasks such as quantization to 8-bit integer for the MCU and quantization to 16-bit floating-point for the NCS2. Our application is representative of an activity detection problem that uses recurrent layers and multiple sensors, and it assumes that the activity will remain constant for a period of time before being replaced by a new activity. The results show that these different devices can work collaboratively at their optimal points, delivering accuracy, performance and better energy efficiency. Future work involves extending the work to other application areas, such as machine health monitoring. In this case the processing requirements will increase to millisecond levels and the operating point of the system could be adjusted at run-time depending on the level of monitoring granularity required. We have released our code to promote research in this field https://github.com/eejlny/subthreshold_hetero_mcu_ncs

Acknowledgements: This research was funded by the Royal Society Industry fellowship, INF\R2\192044 Machine Intelligence at the Network Edge (MINET).

[1] Jia, Z., Tillman, B., Maggioni, M., & Scarpazza, D. (2019). “Dissecting the Graphcore IPU Architecture via Microbenchmarking”. *ArXiv, abs/1912.03413*.

[2] “Silicon to Satisfy the AIoT: xcore.ai”, https://www.eejournal.com/article/silicon-to-satisfy-the-aiot-xcore-ai/ accessed October 2020.

[3] “Low-Power AI Startup Eta Compute Delivers First Commercial Chips”, https://spectrum.ieee.org/tech-talk/semiconductors/processors/lowpower-ai-startup-eta-compute-delivers-first-commercial-chips, accessed October 2020.

[4] “Ambiq Micro Achieves World-Leading Power Consumption Performance with TSMC 40ULP Technology”, https://www.design-reuse.com/news/46420/ambiq-micro-apollo3-blue-wireless-soc-tsmc-40ulp.html, accessed October 2020.

[5] “Ultra-low power and high-performance AI processor GAP8” https://greenwaves-technologies.com/gap8-the-internet-of-things-iot-application-processor/, accessed October 2020.

[6] Farhoodfar, Avid. (2019). “Machine Learning for Mobile Developers: Tensorflow Lite Framework”. https://www.researchgate.net/publication/333659766_Machine_Learning_for_Mobile_Developers_Tensorflow_Lite_Framework

[7] K. Nikov, J. L. Nunez-Yanez and M. Horsnell, "Evaluation of Hybrid Run-Time Power Models for the ARM big.LITTLE Architecture," 2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing, Porto, 2015, pp. 205-210, doi: 10.1109/EUC.2015.32.

[8] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1655-1674, Aug. 2019, doi: 10.1109/JPROC.2019.2921977.

[9] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1738-1762, Aug. 2019, doi: 10.1109/JPROC.2019.2918951.

[10] https://www.gyrfalcontech.ai/solutions/2803s/, accessed October 2020

[11] https://coral.ai/docs/edgetpu/faq/, accessed October 2020

[12] Weiss, K., Khoshgoftaar, T.M. & Wang, D. “A survey of transfer learning”. Journal of Bbig Data 3, 9 (2016). https://doi.org/10.1186/s40537-016-0043-6

[13] J. Zhu, L. Wang, H. Liu, S. Tian, Q. Deng and J. Li, "An Efficient Task Assignment Framework to Accelerate DPU-Based Convolutional Neural Network Inference on FPGAs," in IEEE Access, vol. 8, pp. 83224-83237, 2020, doi: 10.1109/ACCESS.2020.2988311.

[14] Xu, Zhe & Cheung, Ray C.C. (2020). “Binary Convolutional Neural Network Acceleration Framework for Rapid System Prototyping”. Journal of Systems Architecture. 109. 101762. 10.1016/j.sysarc.2020.101762.

[15] Lien-Chih Hsu, Ching-Te Chiu, Kuan-Ting Lin, Hsing-Huan Chou, Yen-Yu Pu, “ESSA: An energy-aware bit-serial streaming deep convolutional neural network accelerator”, Journal of Systems Architecture, Volume 111, 2020, ISSN 1383-7621, https://doi.org/10.1016/j.sysarc.2020.101831

[16] Jiang, Wei & Song, Ziwei & Zhan, Jinyu & He, Zhiyuan & Wen, Xiangyu & Jiang, Ke. (2020). “Optimized Co-Scheduling of Mixed-Precision Neural Network Accelerator for Real-

Time Multitasking Applications". Journal of Systems Architecture. 110. 101775. 10.1016/j.sysarc.2020.101775.

[17] Yizhi Liu, Yao Wang, Ruofei Yu, Mu Li, Vin Sharma, and Yida Wang. 2019. "Optimizing CNN model inference on CPUs". In Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference (USENIX ATC '19). USENIX Association, USA, 1025–1040.

[19] A. E. Eshratifar, M. S. Abrishami and M. Pedram, "JointDNN: An Efficient Training and Inference Engine for Intelligent Mobile Cloud Computing Services," in IEEE Transactions on Mobile Computing, doi: 10.1109/TMC.2019.2947893.

[19] J. Mao, X. Chen, K. W. Nixon, C. Krieger and Y. Chen, "MoDNN: Local distributed mobile computing system for Deep Neural Network," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, Lausanne, 2017, pp. 1396-1401, doi: 10.23919/DATE.2017.7927211.

[20] Z. Zhao, K. M. Barijough and A. Gerstlauer, "DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 11, pp. 2348-2359, Nov. 2018, doi: 10.1109/TCAD.2018.2858384.

[21] Amiri, S, Hosseinabady, M, McIntosh-Smith, S & Nunez-Yanez, J, 2018, 'Multi-precision convolutional neural networks on heterogeneous hardware'. in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018. Institute of Electrical and Electronics Engineers (IEEE), pp. 419-424

[22] E. Park et al., "Big/little deep neural network for ultra low power inference," 2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), Amsterdam, 2015, pp. 124-132, doi: 10.1109/CODESISSS.2015.7331375.

[23] Song, G. and W. Chai. "Collaborative Learning for Deep Neural Networks." NeurIPS (2018).

[24] Geoffrey Hinton and Oriol Vinyals and Jeffrey Dean, 2015. "Distilling the Knowledge in a Neural Network", NIPS Deep Learning and Representation Learning Workshop, arXiv preprint arXiv:1503.02531.

[25] S. Teerapittayanon, B. McDanel and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, 2016, pp. 2464-2469, doi: 10.1109/ICPR.2016.7900006.

[26] E. Flamand et al., "GAP-8: A RISC-V SoC for AI at the Edge of the IoT," 2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP), Milan, 2018, pp. 1-4, doi: 10.1109/ASAP.2018.8445101.

[27] J. Nunez-Yanez, "Energy Proportional Neural Network Inference with Adaptive Voltage and Frequency Scaling," in IEEE Transactions on Computers, vol. 68, no. 5, pp. 676-687, 1 May 2019, doi: 10.1109/TC.2018.2879333

[28] "Is DVFS worth the effort", https://semiengineering.com/is-dvfs-worth-the-effort/, accessed October 2020

[29] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. “A Public Domain Dataset for Human Activity Recognition Using Smartphones”. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013

[30] Ordóñez, F.J.; Roggen, D. “Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition”. Sensors 2016, 16, 115.

[31] Y. Zhang, P. Hutchinson, N. A. J. Lieven and J. Nunez-Yanez, "Remaining Useful Life Estimation Using Long Short-Term Memory Neural Networks and Deep Fusion," in IEEE Access, vol. 8, pp. 19033-19045, 2020, doi: 10.1109/ACCESS.2020.2966827.