Journal of
Low Power Electronics and Applications

MDPI

文章低功耗近亚阈值处理器上的大小自适应神经网络

Zichao Shen [1,*], Neil Howard [2] and Jose Nunez-Yanez [1]

[1] 布里斯托大学电气与电子工程系，英国布里斯托尔，BS8 1UB；{zs16916, j.l.nunez-yanez}@bristol.ac.uk[2] Sensata Systems，Interface House，英国斯温顿，SN4 8SY；njhoward.mobile@gmail.com* 通讯作者：zs16916@bristol.ac.uk

arXiv:2304.09695v1 [cs.LG] 19 Apr 2023

摘要： 本文研究近亚阈值处理器在边缘人工智能应用中能够实现的节能效果，并提出在保持应用准确率的同时提升节能效果的策略。所选处理器采用自适应电压缩放技术，在该技术中，处理器核心的频率和电压水平由运行时决定。在这些系统中，为了节省功率，嵌入式随机存取存储器和闪存的容量通常限制在 1 兆字节以下。存储器容量有限，限制了可映射到这些设备上的神经网络模型的复杂度，也使准确率与电池寿命之间必须作出权衡。针对这些问题，我们提出并评估了多种“大—小”神经网络策略，以在保持预测准确率的同时延长电池寿命。这些策略应用于人体活动识别任务，并以此作为演示案例。结果表明，与原始网络相比，最佳配置在保持原有推理准确率的同时，能量消耗可降低 80%。

关键词： 近亚阈值处理器；能效；边缘计算；神经网络；自适应计算

check for updates

Citation: Shen, Z.; Howard N.; Nunez-Yanez, J. Big–Little Adaptive Neural Networks on Low-Power Near-Subthreshold Processors. *J. Low Power Electron. Appl.* **2022**, *12*, 28. https://doi.org/10.3390/jlpea12020028

Academic Editor: Weidong Kuang

Received: 10 March 2022
Accepted: 12 May 2022
Published: 18 May 2022

P出版者注： MDPI 保持中立
w 对于已发表地图中的管辖权声明以及机构隶属关系。

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. 引言

过去几十年来，物联网（IoT）和深度学习的快速发展，增加了将深度神经网络（DNN）部署到低功耗设备上的需求 [1]。由于延迟较高且存在隐私问题，图像识别和自然语言处理等领域的云计算任务正逐渐转移到边缘端 [2]。存储器容量和计算能力的限制意味着，拥有数百万个参数的大型神经网络难以轻松部署在微控制器单元（MCU）等边缘设备上；在许多情况下，这类设备的闪存容量不足 1 兆字节 [1,2]。为了节省成本并降低功耗，设备会将存储器容量保持在较低水平，因为这些设备不具备对未使用的存储器模块进行电源门控的功能。

最大限度延长设备的使用时间是一项重要目标。围绕这一目标，我们研究了一种自适应大小神经网络系统，该系统由一个 big 网络和多个 little 网络组成，通过限制 big 网络的执行次数，在不降低准确率的情况下实现节能推理。我们将这种架构称为“大小”架构，因为它借鉴了 ARM 推广的 “ 大小 ” 技术，该技术将复杂处理器和轻量级处理器整合在同一个片上系统中。big 网络的准确率更高，但推理时间更长；little 网络的推理速度则更快。大多数时候，big 网络都保持在睡眠模式，只有当 little 网络判断自己无法以所需的置信度处理任务时，才会将其激活。

本研究重点是在边缘设备上构建并部署完整的自适应神经网络系统。我们以人体活动识别（HAR）应用这一常见的边缘应用为例，探讨如何管理主网络和次级网络，从而实现更快、更准确且能效更高的性能。本研究的贡献概述如下：

- 我们评估了采用自适应电压缩放的先进近阈值处理器，并将其与标准边缘处理器进行了比较。
- 我们针对基于*TensorFlow* 的人体活动识别（人体活动识别）模型，使用不同厂商的工具链和编译器，对一种热门边缘应用进行了优化，以实现微控制器部署。
- 我们提出了适用于自适应神经网络系统的新型“大–小”策略，以实现快速推理并节省能量。
- 我们已将研究成果开源，网址为https://github.com/DarkSZChao/Big-Little_神经网络_策略（访问日期：2022年3月9日），以进一步促进该领域的研究工作。

本文结构安排如下：第2节概述低功耗边缘 AI 的先进硬件、框架及相关算法技术。随后，第3节对近阈值微控制器和标准MCU的性能与能量成本进行了初步评估。第4节提出并评估了三种具有不同特性和性能特点的自适应神经网络系统配置。第5 节介绍并展示了面向所选低功耗微控制器所需的实现步骤。第6节呈现了速度、准确率和能量方面的结果。最后，第7节讨论结论与未来工作。

2. 背景与相关工作

本节概述适用于边缘人工智能、功率约为 1 瓦或更低的当前先进硬件，随后介绍针对这类硬件优化的算法技术和框架。

2.1. 低功耗边缘 *AI* 硬件

边缘人工智能应用的高需求推动了针对低功耗水平进行优化的硬件显著增加。例如，Google推出了名为Edge TPU的轻量版张量处理单元（TPU），能够以每瓦每秒2万亿次乘加运算（2TMAC/s/W）提供高能效推理 [3]。这款先进设备能够以接近400 FPS的速度运行MobileNet V2等移动版模型。Cloud TPU专注于复杂模型的训练，而Edge TPU则设计用于在低功耗系统中执行推理。为实现显著低于Edge TPU的功耗，Ambiq推出了基于32位ARM Cortex-M4F处理器的Apollo系列近阈值处理器。这些设备能够实现低得多的能耗：在工作模式下、3.3 V电压时仅为6 µA/MHz，在睡眠模式下、3.3 V电压时仅为1 µA/MHz。SparkFun开发板中的Apollo3设备配备了1 MB闪存和384 KB低漏电RAM [4]。同样，Eta Compute也利用ECM3532处理器，致力于提供能效出色的端点人工智能解决方案。该设备基于ARM Cortex-M3 32位CPU，并配备独立的CoolFlux DSP，以更加节能的方式加速机器学习运算。人工智能视觉开发板中的ECM3532在正常工作模式下的功耗低于5 µA/MHz，在睡眠模式下为1 µA/MHz。据Eta Compute介绍，其自定时连续电压与频率缩放技术（CVFS）实现的功率曲线仅为1 mW [5,6]。这些近阈值设备的一个特点是，电压缩放应用于核心，但由于存储单元可实现的裕量有限，并不应用于设备的SRAM/闪存。

Apollo3和ECM3532都基于广泛应用的ARM架构，但近年来，开源指令集架构RISC-V也在这一领域受到广泛关注。例如，GreenWaves Technologies开发的GAP8配备了由8个RISC-V处理器组成的计算集群，以及一个额外的CNN加速器 [7]。该计算集群还连接了一个额外的超低功耗微控制器，其状态保持睡眠功耗为30 µW，用于控制和通信功能。对于CNN推理（90 MHz，1.0 V），GAP8可实现600 GMAC/s/W的能效，最坏情况下的功率范围为75 mW [7]。

其他探索近阈值工作区的公司还包括 Minima，该公司参与了多项设计，展示了可实现的功率节省效果 [8]。Minima 提供超宽动态电压频率调节（DVFS），可根据工作负载调节频率和/或工作电压。结合 Minima 和 ARM 采用的动态裕量调节方法，这种方案最多可节省 15× 至20× [9] 的能量。对自适应电压调节硬件的兴趣促成了一个由意法半导体牵头、金额为 €100 m 欧元的欧洲项目，旨在利用低功耗 FD-SOI 和相变技术开发新一代边缘 AI 微控制器及软件。该项目旨在为汽车和工业市场提供芯片组及解决方案，实现每瓦 10 TOPS 的超高计算能力，性能远超现有微控制器 [10]。

2.2. 低功耗边缘 *AI* 的算法技术

多年来，为优化边缘设备上的推理，出现了不同的算法方法，重点关注量化、剪枝、异构模型和提前终止等技术。对网络权重和激活值进行深度量化，是优化边缘部署网络模型的一种广为人知的方法 [11,12]。示例包括 [13],，其对权重和激活值采用极低精度（例如 1 位或 2 位），在 AlexNet 中实现了 51% 的 top-1 准确率和 7 倍加速 [13]。 [14] 的作者展示了一种二值化神经网络（BNN），其中权重和激活值均经过二值化。在前向传播过程中，BNN 大幅减少内存访问，并以按位运算替代大多数算术运算。文献 [14] 证明，通过使用其二值矩阵乘法内核，在 MNIST、CIFAR-10 和 SVHN 数据集上，结果实现了 32 倍压缩率，并将性能提升了 7 倍。然而， [15]观察到准确率大幅下降（最高达 28.7%）。 [15] 中的研究通过在 XNOR-Net 的每个卷积层之前部署一个全精度归一化层，解决了这一缺陷。XNOR-Net 对输入和卷积层权重均采用二值，并能够将计算工作负载降低约 58 倍，同时在 ImageNet 上仅损失 10% 的准确率 [15]。总体而言，这些网络能够使边缘设备摆脱整数运算所导致的繁重工作负载，但必须妥善管理准确率损失。CoopNet [16]进一步降低了准确率损失。与 [17], 中的多精度 CNN 概念类似，CoopNet [16]采用两个卷积模型：推理速度更快的二值网络 BNN，以及准确率相对较高的 INT8 整数网络，以平衡模型的效率和准确率。在 RAM 有限的低功耗 Cortex-M 微控制器上（≤ 1 MB），文献 [16] 实现了约 3 倍的压缩率和 60% 的加速，同时在 CIFAR-10、GSC 和 FER13 数据集上保持了较高的准确率。与 CoopNet 为主网络和次级网络采用相同网络结构不同，我们为次级网络采用了更加简单的结构，其中每个网络都经过训练，用于识别 HAR 任务中的一个类别。这一优化形成了一种配置，在所有经过评估的 MCU 平台上都能以相近的准确率实现约 80% 的加速和节能。基于 XNOR-Net，文献 [18] 构建了一种剪枝-置换-打包网络，将二值化与稀疏性相结合，将模型大小压缩至极低水平。在 Nucleo 平台上

以及树莓派上，3PXNet 将模型大小最多减少至 38×，并将运行时间和能耗提升 25×；与已经较为紧凑的传统二值化实现相比，其准确率下降不到 3%。TF-Net 是一种替代方法，为深度神经网络模型选择三值权重和四位输入。文献 [19] 提供了这种配置，以在微控制器上实现模型准确率、计算性能和能效之间的最佳平衡。他们还解决了这样一个问题：由于内存按字节寻址，三值权重和四位输入无法直接访问，因此需要在计算前从比特流中解包这些值。在配备 ARM Cortex-M4 的 STM32 Nucleo-F411RE 微控制器上，文献 [19] 将计算性能和能效分别提升了 1。83× 和 2。28×。因此，可以认为 3PXNet/TF-Net 与我们的“大-小”研究正交，因为它们可以作为本研究所考虑的 8 位整数模型的替代方案。与我们的方法相关的一种架构 BranchyNet 及其提前退出机制，在 [20]中提出。该架构具有多个退出点，可减少逐层权重计算和输入/输出成本，从而实现更快的推理速度并节省能量。然而，由于存在多个分支，它包含数量庞大的参数，这将显著增加边缘设备的存储需求。

近年来，人们提出了主网络和次级神经网络的配置，以加速边缘设备上的推理过程。参考文献 [17,21] 构建了具有相同输入和输出数据结构的‘big’和‘little’网络。‘big’网络由‘little’网络生成的评分指标触发。 [22], 也提出了类似的配置，但其‘big’和‘little’网络是独立训练的。‘Big’和‘little’网络不共享相同的输入和输出数据结构。参考文献 [22] 提出了一种异构配置，在 NCS2 等先进的边缘神经网络加速器上部署‘big’网络，同时在 ECM3531 和 Apollo3 等近阈值处理器上部署‘little’网络。参考文献 [22] 通过在‘big’和‘little’网络之间切换这一异构系统，在人体活动分类任务上成功实现了 93% 的准确率，能耗约为 4 J。参考文献 [22] 考虑了异构硬件，而我们的方法采用‘big–little’概念，但侧重于在单个微控制器设备上部署所有模型。与 [22]分别在 NCS2 硬件加速器和近阈值处理器上部署‘big’和‘little’模型的做法不同，我们在近阈值MCU上部署两个神经网络模型，用于活动分类任务。我们设置了一种切换算法，在‘big’和‘little’网络模型之间切换，从而大幅降低能耗，同时保持相近的准确率。相关研究 [23] 利用卷积层和长短期记忆（LSTM）层执行活动识别任务，取得了出色的准确率和性能。由于微控制器的闪存容量有限，我们决定不使用如 [23]所示具有数百万个参数的 LSTM 层。所提出的自适应系统适用于现实世界的任务，例如人体活动分类，因为活动不会以极高的速度发生变化。一个人会持续进行某项动作一段时间，通常为几十秒 [24], ，这意味着没有必要让系统始终以满负荷运行（使用主‘big’网络执行推理）。由于主网络会增加推理时间和计算开销，因此主网络调用次数越少，推理过程就越快，能耗需求也越低 [16,17,21,22]。

2.3. Frameworks for Low-Power Edge AI

过去几年中，出现了许多框架，用于简化资源受限的边缘设备上的神经网络模型部署。在 [25], 中，提供了一种名为 FANN-on-MCU 的框架，专门用于在低功耗微控制器上快速部署多层感知机（MLP）。该框架不仅支持非常流行的 ARM Cortex-M 系列微控制器，还支持 RISC-V 并行超低功耗（PULP）处理器。结果

表明 [25] 基于 PULP 的“Mr.Wolf”片上系统相较于单核实现，最高可实现 71× 倍加速，相较于 ARM Cortex-M4 则可实现 135× 倍加速。此外，借助 FANN-on-MCU，一个包含 103,800 次乘加运算的相对较大的神经网络，可以在搭载单个 ARM Cortex-M4 的 Nordic nRF52832 微控制器上于 17.6 毫秒内执行完毕，能耗为 183 µJ。在“Mr.Wolf”上使用八个基于 RISC-V 的 RI5CY 核心运行同一个神经网络时，耗时不到 1 毫秒，能耗约为 50 µJ [25]。与 FANN-on-MCU 类似，参考文献 [26] 提供了一个名为微控制器上的神经网络（*NNoM*）的快速微控制器框架部署方案，该框架支持来自 Keras 的 ResNet 和 DenseNet 等更复杂的模型拓扑结构。该框架为嵌入式开发者构建了易用的 API 和高性能后端选项，用于在低功耗微控制器设备上部署 Keras 模型。此外，一些商业公司也开发了面向低功耗边缘设备的部署框架。例如，Google 借助广受欢迎的 *TensorFlowLite* 框架 [27]。结合模型训练框架 *TensorFlow*，Google 能够提供从神经网络模型训练到边缘设备模型部署的完整解决方案。 *STM32Cube.AI* 由意法半导体提供 [28]，同样是一种人工智能部署框架，但它仅针对 STM 系列设备进行设计，例如 STM32 Nucleo-L4R5ZI 和 STM32 Nucleo-F411RE。Eta Compute 创建了 *TENSAIFlow*部署框架，为 Eta 系列微控制器产品（例如 ECM3531 和 ECM3532 [29]）提供性能和效率优化。在我们的方法论中，由于某些框架缺乏对特定设备的支持，我们结合使用了不同供应商提供的工具。我们采用了来自 [26,27,29] 的框架，用于在 ECM3532 和 STM32L4 等微控制器上进行模型部署（详见第 5节）。

3. 低功耗微控制器评估

本文对意法半导体、Ambiq 和 Eta Compute 推出的四款面向节能应用的商用微控制器设备进行了比较。表1 列出了这四款微控制器的技术细节。其中三款（STM32L4R5ZI、Apollo2 Blue 和 SparkFun Edge（Apollo3 Blue））基于带浮点运算单元（FPU）的 Cortex-M4 微架构 [4,30,31]，，而 ECM3532 则基于搭载“CoolFlux”16 位数字信号处理器的 Cortex-M3 微架构 [5]。32 位 ARM Cortex-M3 和 M4 是相近的微架构，均采用三级流水线并实现 Thumb-2 指令集，但可用指令数量有所不同。例如，额外的 16/32 位乘加指令和单精度 FPU 仅适用于 Cortex-M4。

采用搭载 STM32L4R5ZI 微控制器的 STM32 Nucleo-144 开发板作为比较对象；这款 STM 设备与其他三款设备的主要区别在于功率优化方法。STM 设备的内核供电电压为 1 V，显著高于 Ambiq 和 Eta Compute 的近阈值设备，其内核电压仅约为 0.5 V。理论上，亚阈值内核供电电压可低至 0.3 V，因而能效更高。然而，在 0.3 V 下，晶体管开关时间会更长，从而导致漏电流增大。当阈值电压约为 0.2 V 时，漏电功率可能超过总功耗的 50% [32]。因此，实践表明，选择近阈值电压点而非亚阈值电压点，是一种能效更高的方案 [32]。为根据任务需求优化能量使用，STM32L4 采用标准动态电压和频率缩放（DVFS），使用预定义的电压与频率数据对；而 Ambiq 和 Eta Compute 的设备则采用自适应电压缩放（AVS），通过反馈环路确定给定频率下的电压，以便在运行时处理任务 [33]。

对比数据手册可知，STM32L4 的时钟频率最高，因此在处理速度方面具有优势。Ambiq 和 EtaCompute 的近阈值设备所需的内核供电电压仅约为 STM32L4 的一半。所有纳入考量的处理器均

配备了容量有限的闪存，大小为 0.5 MB 至 1 MB，静态随机存取存储器（SRAM）约为 300 KB。这意味着部署的神经网络模型必须足够小，才能适应有限的存储器容量。因此，我们使用 *TensorFlow* 框架和 *TensorFlow Lite* 转换器，创建一个用于人体活动识别（HAR）的简单预训练卷积神经网络模型，该模型基于 UCI [34]（如图 1所示），用于对四种微控制器设备进行初步能量评估。

Table 1. Low-power MCU comparison.

设备	制造商	架构	Embedded Memory (Flash/SRAM/-缓存	时钟频率	内核电压	睡眠模式电流	工作模式电流	Power Scaling功能
STM32L4R5ZI	STMicroelectronics	32-bit Cortex-M4 CPU with FPU	1 MB/320 KB	up to 120 MHz	1.05 V	<5 µA with RTC	110 µA/MHz	Dynamic voltage scaling with two main voltage 范围
Apollo2 Blue	Ambiq	32-bit Cortex-M4 CPU with FPU	1 MB/256 KB/16 KB	up to 48 MHz	0.5 V	<3 µA with RTC	10 µA/MHz	SPOT (Subthreshold Pow Optimized Technology)
SparkFun Edge (Apollo3 Blue)	Ambiq	32-bit Cortex-M4 CPU with FPU	1 MB/384 KB/16 KB	up to 96 MHz (with TurboSPOT Mode)	0.5 V	1 µA with RTC	6 µA/MHz	SPOT (Subthreshold Pow Optimized Technology) with TurboSPOT
ECM3532	Eta Compute	32-bit Cortex-M3 CPU with 16-bit 'CoolFlux' DSP	512 KB/256 KB/0 KB	up to 100 MHz	0.55 V	1 µA with RTC	5 µA/MHz	CVFS (Continuous Voltag Frequency Scaling) with multiple frequency and voltage points

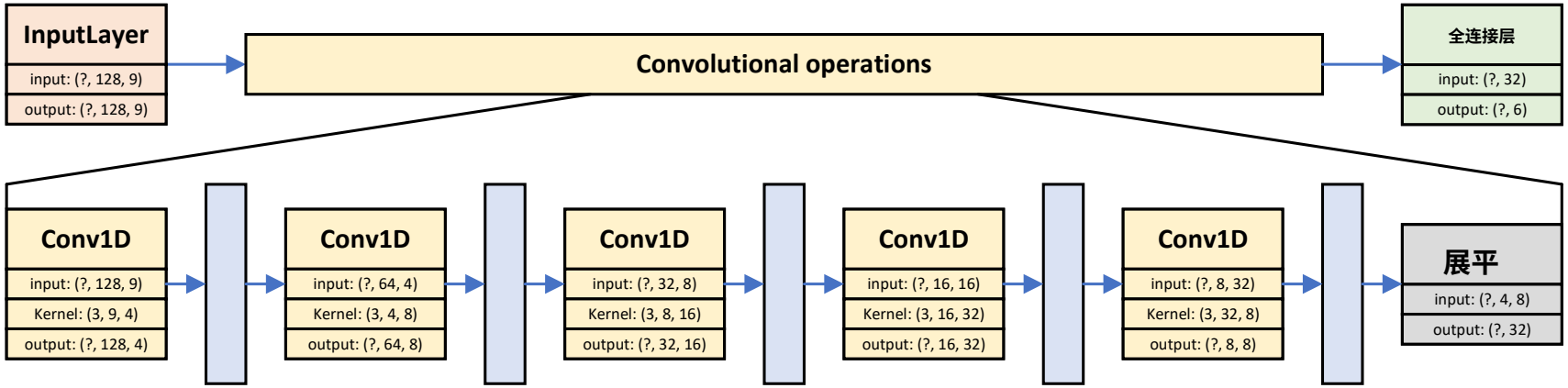


Figure 1. Convolutional neural network for the initial performance and energy evaluation of the MCU.

The energy board X-NUCLEO-LPM01A from STMicroelectronics is used to evaluate the performance and energy consumption measuring the current used by the target board under a given supply voltage of 3.3 V (lower core voltages are regulated internally in the device). The power consumption of the four tested boards is shown in Figure 2. STM32L4 operates at a much higher power level which is around six times that of the near-threshold processors. The near-threshold processors Apollo2, Apollo3 and ECM offer significantly lower power, consuming less than 5 mW at the normal frequency of 48MHz and around 10 mW in the burst mode of 96 MHz. The reason why SparkFun Edge (Apollo3) consumes more power than Apollo2 is that the Apollo3 core is highly integrated into the SparkFun Edge board with peripheral sensors and ports which cannot be disabled during the power evaluation. Therefore, the peripheral devices on SparkFun Edge (Apollo3) are responsible for a component of the power consumption, which leads to a higher power than Apollo2 at each frequency level. Apollo2 and ECM3532 share a similar level of power consumption at 24 and 48 MHz. Apollo2 does not support running at a frequency higher than 48 MHz; therefore, there is no value for Apollo2 at the 96 MHz frequency point.

图 3 展示了四款受测处理器对图 1中的预训练卷积神经网络模型进行一次推理的执行时间。Apollo2 速度最慢，在 24 兆赫频率下需要超过 100 毫秒，在 48 兆赫下约需 50 毫秒才能完成推理。与 Apollo2 相比，SparkFun Edge 开发板（Apollo3）的执行时间缩短了约 40%。在突发模式（96 兆赫）下运行时，执行时间甚至可降至 20 毫秒以下。由于 STM32L4 的内核供电电压较高（见表 1 ），晶体管开关和处理速度更快，因此它在所有设备中速度排名第二。ECM3532 的执行时间最短，在 24 兆赫、48 兆赫和 96 兆赫下分别为 28 毫秒、15 毫秒和 8 毫秒。ECM3532 设备的显著优化归功于 *TENSAIFlow*编译器。

图 4展示了使用 X-NUCLEO-LPM01A 能量测量板测得的能耗值。由于图 2 中的标准微控制器 STM32L4 的功耗是近阈值微控制器的六倍，而且在相同频率下，其处理速度并无明显优势，因此，在 24 至 96 兆赫的所有工作频率下，STM32L4 在能耗方面都是表现最差的设备。SparkFun Edge（Apollo3）在 24 和 48 兆赫下的能耗略高于 Apollo2，这是由于板载外围设备消耗了额外的能量。在能耗测试中，ECM3532 在常规频率点（24 和 48 兆赫）下实现了最低能耗，这是因为它在功耗和时间评估中均取得了更好的结果。然而，在 96 兆赫突发模式下运行时，ECM3532 需要更高的功率来获得更快的处理速度，从而导致能耗略有增加。SparkFun Edge 开发板也呈现出相同的情况。

Overall, compared to the STM32L4 reference point all three near-threshold MCUs have a significant advantage in power and energy consumption which is around 80% to 85% lower. Although the near-threshold MCUs are comparable with the standard MCU STM32L4 in terms of inference time, their lower core voltage supplies (Table 1) result in lower power (Figure 2) at the same frequency level. Therefore, in our model inference evaluation, the near-threshold MCU devices can achieve better results in energy consumption compared to STM32L4 at 24, 48 and 96 MHz. Thanks to the additional model optimization obtained with the *TENSAIFlow* compiler provided by Eta Compute, ECM3532 offers a good balance between performance and energy efficiency to reach a lower execution time, enabling the lowest energy consumption for model inference from 24 to 96 MHz. In contrast, Apollo2, with a relatively slow processing speed, needs more time for model inference, which leads to higher values in energy consumption at 24 and 48 MHz. Due to the energy consumed by the inaccessible peripheral equipment on SparkFun Edge (Apollo3), this device consumes higher energy than Apollo2 (Figure 4).

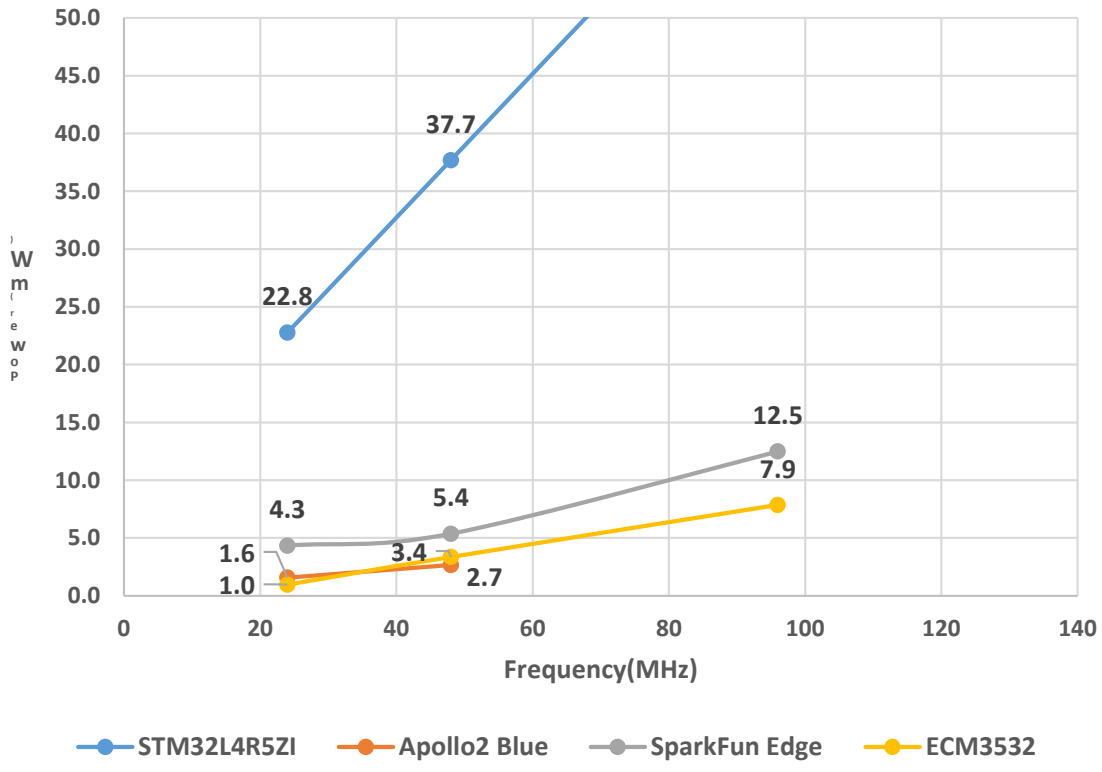


Figure 2. MCU initial evaluation in terms of power consumption.

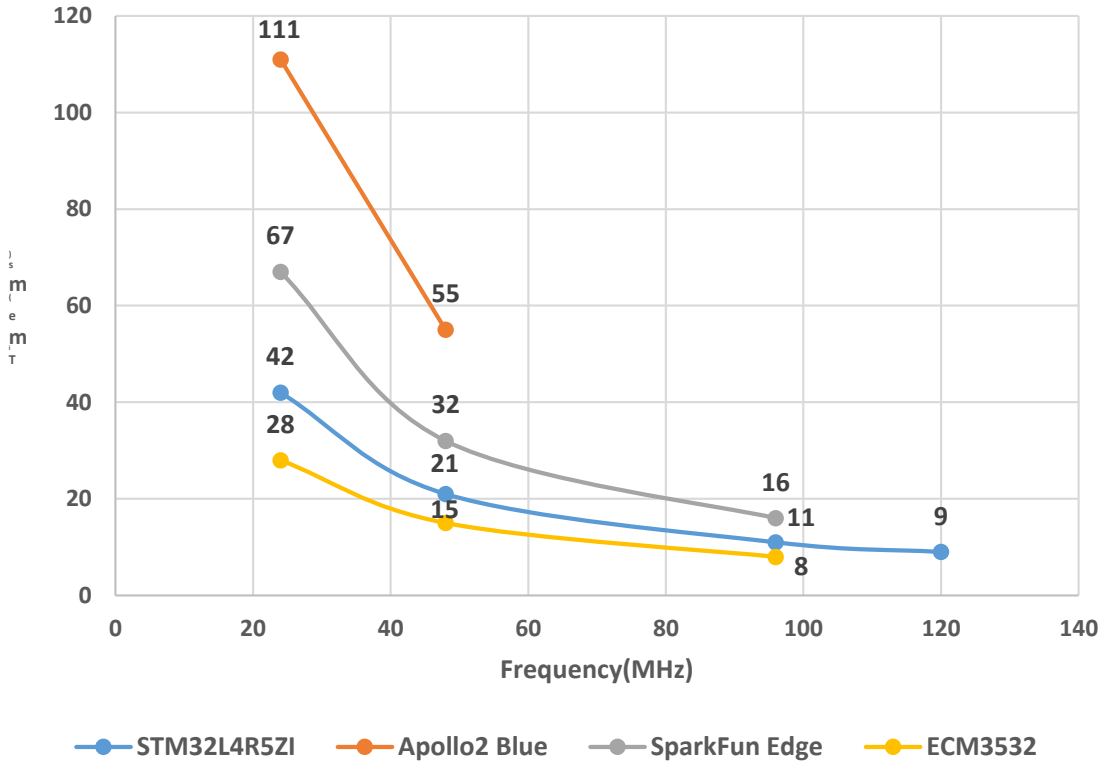


Figure 3. MCU initial evaluation in terms of time cost.

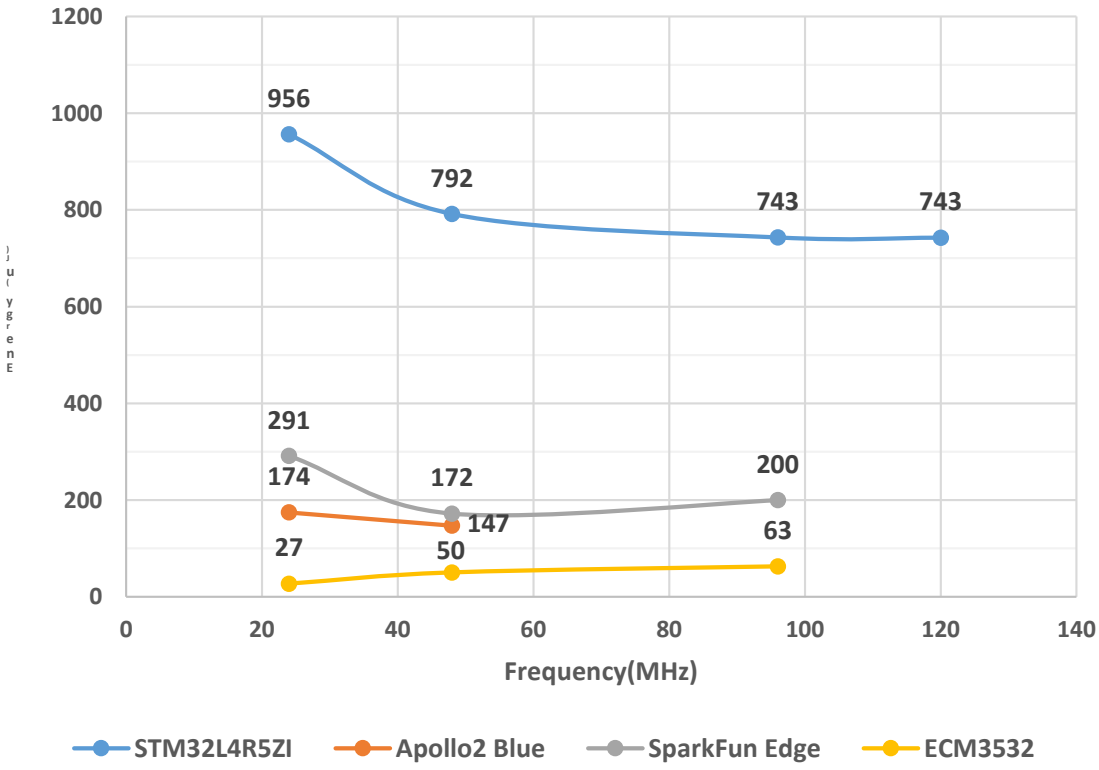


Figure 4. MCU initial evaluation in terms of energy consumption.

4. Adaptive Neural Network Methodology

To create the adaptive neural network system, we employ Python version 3.6.8 and *TensorFlow* 1.15 with its dependencies installed on a desktop PC with Intel(R) Core (TM) i7-10850H CPU 2.70 GHz, NVIDIA GeForce MX250 GPU, and 16 GB RAM. There are several framework alternatives to train the neural networks, such as PyTorch and Caffe. Due to the reasons of MCU compatibility and stability, our approach uses *TensorFlow* 1.15 to train the primary and secondary network models. After that, we use *TensorFlow Lite* and *NNoM* Converter to convert the models using single-precision floating-points (FP32) to the unsigned integer 8-bit (UINT8) format which can be deployed on the MCUs.

We consider human activity recognition using the UCI data set [34] as our raw data set. This application is a demonstrator which assumes that the activity will remain constant for a short period of time before being replaced by the next activity. To save energy via a reduction in execution time, we propose the adaptive neural network system which is able to disable the primary model and activate a secondary model when the activity remains unchanged. Therefore, we aim at achieving both latency and energy reductions without affecting prediction accuracy.

UCI-HAR 数据集使用身体加速度计、身体陀螺仪和总加速度计的三个轴，为 30 名志愿者执行的六种动作（坐着、站立、躺、步行、步行_上楼，以及步行_下楼）提供身体信息。所有数据均以固定宽度的滑动窗口进行采样，每个窗口包含 128 个采样点，并随机划分为两个集合，其中 70% 的数据样本用于训练，30% 用于测试。因此，训练数据的形状为 (7352, 128, 3, 3)，测试数据的形状为 (2947, 128, 3, 3)。我们将这三个传感器的测试数据应用于次级网络，得到的准确率如图 5 所示。总加速度计传感器的总体准确率最高。因此，选择该传感器用于次级网络推理。UCI-HAR 的训练数据集和测试数据集使用

取值范围较广的浮点值，因此在训练模型之前，所有数据都已重新缩放为量化整数值，取值范围为 [−128, 127]。在以下各节中，这些动作分别称为：

- Activity I = WALKING
- Activity II = WALKING_UPSTAIRS
- Activity III = WALKING_DOWNSTAIRS
- Activity IV = SITTING
- Activity V = STANDING
- Activity VI = LAYING

In principle, if the system uses the 'big'-only configuration, then the accuracy will be higher. The challenge is to reduce the 'big' activation count without reducing accuracy. To achieve this, we propose the three alternative configurations shown below:

- 'Big'-only (original method)
- 'Big' + six 'little'
- 'Big' + 'dual'
- 'Big' + distance.

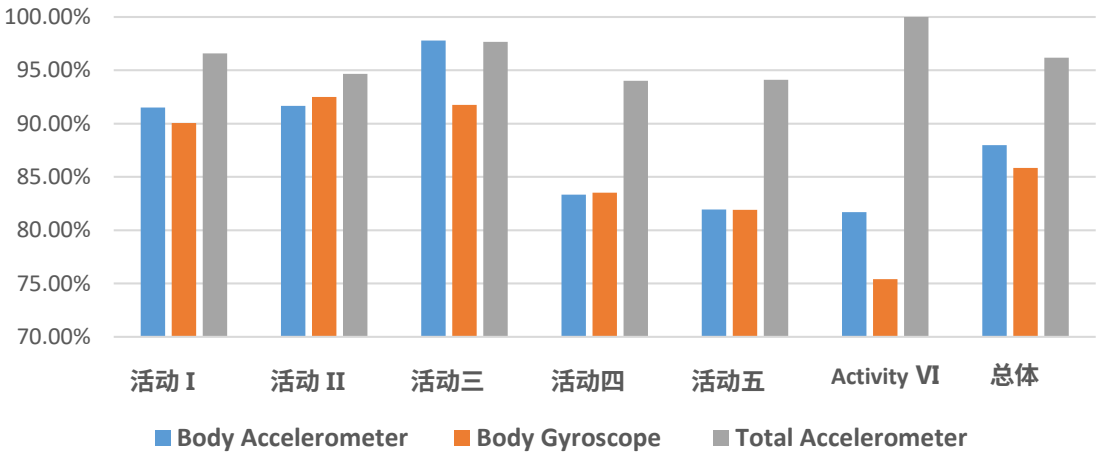


图 5. 次级网络上三个传感器的测试准确率。选择总体准确率最高的总加速度计传感器用于次级网络。

4.1.「大」 +「小」 配置

主要思路是在微控制器设备中部署一个由一个'大'（输出六个类别）和六个'小'（每个输出两个类别）网络模型组成的自适应系统。如图 6所示，一旦'大'网络提供一个类别并将其存储在寄存器中，相应的'小'网络就会被触发，以推理下一个传入的活动。'小'网络负责检测当前活动是否属于'大'网络提供的上一个类别。例如，在通过触发'大'网络将第一个活动检测为坐着后，第二个传入的活动数据将由'小'网络处理，该网络能够区分第二个活动是否仍然是坐着。一旦下一个传入的活动不再是坐着，'大'网络将再次切换开启。否则，'小'网络将保持活动状态，以节省功率和推理时间。因此，我们有六个'小'网络，这与'大'网络中分类的类别数量相同。

图 7 展示了'big'网络和'little'网络的模型拓扑结构，表2则列出了网络层的参数详情。由于我们使用了来自三个传感器的所有数据

对 UCI-HAR 数据集中的六项活动进行分类，‘big’网络有三个输入，总参数量约为 9000。Keras 的一维卷积层和最大池化层堆叠在一起，构成图 7中的三个‘big’分支。随后，这些分支的输出通过拼接层汇合，再接入一个包含六个神经元的全连接层，以对应六个类别。每个传感器的数据形状为 (7352, 128, 3)，这意味着我们有 7352 个数据样本，每个轴的长度为 128。在‘big’网络的训练和测试过程中，数据集的标签为 0 到 5，分别表示各项活动。

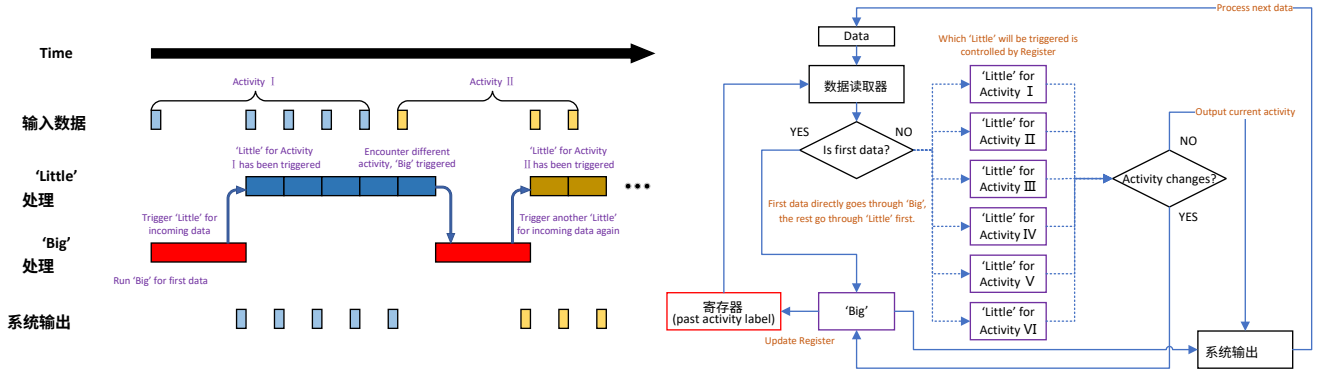


图6. 自适应神经网络系统“大” + 六“小”配置的处理步骤（左）和流程图（右）。左图中，深蓝色和棕色分别表示与输入活动相对应的两个“小”网络模型。右图中的虚线表示六个“小”网络模型中每次只调用一个。

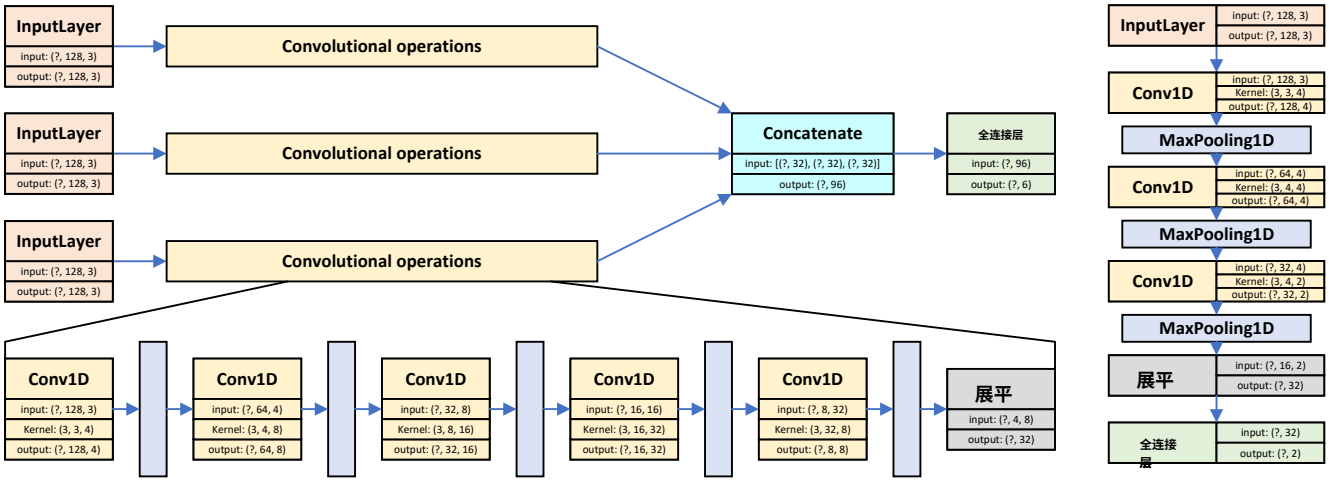


图7. Keras 中的“大”（左）和“小”（右）模型结构。

<escape>每个‘little’网络仅通过使用若干一维卷积层和最大池化层对两个类别进行分类，其总参数量为 184。因此，根据图 5中的结果，仅选择总体准确率最高的总加速度计传感器作为‘little’网络的输入。‘little’网络的输出为一个包含两个神经元的全连接层，用于表示两个类别，如表2所示。由于 UCI-HAR 数据集的规模有限 [34], , 每种活动类别的数据元素少于 2000 个。因此，我们使用所有这些数据元素，并将数据标签从六个类别转换为两个类别，以训练‘little’模型。具体而言，对于每个‘little’模型，将对应活动的标签设为数字 1，而其他活动的标签均设为数字 0。最后，我们可以生成 Keras 格式的模型。</escape>

Table 2. 'Big' (**left**) and 'little' (**right**) model parameter details. The pooling layers are hidden. For more info, see Figure 7.

Model: 'Big'			**Model: 'Little'**		
Layer (Type)	**输出形状**	**参数数量**	**Layer (Type)**	**输出形状**	**参数数量**
model_input1	[(None, 128, 3)]	0	model_input	[(None, 128, 3)]	0
model_input2	[(None, 128, 3)]	0	conv1d	(None, 128, 4)	40
model_input3	[(None, 128, 3)]	0	conv1d_1	(None, 64, 4)	52
conv1d	(None, 128, 4)	40	conv1d_2	(None, 32, 2)	26
conv1d_5	(None, 128, 4)	40	model_output	(None, 2)	66
conv1d_10	(None, 128, 4)	40			
conv1d_1	(None, 64, 8)	104			
conv1d_6	(None, 64, 8)	104			
conv1d_11	(None, 64, 8)	104			
conv1d_2	(None, 32, 16)	400			
conv1d_7	(None, 32, 16)	400			
conv1d_12	(None, 32, 16)	400			
conv1d_3	(None, 16, 32)	1568			
conv1d_8	(None, 16, 32)	1568			
conv1d_13	(None, 16, 32)	1568			
conv1d_4	(None, 8, 8)	776			
conv1d_9	(None, 8, 8)	776			
conv1d_14	(None, 8, 8)	776			
concatenate	(None, 96)	0			
model_output	(None, 6)	582			
Total params: 9246			Total params: 184		

4.2. 'Big' + 'Dual' Configuration

"大" + "双"配置是自适应神经网络系统的一种替代方法。我们用一个称为"双路"的小型神经网络替换六个"小"模型。与"大" + 六个"小"模型相比，该模型仅由一个主网络模型和一个次级网络模型组成，而不是六个 + 网络。为了用一个"双路"网络替换针对六个类别设计的六个"小"网络，需要将先前活动的数据样本存储在寄存器中，并与当前活动的数据样本进行比较，如图8所示。随后，"双路"网络可以识别这些模式，以区分当前活动是否发生变化。例如，第一个活动被"大"网络分类为站立，而第二个坐着活动则与"双路"网络先前存储的活动进行比较。如果"双路"网络检测到这两种活动并不相同，则会触发"大"网络进行进一步推理。否则，"双路"网络将继续保持激活状态，以节省时间和能量，如图8所示。

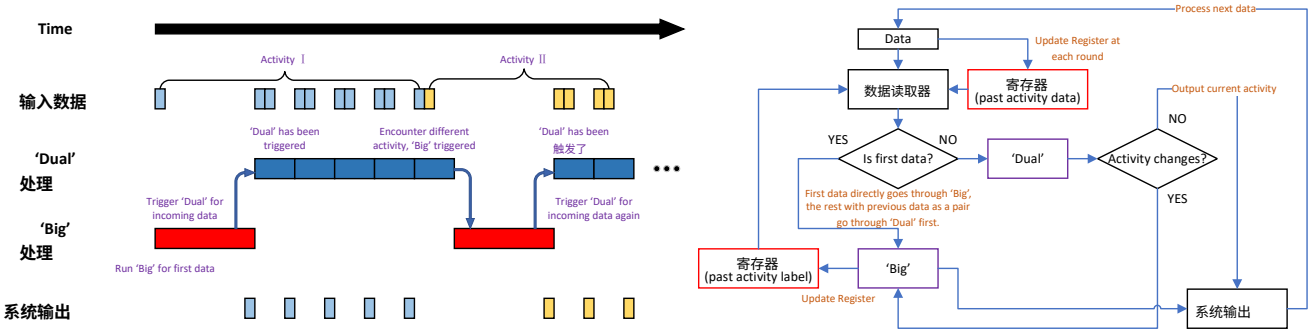


图8。 自适应神经网络系统“大” + “双”配置的处理步骤（**左**）和流程图（**右**）。在左图中，两个输入数据块表示“双路”网络所需的一对相邻数据样本。在右图中，寄存器存储“双路”网络当前处理过程中的先前数据和标签。

The ‘big’ network is the same as the one introduced in the previous configuration, while the secondary ‘dual’ network has been reconstructed as shown in Figure 9 and Table 3. In the same way as for the ‘little’ network, the single input data from the total accelerometer sensor are selected for the ‘dual’ network. Therefore, the input data shape of the ‘dual’ network becomes (1, 128, 3, 2), which contains two adjacent input data samples. As there is a significant increase in the input data shape, the number of parameters increases from 184 in the ‘little’ network to 300 in the ‘dual’ network.

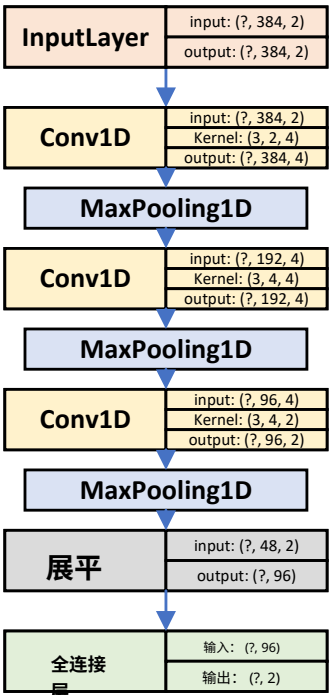


Figure 9. ‘Dual’ model structures in Keras.

Table 3. 'Dual' model parameter details. The pooling layers are hidden. For more info, see Figure 9.

Model: 'Dual'		
Layer (Type)	**Output Shape**	**Param#**
model_input	[(None, 384, 2)]	0
conv1d	(None, 384, 4)	28
conv1d_1	(None, 192, 4)	52
conv1d_2	(None, 96, 2)	26
model_output	(None, 2)	194
参数总数：300		

4.3. '大' +距离配置

Finally, we consider whether the wake-up module in the adaptive system can be replaced by a simpler algorithm instead of using neural networks such as 'little' and 'dual' networks. This configuration, which is similar to the second configuration, replaces the 'dual' network model with a distance calculator measuring the difference in the distance between two adjacent input samples. In order to pick up on an activity change, a distance calculator using Minkowski distance and Mahalanobis distance is applied to trigger the 'big' network when the difference in distance reaches a pre-set threshold value as shown in Figure 10.

$$D(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \tag{1}$$

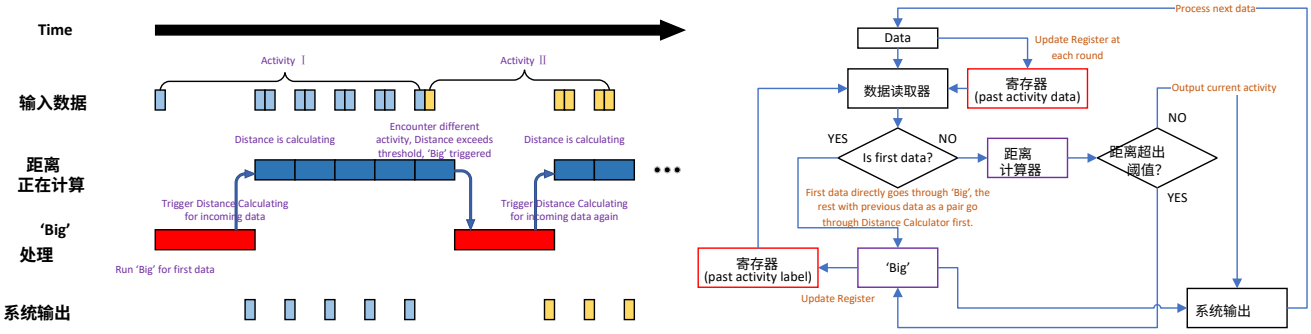


图 10. 自适应神经网络系统的"大" + 距离配置的处理步骤（**左侧**）和流程图（**右侧**）。在左图中，两个输入数据块表示距离计算器所需的一对相邻数据样本。在右图中，寄存器存储距离计算器当前处理过程中使用的先前数据和标签。

<escape></escape>欧几里得距离是一种典型的度量，用于测量 N 维空间中两点之间的实际距离。如公式（<escape></escape>1<escape></escape>）所示，闵可夫斯基距离是欧几里得距离的一种广义形式。当 <escape></*escape*>*p* = 2<escape></escape> 时，它等价于欧几里得距离；而当 <escape></*escape*>*p* = 1<escape></escape> 时，它等价于曼哈顿距离。此外，马氏距离用于衡量目标点 P 与分布 D 的均值点之间的距离。如果点 P 沿着 D 的各个主成分轴远离均值点移动，该距离就会增大。当这些轴经过缩放并具有单位方差时，马氏距离就会变为欧几里得距离</escape><escape></escape>。

" 双"网络的输入数据形状为 (1, 128, 3, 2)，应拉伸为 (1, 384, 2)，其中数值 2 表示距离计算器需要两个相邻的数据样本。然后，计算器会测量这两个数据样本之间的闵可夫斯基距离。

遵循公式（1）的两种情况下的相邻数据样本，其中 $p = 1$ 和 $p = 2$。计算前，马氏距离需要数据集的协方差矩阵。要唤醒‘big’模型，可以选择多个阈值来实现多种灵敏度。只有当先前数据样本与当前数据之间的距离超过预设阈值时，才会触发‘big’模型。因此，较低的阈值将达到更高的推理准确率，因为‘big’网络会被更频繁地调用。相反，较高的阈值意味着‘big’网络被调用的次数较少，从而缩短推理时间。

5. Neural Network Microcontroller Deployment

The neural network models in the Keras format are quantized to the UINT8 format to reduce the amount of memory needed before MCU deployment. According to Equation (2) in [37], as shown below, the real value is the input value of the training process in the range of [−128, 127], while the quantized value is the target value after the quantization, which is in the UINT8 range of [0, 255]. The mean and the standard deviation values can be calculated as 128 and 1, respectively. Finally, the model in a quantized format is obtained.

$$real_value = (quantized_value - mean_value)/std_dev_value \quad (2)$$

我们使用来自 UCI-HAR 的现有数据样本 [34] 而非实时数据，以便在不同平台之间进行公平比较。因此，当微控制器运行应用程序时，可以正确访问存储的数据和网络模型。此外，第 4 节中介绍的自适应系统模型切换算法是在 C 代码层面而非网络模型层面实现的。‘big’ 和 ‘little’模型能够独立调用，这意味着自适应系统在寻求性能与能耗之间的平衡时具有更高的灵活性和有效性。最后，在将程序烧录到目标开发板之前，必须使用适用于 GCC 的交叉编译工具 [38], ARM 编译器 [39]以及 *Eta Compute* 的 TENSAIFlow 编译器 [29]将应用程序编译为可执行二进制文件。模型部署过程如图 11 和 12所示。

5.1. STM32L4R5ZI

STM32Cube.AI 意法半导体推出的 [28]是一个旨在优化 STM32L4 等 STM 设备的框架。然而，由于其专有环境的局限性，主网络与次级网络之间的切换算法无法在 C 代码层面进行部署。另一方面 [26], 其设计侧重于在不同微控制器板上实现通用且灵活的部署。 *NNoM* 转换器能够将 Keras 格式的预训练模型转换为 C 代码，其神经网络库可用于部署该模型。因此，选择 *NNoM* 框架而非*STM32Cube.AI* ，用于在 STM32L4 上部署模型（见图 12）。

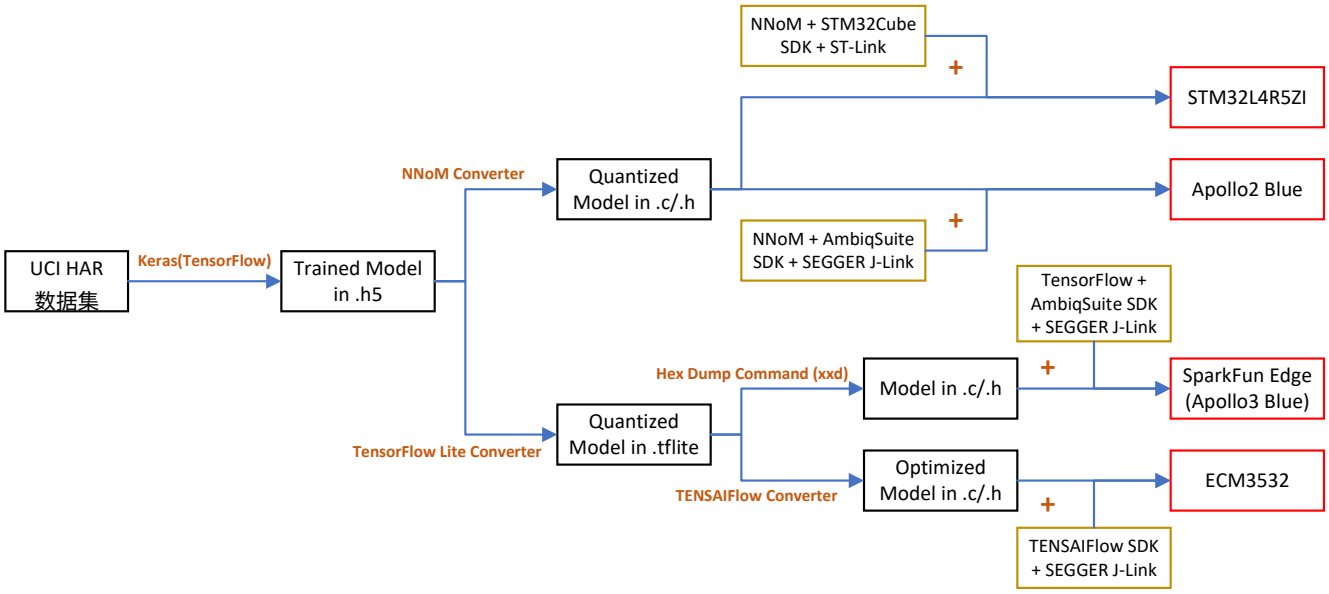


图 11。 目标微控制器板上神经网络模型部署步骤的示意图。黑色框表示训练好的模型，棕色框表示所使用的库源代码，红色框表示微控制器板。

设备	NN Model Training	Conversion of NN Model	NN Library in C	底层开发库	连接
STM32L4R5ZI	Keras(TensorFlow)	NNoM Converter	NNoM lib	STM32Cube SDK	ST-Link
Apollo2 Blue	Keras(TensorFlow)	NNoM Converter	NNoM lib	AmbiqSuite SDK	SEGGER J-Link
SparkFun Edge (Apollo3 Blue)	Keras(TensorFlow)	TensorFlow Lite Converter	TensorFlow lib	AmbiqSuite SDK	SEGGER J-Link
ECM3532	Keras(TensorFlow)	TensorFlow Lite 转换器 / TENSAIFlow 转换器	TENSAIFlow lib	TENSAIFlow SDK	SEGGER J-Link

图 12。 不同微控制器各部署阶段所使用软件的比较。

需要使用意法半导体提供的 STM32Cube 软件开发套件 1.17.0 版本，其中包含实用工具和示例项目，以驱动 STM32L4R5ZI 微控制器开发板。选择 ARM 提供的 Keil uVision IDE 来搭建支持 STM32L4 的编程环境。需要通过 Keil 的软件包安装程序安装 STM32L4 驱动程序包。使用 Micro-USB 电缆将 STM32L4 的 CN1 端口连接到台式个人电脑。随后，可在目标调试页面中选择 ST-Link 调试器，个人电脑即可连接并检测到 STM32L4 设备。另外，如果连接失败，可使用意法半导体提供的 STM32 ST-LINK Utility 擦除开发板，以避免软件冲突。

神经网络模型由 Keras（*TensorFlow*_v1.15）训练完成后，需要通过应用 *NNoM* 转换器命令进行量化，如清单 1所示。然后，可以使用以下函数生成包含模型权重的头文件。在构建项目之前，应通过 Keil 项目管理项添加权重头文件、输入数据文件以及 *NNoM* 库中的文件。最后，可以执行构建项目并将其闪存到开发板的步骤。若要在调试查看器中观察输出，目标调试设置的跟踪页面中的内核时钟应与设备的工作时钟匹配。

清单 1NNoM 转换器用于生成量化模型的 Python 函数。

```
生成_模型（模型，x_测试，名称='weight.h'）
```

5.2. Apollo2 Blue

Ambiq 提供的 AmbiqSuite SDK 2.2.0 支持在 Apollo2 Blue 上进行模型部署。ARM 提供的 Keil uVision IDE 用于设置编码环境。安装 Apollo2 的驱动包后，通过 Micro-USB 电缆将 Apollo2 开发板连接到个人电脑，并在 Keil 的目标调试页面中选择 J-Link，如图 11所示。与 STM32L4 类似，Apollo2 不支持 *TensorFlowLite* 和 *TENSAIFlow*。因此，使用 *NNoM*转换器将 Keras 格式的预训练模型转换为量化格式，如清单 1所示。然后，应通过 Keil 项目管理项将模型权重、数据头文件和 *NNoM*库添加到项目中。构建项目并将其烧录到目标开发板后，可以使用 Keil 调试查看器观察模型输出。

5.3. SparkFun Edge (Apollo3 Blue)

TensorFlow 由 Google 开发，不仅能够训练神经网络模型，还包含 *TensorFlow Lite* ，可在微控制器等边缘设备上部署网络模型 [27]。以 Keras 格式保存的训练后网络模型可使用 *TensorFlow Lite* 转换器转换为量化格式，见清单 2 和3。我们提供了 C 语言库源代码和开发板 SDK 文件，以支持在微控制器上部署模型（见图11）。我们使用*TensorFlow Lite* 支持在 SparkFun Edge（Apollo3）微控制器开发板上部署模型。

AmbiqSuite SDK 2.2.0 版本包含 Ambiq 提供的实用工具和驱动程序，以支持 SparkFun Edge（Apollo3 Blue）。*TensorFlowLite* 1.15 版本用于将使用浮点参数的 Keras 模型转换为具有 UINT8 参数的 TFLite 模型。按照清单 2 和 3中的相应命令行，即可生成量化模型文件并准备好进行部署。使用十六进制转储命令“xxd”，可将 TFLite 模型转换为 TensorFlow Lite 库可读取的十六进制文件。最后，我们用 Micro-USB 电缆将 Apollo3 连接到个人电脑，并使用 AmbiqSuite SDK 提供的烧录工具将二进制文件烧录到目标开发板。

Listing 2 TensorFlow Lite converter command lines for ‘big’ model quantization.

```
tflite_convert \
    --keras_model_file=./Output_Models/${MODELNAME}.h5 \
    --output_file=./Output_Models/${MODELNAME}.tflite \
    --inference_type=QUANTIZED_UINT8 \
    --input_shapes=1,128,3:1,128,3:1,128,3 \
    --input_arrays=model_input1,model_input2,model_input3 \
    --output_arrays=model_output/BiasAdd \
    --default_ranges_min=0 --default_ranges_max=255 \
    --mean_values=128,128,128 --std_dev_values=1,1,1 \
    --change_concat_input_ranges=false \
    --allow_nudging_weights_to_use_fast_gemm_kernel=true \
    --allow_custom_ops
```

Listing 3 TensorFlow Lite converter command lines for ‘little’ model quantization.

```
tflite_convert \
    --Keras_模型_文件=./Output_Models/$
```

```
--输出_数组=模型_输出/BiasAdd \--默认_范围_最小值
```

5.4. ECM3532

Eta Compute 的 *TENSAIFlow* 是一个框架，旨在为 Eta 产品（如 ECM3531 和 ECM 3532 [29]）部署预训练网络模型。该框架针对 Eta Compute 产品进行了高度优化，以实现性能与效率之间的最佳平衡。该框架无法训练 *TensorFlow* 等神经网络模型；它仅提供训练后的模型转换和部署功能。预训练模型先由*TensorFlow Lite* 转换为量化 *TFLite* 格式，然后 TENSAIFlow 会将 TFLite 模型转换为可与库源代码配合调用的 C 代码。ECM3532 开发板不支持 *NNoM* 或 *TensorFlow Lite*。因此，*TENSAIFlow* 软件开发工具包 2.0.2 版本包含 Eta Compute 的 *TENSAIFlow* 转换器和神经网络库，以支持在 ECM3532 上部署模型。如图 11 所示，来自 Keras 的预训练模型（*TensorFlow*_v1.15）首先使用*TensorFlow Lite*转换器量化为 UINT8 格式（清单 2 和3），然后使用*TENSAIFlow*转换器转换为*TENSAIFlow* 库可读取的格式（清单 4）。接着，我们构建项目并将其烧录到目标 ECM3532。

Listing 4 TENSAIFlow converter command lines for model conversion.

```
./tensaiflow_compile \
    --tflite_file ../model_zoo/${MODELNAME}.tflite \
    --out_path ../../../Applications/${PROJECTNAME}/src/ \
    --weights_path ../../../Applications/${PROJECTNAME}/include/
```

6. Results and Discussion

使用完整 HAR 测试数据集时，不同配置与原始配置的准确率如图 13所示。本文不考虑不同的随机初始化种子，而是对不同的微控制器使用同一个经过训练的网络，以进行公平比较。我们还谨慎选择学习率，采用相对较慢的学习率和 SGDR，以防止模型陷入局部最优点而非全局最优点。我们使用留出交叉验证划分整个数据集：70% 用于训练数据集，15% 用于验证数据集，15% 用于测试数据集。

在图 13 中，仅使用‘big’的配置具有 91.3% 的准确率，但该模型的调用次数较多，会导致显著的延迟。‘big’ + 六个‘little’的配置达到了相当的准确率水平，并且‘big’模型的调用次数从 2947 次减少到 406 次，使 ‘big’ 模型的推理时间缩短了三分之二。‘big’+ “ 双 ” 配置由于次级“双”网络的准确率较低，无法达到类似的准确率。‘big’ + 距离配置的测试准确率相对较低，并且在 2947 个数据样本中调用了‘big’网络 669 次。

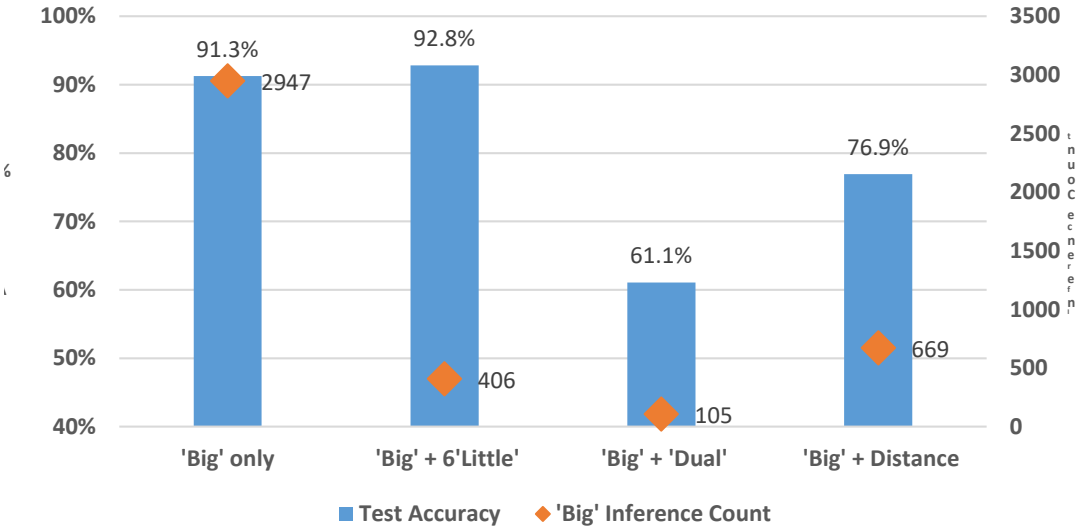


Figure 13. The accuracy and the 'big' inference counts for four configurations (quantized TFLite format) on the test data set (2947 samples) have been evaluated on a PC.

为了建立相同的测试环境，并为所有微控制器板提供相同的测试样本，我们选择使用 UCI-HAR 测试数据集中的数据样本，而不是来自板载传感器的实时信号。由于内存限制，只能从 UCI-HAR 测试数据集中选取 60 个数据样本，以便与模型和切换算法一同装入微控制器板。因此，我们为每种活动选取十个数据样本，并将它们按活动 I 至 VI 的顺序组成一个序列。这意味着测试数据序列中有五次活动变化。我们已验证，使用这 60 个样本得到的分类结果与使用完整数据集得到的结果相当；尽管由于工具链不同，各设备之间存在一些可忽略的差异。以下评估均在 48 兆赫工作频率下进行，且未启用调试。下面评估自适应神经网络的四种配置：

6.1. 'Big' Only

As shown in Figure 13, after removing the LSTM layers used in [23], we still maintain an accuracy level of around 90% for the 'big' network on the activity classification task compared to the results in [22,23]. The original 'big' model method performs 2947 inferences on all test data samples. Due to the large topology of the 'big' model and a large number of inferences, the 'big'-only configuration has the highest execution time. This can be seen in Figure 14: for all four MCUs working at the same operating frequency of 48 MHz, the latency of the 'big'-only model is the highest among all four configurations. The power consumption values for each configuration show negligible variations for each MCU in Figure 15. Therefore, the energy consumption for each configuration is only affected by the inference time. As shown in Figure 15, the 'big'-only configuration consumes the highest value of energy.

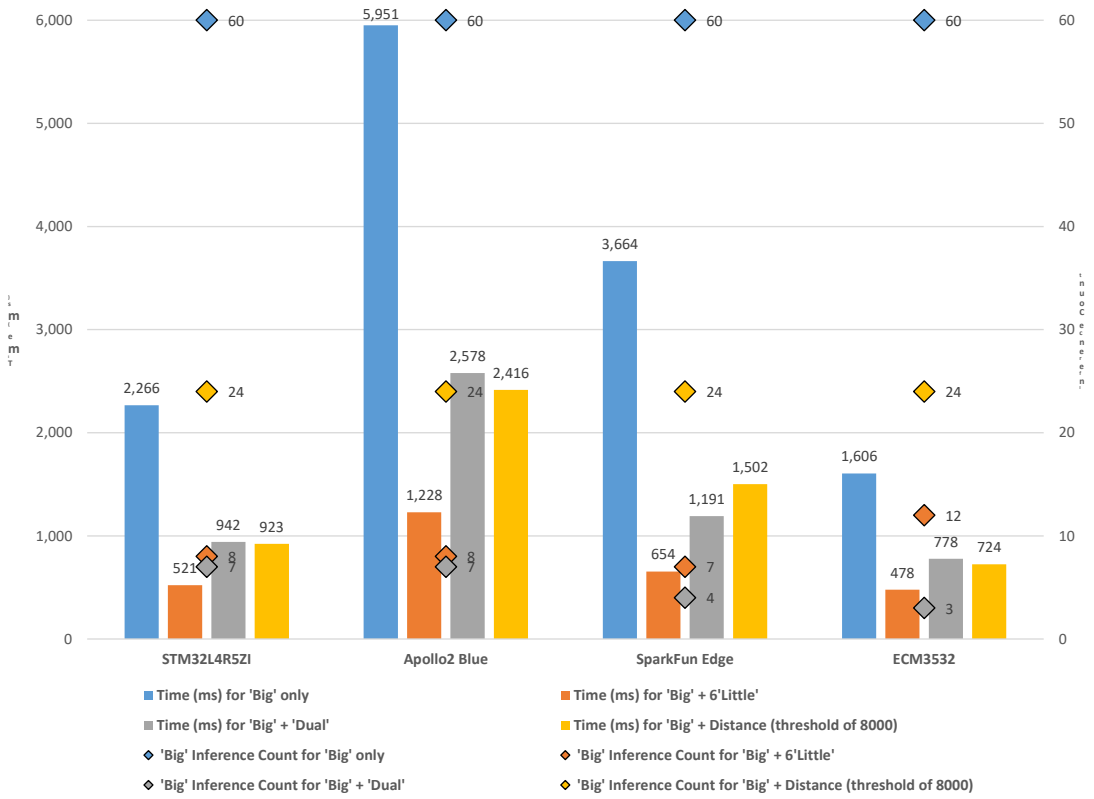


Figure 14. The time evaluation of four adaptive configurations on MCU boards with the 'big' inference counts. A total of 60 data samples extracted from the UCI-HAR test data set are tested to form the evaluation.

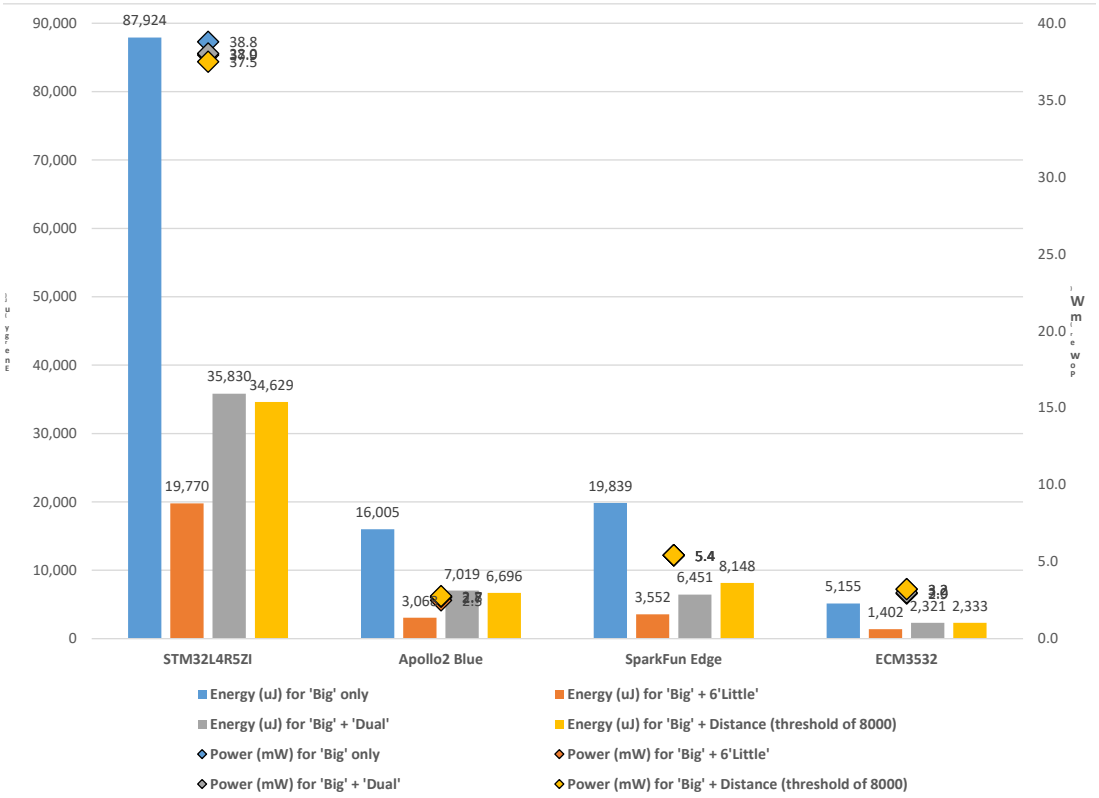


Figure 15. The power and energy evaluation of four adaptive configurations on MCU boards. A total of 60 data samples extracted from the UCI-HAR test data set are tested to form the evaluation.

6.2. 'Big' + Six 'Little'

The difference between the inference time of the 'big'-only and 'big' + six 'little' configurations is shown in Figure 14. The inference latency of the 'big' model is around 12 times longer than the latency of the 'little' model. Therefore, the lower the number of times the 'big' network gets invoked, the higher the efficiency of the system is. In this configuration, six 'little' models are applied to save time by restricting the 'big' inference count to around ten times. In all MCU evaluations in Figure 14, the time result of the 'big' + six 'little' configuration is the lowest and this reduces the execution time by around 80% compared to the original 'big'-only configuration and around 50% compared to the others. For all four configurations, the power is largely equivalent, as can be seen in Figure 15. Due to the significant advantage of the 'big' + six 'little' configuration in terms of execution time, this configuration achieves energy savings of around 80% compared to the original 'big' method on all MCUs.

6.3. 'Big' + 'Dual'

与“大” + 六个“小”配置相比，“大” + “双”配置不受需要分类的类别数量限制。先前配置中的“小”网络数量由类别数量决定，当类别数量较多时（例如 CIFAR-100 数据集），这会给模型部署带来困难。通过应用一个专注于检测活动变化的网络，“大” + “双”配置可以通过比较当前活动和先前活动来识别活动变化。然而，该配置存在两个缺陷。首先，在 7352 个训练数据样本中，只有 280 个活动切换案例。我们提取了 280 个带有“活动变化”标签的数据样本和 7072 个带有“活动持续”标签的数据样本来训练

“双”模型，导致训练数据集不平衡。其次，还存在错误传播问题，即在发生“活动变化”时，“双”分类出现错误。例如，在图 16 中，“双”模型在第七个数据样本处出现错误，此时活动从 I 切换到 III，跳过了“大”模型推理，并误导自适应系统输出活动 I。此后，“双”模型在剩余数据中没有再出现错误，也未检测到任何活动变化。由于第七个输出被设置为活动 I 而非 III，该自适应系统会持续传播输出错误。与“大” + 六个“小”模型配置相比，“大”模型同样在第七个数据样本处被跳过，因为“小”模型未能检测到任何变化（发生错误）。然而，在接下来的数据输入后，“小”模型能够识别出当前活动已不再是活动 I。随后，系统调用“大”模型，输出正确的活动标签，并恢复到正确状态。

数据编号	…	1	2	3	4	5	6	7	8	9	10	…
Real Activity Label		II	II	II	I	I	I	III	III	III	III	
'Little' Output		II	II	II	不是 II	I	I	I	不是我	III	III	
'BIG' Output		Not 已调用	Not 已调用	Not 已调用	I	Not 已调用	Not 已调用	Not 已调用	III	Not Invoked	Not Invoked	
'Big' + 6 'Little' Output		II	II	II	I	I	I	I	III	III	III	
'Dual' Output		Remains	Remains	Remains	Changes	Remains	保持不变	保持不变	保持不变	保持不变	保持不变	
'BIG' Output		Not 已调用	Not 已调用	Not 已调用	I	Not 已调用	Not 已调用	Not Invoked	Not Invoked	Not Invoked	Not Invoked	
'Big' + 'Dual' Output		II	II	II	I	I	I	I	I	I	I	

图 16. 发生“活动变化”时，两种配置的输出比较。错误以红色标出。下方展示了主模块、次模块以及整体自适应系统的结果。

尽管“双路”模型能够解决大类别问题，但由于训练数据集不平衡，其训练并不充分。由于“双路”模型的准确率较低，图 16 中所述的错误传播会发生，导致系统在检测到活动变化、需要进一步推理时，无法切换到“大”模型。因此，“大”模型的推理次数极少，但准确率方面的性能相对较差。因此，如图 13所示，该自适应系统的总体准确率（在个人电脑上对所有测试数据约为 60%）低于其他配置。此外，由于“双路”模型的复杂度相对较高，且持续处于激活状态，推理过程的复杂度也随之增加。此外，作为“双路”模型输入的前一数据样本与当前数据样本的组合需要经过预处理。因此，尽管“大”模型的推理次数最少（图 13），其延迟和能耗仍是最佳配置的两倍；该最佳配置采用“大”模型与 + 六个“小”模型，如图 14 和 15所示。

6.4. ‘Big’ + Distance

“大” +距离配置，如图 17所示，曼哈顿距离和欧几里得距离在区分活动 I 至 III（即步行、步行_上楼和步行_下楼）方面的性能较差。同一活动的数据样本之间的距离超过了不同活动的数据样本之间的距离（见图 17中的数据8至10）。因此，由于这些无法区分的值，无法设置明确的阈值边界来将“活动变化”情况与未发生变化的活动区分开来。

数据编号	1	2	3	4	5	6	7	8	9	10	11	12	
活动标签	V	V	IV	IV	VI	VI	I	I	III	III	II	II	活动保持不变
曼哈顿系距 Distance (p = 1)	NULL	154	10,230	230	19,717	4,153	28,782	8,642	7,300	8,518	9,174	5,376	Activity changes
Euclidean Distance (p = 2)		11	582	20	1,090	124	1,693	565	485	756	604	389	Distinguishable 价值观
Mahalanobis 距离		201,532	1,609,852	263,481	1,349,489	746,925	418,438	511,839	653,612	547,821	495,100	586,691	Indistinguishable 值

图 17. 不同相邻数据样本对的距离结果。同一种距离度量（同一行）中的数值可以进行比较。

在“big” + 距离配置中，评估图 13 和 14时，曼哈顿距离的阈值点选为 8000。该阈值 8000 会更频繁地触发“big”模型，因此可以认为它较为敏感。与“big” + “dual”模型一样，“ big” + 距离模型也存在错误传播问题，严重影响总体准确率。与“big” +六个“little”配置相比，该配置的准确率相对较低，约为 76%；如图 13所示，其“big”模型调用次数更多。此外，该配置的延迟和能量成本显著增加，是“big” + 六个“little”模型的两倍；如图 + 所示，它与“big”“ dual”配置相似，见图 14 和 15。

总体而言，在所有微控制器中，我们最佳的自适应网络配置，即‘big’网络 + 六个‘little’配置，实现了约 90% 的高预测准确率，与原始的纯‘big’方法相当。如第 3 节对微控制器的初步评估所述，ECM3532 的处理速度最快，其次是 STM32L4、SparkFun Edge (Apollo3) 和 Apollo2（按速度从快到慢排列）。图 14 中相同配置下的执行时间表明，四种微控制器板均呈现一致的结果。对于‘big’网络 + 六个‘little’配置，与原始方法相比，‘big’推理次数减少了约 85%，微控制器的速度最高可提升 5× 。由于微控制器板在运行不同配置时均处于工作模式，这些配置的功耗与图 15 所示的微控制器功耗相近。由于不同网络配置的功耗差异可忽略不计，各微控制器配置的能量消耗分布与图 14 中的时间开销分布一致。如图 15 所示，在所有设备中，‘big’网络 + 六个‘little’算法配置相比原始的纯‘big’方法可节能约 80%，相比另外两种配置可节能约 50%。此外，与仅运行‘big’网络的标准MCU相比，最佳配置——‘big’网络 + 六个‘little’模型，结合最佳的近阈值先进硬件，可将能耗降低多达 98%，从而使电池供电活动检测应用的运行寿命 62× 延长。

7. Conclusions

本研究比较了市售近阈值微控制器与标准MCU在性能和能耗方面的表现。在相同工作频率下，近阈值微控制器在功耗方面具有显著优势，比标准MCU低80%。由于处理速度相近，低功耗近阈值MCU与标准MCU STM32L4相比，可节能约80%。此外，我们提出了三种自适应神经网络配置，并研究了如何通过在微控制器上的部署应用我们的算法，在保持预测准确率的同时降低能耗。我们证明，尽管这些设备可用的存储器容量有限，仍然能够部署“大—小”配置，从而实现

显著更优的能耗与性能表现。所提出的算法可成功部署于 STM32L4R5ZI、Apollo2 Blue、SparkFun Edge (Apollo3 Blue) 和 ECM3532。UCI-HAR应用体现了活动识别任务的特点：假设某项活动会持续一段时间，然后才切换到另一项活动。为节省时间和能量，当活动保持不变时，我们会启用推理速度更快的次级模型，使主模型暂停运行。在UCI-HAR测试中，表现最佳的自适应网络配置——“大” + 六个“小”配置——能耗降低了80%，且预测准确率与原始方法相当。结果证明，在我们测试的所有设备上，所提出的方法都能实现不同程度的时间与能耗降低，同时保持准确率。此外，结合近阈值微控制器后，最佳配置在UCI-HAR上的电池寿命最高可提升62倍，相较于使用标准MCU的原始非自适应方法。

未来的工作包括将这项研究扩展到其他应用领域，例如机器健康监测和异常检测。此外，我们计划研究如何通过采用额外的网络层级，将该方法扩展到可能输出类别数量众多的应用，同时避免存储器需求激增。最后，未来的研究方向之一是开发一个框架，能够根据总体准确率和能量，从“大”配置中自动提取最优的“小”配置，以取代人工分析。

作者贡献： 方法论，Z.S.、N.H. 和 J.N.-Y.；软件，Z.S. 和 J.N.-Y.；验证，Z.S.；资源，Z.S.；数据整理，Z.S.；撰写——初稿准备，Z.S.；撰写——审阅与编辑，Z.S.、N.H. 和 J.N.-Y.；可视化，Z.S.；监督，J.N.-Y.；所有作者均已阅读并同意稿件的已发表版本。

Funding: This work was partially funded by the Royal Society INF/R2/192044 Machine Intelligence at the Network Edge (MINET) fellowship.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Our work can be found here: (accessed on 9 March 2022) https://github.com/DarkSZChao/Big-Little_NN_Strategies.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MCU	Microcontroller Unit
LoT	Internet of Things
CNN	Convolutional Neural Network
UCI-HAR	UCI-Human Activity Recognition

参考文献

1. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. 边缘智能：利用边缘计算打通人工智能的最后一公里。*IEEE* 会议论文集 **2019**，*107*，1738–1762。2. Chen, J.; Ran, X. 边缘计算中的深度学习：综述。*IEEE* 会议论文集 **2019**，*107*，1655–1674。3. Coral。Edge TPU。在线获取：https://coral.ai/docs/edgetpu/faq/ （访问日期：2022年2月20日）。4. Ambiq Micro。Apollo3 Blue 数据手册。在线获取：https://cdn.sparkfun.com/assets/learn_tutorials/9/0/9/Apollo3_Blue_微控制器_数据_手册_版本0_9_1.pdf （访问日期：2021年12月15日）。5. Eta Compute。Eta Compute ECM3532 AI 传感器产品简介。在线获取：https://media.digikey.com/pdf/Data%20Sheets/Eta%20Compute%20PDFs/ECM3532-AI-Vision-Product-Brief-1.0.pdf （访问日期：2021年12月15日）。

6. Chaudhary, H. Eta Compute 的 ECM3532开发板可通过单节电池提供数月的 AI视觉。在线获取：https://opencloudware.com/eta-computes-ecm3532-board-provides-ai-vision-that-can-work-for-months-on-a-single-battery/（访问日期：2022年2月20日）。 7. Flamand, E.; Rossi, D.; Conti, F.; Loi, I.; Pullini, A.; Rotenberg, F.; Benini, L. GAP-8：面向物联网边缘人工智能的 RISC-V 片上系统。载于：2018年IEEE第29届专用系统、架构与处理器国际会议（ASAP）论文集，意大利米兰，2018年7月10–12日；第1–4页。 8. Clarke, P. CEO专访：Minima 的 Tuomas Hollman 谈静态时序签核为何已成过去。在线获取：https://www.eenewseurope.com/en/ceo-interview-minimas-tuomas-hollman-on-why-static-timing-sign-off-is-over/（访问日期：2022年2月20日）。 9. Clarke, P. Minima 与 ARM 将“实时”电压调节应用于 Cortex-M3。在线获取：https://www.eenewsanalog.com/news/minima-arm-apply-real-time-voltage-scaling-cortex-m3（访问日期：2022年2月20日）。 10. Flaherty, N. €100m项目将开发低功耗边缘人工智能微控制器。在线获取：https://www.eenewseurope.com/en/e100m-project-to-develop-low-power-edge-ai-microcontroller/（访问日期：2022年2月20日）。 11. Novac, P.E.; Hacene, G.B.; Pegatoquet, A.; Miramond, B.; Gripon, V. 微控制器上深度神经网络的量化与部署。《传感器》2021，21，2984。 12. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. 面向高效纯整数运算推理的神经网络量化与训练。载于：IEEE计算机视觉与模式识别会议论文集，美国犹他州盐湖城，2018年6月18–23日；第2704–2713页。 13. Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; Bengio, Y. 量化神经网络：以低精度权重和激活值训练神经网络。《机器学习研究杂志》2017，18，6869–6898。 14. Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; Bengio, Y. 二值化神经网络：将深度神经网络的权重和激活值限制为 +1 or −1进行训练。arXiv 2016，arXiv:1602.02830。 15. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. Xnor-net：使用二值卷积神经网络进行 ImageNet 分类。载于《计算机视觉——ECCV 2016》，欧洲计算机视觉会议论文集，荷兰阿姆斯特丹，2016年10月11–14日；Springer：瑞士 Cham，2016；第525–542页。 16. Mocerino, L.; Calimera, A. CoopNet：面向低功耗微控制器的协作式卷积神经网络。载于：2019年IEEE第26届电子、电路与系统国际会议（ICECS）论文集，意大利热那亚，2019年11月27–29日；第414–417页。 17. Amiri, S.; Hosseinabady, M.; McIntosh-Smith, S.; Nunez-Yanez, J. 异构硬件上的多精度卷积神经网络。载于：2018年欧洲设计、自动化与测试会议暨展览（DATE）论文集，德国德累斯顿，2018年3月19–23日；第419–424页。 18. Romaszkan, W.; Li, T.; Gupta, P. 3PXNet：面向边缘机器学习的剪枝置换打包XNOR网络。《ACM嵌入式计算系统汇刊》2020，19，5。 19. Yu, J.; Lukefahr, A.; Das, R.; Mahlke, S. TF-Net：在微控制器上部署亚字节深度神经网络。《ACM嵌入式计算系统汇刊》2019，18，45。 20. Teerapittayanon, S.; McDanel, B.; Kung, H.T. BranchyNet：通过深度神经网络提前退出实现快速推理。载于：2016年第23届国际模式识别会议（ICPR）论文集，墨西哥坎昆，2016年12月4–8日；第2464–2469页。 21. Park, E.; Kim, D.; Kim, S.; Kim, Y.D.; Kim, G.; Yoon, S.; Yoo, S. 面向超低功耗推理的大小核深度神经网络。载于：2015年硬件/软件协同设计与系统综合国际会议（CODES+ ISSS）论文集，荷兰阿姆斯特丹，2015年10月4–9日；第124–132页。 22. Nunez-Yanez, J.; Howard, N. 采用近阈值处理器和硬件加速器的高能效神经网络。《系统架构杂志》2021，116，102062。 23. Ordóñez, F.J.; Roggen, D. 用于多模态可穿戴活动识别的深度卷积神经网络和LSTM循环神经网络。《传感器》2016，16，115。 24. Turaga, P.; Chellappa, R.; Subrahmanian, V.S.; Udrea, O. 人类活动的机器识别：综述。《IEEE电路与系统视频技术汇刊》2008，18，1473–1488。 25. Wang, X.; Magno, M.; Cavigelli, L.; Benini, L. FANN-on-MCU：用于物联网边缘高能效神经网络推理的开源工具包。《IEEE物联网杂志》2020，7，4403–4417。 26. Ma, J.; parai.; Mabrouk, H.; BaptisteNguyen; idog ceva; Xu, J.; LÊ, M.T. Majianjia/nnom，版本0.4.3；Zendo：瑞士日内瓦，2021。https://doi.org/10.5281/zenodo.5737372。 27. TensorFlow。TensorFlow Lite 指南。在线获取：https://www.tensorflow.org/lite/guide（访问日期：2021年5月10日）。 28. 意法半导体。STM32人工智能生态系统。在线获取：https://www.st.com/content/st_com/en/ ecosystems/artificial-intelligence-ecosystem-stm32.html（访问日期：2021年6月8日）。 29. Eta Compute。TENSAI®Flow。在线获取：https://etacompute.com/tensai-flow/（访问日期：2021年5月10日）。 30. 意法半导体。STM32L4R5xx数据手册。在线获取：https://www.st.com/resource/en/datasheet/stm32l4r5zg.pdf（访问日期：2021年6月8日）。

31. Ambiq Micro. Apollo2 MCU Datasheet. Available online: https://ambiq.com/wp-content/uploads/2020/10/Apollo2-MCU-Datasheet.pdf (accessed on 19 July 2021).
32. Yeo, K.S.; Roy, K. *Low Voltage, Low Power VLSI Subsystems*; McGraw-Hill, Inc.: New York, NY, USA, 2004; p. 44.
33. Nunez-Yanez, J. Energy proportional neural network inference with adaptive voltage and frequency scaling. *IEEE Trans. Comput.* **2018**, *68*, 676–687.
34. UCI Machine Learning. Human Activity Recognition Using Smartphones Data Set. Available online: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones (accessed on 2 March 2021).
35. De Maesschalck, R.; Jouan-Rimbaud, D.; Massart, D.L. The mahalanobis distance. *Chemom. Intell. Lab. Syst.* **2000**, *50*, 1–18.
36. McLachlan, G.J. Mahalanobis distance. *Resonance* **1999**, *4*, 20–26.
37. TensorFlow. Converter Command Line Reference. Available online: https://github.com/tensorflow/tensorflow/blob/master/tensorflow/lite/g3doc/r1/convert/cmdline_reference.md (accessed on 29 March 2021).
38. GCC team. GCC, the GNU Compiler Collection. Available online: https://gcc.gnu.org/ (accessed on 25 May 2021).
39. Arm Developer. GNU Arm Embedded Toolchain. Available online: https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm (accessed on 25 May 2021).