Ultra-Efficient On-Device Object Detection on AI-Integrated Smart Glasses with TinyissimoYOLO

Julian Moosmann[1](✉), Pietro Bonazzi[1], Yawei Li[1], Sizhen Bian[1], Philipp Mayer[1], Luca Benini[1,2], and Michele Magno[1]

[1] ETH Zürich, 8092 Zürich, Switzerland
julian.moosmann@pbl.ee.ethz.ch
[2] University of Bologna, 40126 Bologna, Italy

Abstract. Smart glasses are rapidly gaining advanced functions thanks to cutting-edge computing technologies, especially accelerated hardware architectures, and tiny Artificial IntelligenceAI algorithms. However, integrating AI into smart glasses featuring a small form factor and limited battery capacity remains challenging for a satisfactory user experience. To this end, this paper proposes the design of a smart glasses platform for always-on on-device object detection with an all-day battery lifetime. The proposed platform is based on GAP9, a novel multi-core *RISC-V* processor from *Greenwaves Technologies*. Additionally, a family of sub-million parameter TinyissimoYOLO networks are proposed. They are benchmarked on established datasets, capable of differentiating up to 80 classes on MS-COCO. Evaluations on the smart glasses prototype demonstrate TinyissimoYOLO's inference latency of only 17 ms and consuming 1.59 mJ energy per inference. An end-to-end latency of 56 ms is achieved which is equivalent to 18 frames per seconds (FPS) with a total power consumption of 62.9 mW. This ensures continuous system runtime of up to 9.3 h on a 154 mAh battery. These results outperform MCUNet (TinyNAS+TinyEngine), which runs a simpler task (image classification) at just 7.3 FPS, while the 18 FPS achieved in this paper even include image-capturing, network inference, and detection post-processing. The algorithm's code is released open with this paper and can be found here: github.com/ETH-PBL/TinyissimoYOLO.

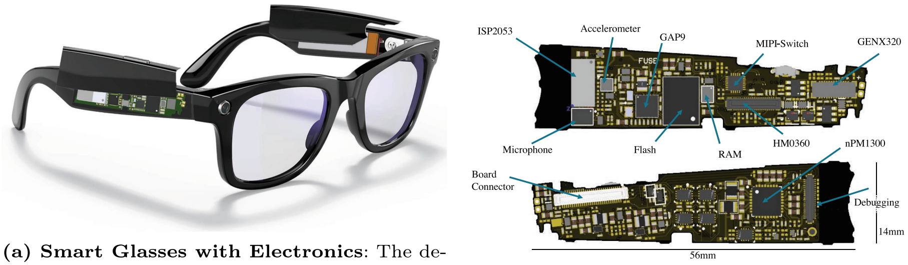
Keywords: AIoT · edge processing · image processing · neural networks · object detection · smart glasses · system design · TinyML · YOLO

1 Introduction

The rapid integration of advanced perception techniques into cutting-edge wearable computing devices has ushered in a transformative era, redefining how we engage with our surroundings and environment [2,33]. Among innovative wearables, smart glasses stand out as the next big thing in wearable computing [45].

A. Del Bue et al. (Eds.): ECCV 2024 Workshops, LNCS 15645, pp. 262–280, 2025.
https://doi.org/10.1007/978-3-031-91989-3_17

Their multifaceted applications span across a diverse spectrum, offering valuable support for professional applications [43], while at the same time enhancing user experiences in entertainment and education [15], and most importantly, improving the quality of life for individuals with disabilities [1,25,29].



(a) Smart Glasses with Electronics: The designed smart glasses hardware. The left temple holds the miniaturized electronics. The right temple contains the battery, with an energy content of up to 154 mAh.

(b) Miniaturized Smart Glasses Electronics: The proposed smart glasses platform based on GAP9 from *Greenwaves Technologies*.

Fig. 1. The designed smart glasses hardware, which retrofits commercial temples of smart glasses.

The mass adoption of current Artificial Intelligenceai technology is accentuating the trend in edge intelligence [10,50], targetting computer vision [4,10], biomedical applications [30,49], natural language processing [53], and many other spaces [2]. Similar to environmental mapping [28,39] and navigation [7], the detection and localization of objects [14,22] plays a pivotal role for understanding the visual context in which smart devices such as Artificial Intelligence of ThingsAIoT devices are operating.

Embedded machine learning for semantic scenery understanding [25] in smart glasses enhances user interaction and brings intelligence right to the user's forehead while protecting user data, minimizing latency and energy. In particular, running object detection algorithms is a crucial functionality to enable smart decisions and information about the user's surroundings [25,34].

Commercially available smart glasses such as the newly released *RayBan-Meta*, *Vuzix*® smart glasses family, are powered by *Qualcomm*'s *Snapdragon* AR1 and XR1 platforms with a power budget of a few hundred milliwatts. Despite the choice of using a relatively high-performance and powerful System-on-ChipSoC, this capable processor is mostly used for transmitting the vast amount of data taken with high-resolution cameras and microphones. However, image or speech processing is only very partially done on-device[1]. This makes the 'smart' glasses primarily an input device for an AI system working on the connected mobile phone or in the cloud [25].

[1] https://about.fb.com/news/2023/09/introducing-ai-powered-assistants-characters-and-creative-tools/.

To bridge the gap between the available computational resources and the demands of AI algorithms, novel energy-efficient and ultra-low-power Microcontroller Unit MCUs with cluster cores for parallel processing, as well as dedicated accelerated hardware are being designed and are now available on the market [6,16,38]. At the same time, new lightweight, and quantized networks are being proposed to achieve near state-of-the-art (SOTA) accuracy while having small-sized memory footprints [4,32] to fit the requirements of MCUs and accelerators. Combining these novel multi-core MCUs with a dedicated system design for ultra-low-power consumption enables a hardware platform for continuous AI inference in smart glasses, with privacy protection and all-day long battery lifetime.

This paper presents the design and implementation of such an energy-efficient intelligent smart glasses system—see Fig 1a—equipped with GAP9, an AI-capable MCU from *Greenwaves Technologies*, which consumes power in the milliwatts range. We present the first end-to-end image processing pipeline on a MCU which can process up to 18 frames per seconds (FPS) from image capturing to object localization on a MCU device. Additionally, we propose an open family of quantized and lightweight TinyissimoYOLO [31,32] networks with less than 1 MB memory. All the contributions can be summarized as follows:

1. **Smart Glasses' System Architecture**: We propose a system architecture tailored for smart glasses applications with integrated ML acceleration.
2. **Sub-Million Parameter YOLO Architectures**: We extend the TinyissimoYOLO series to support the latest datasets with up to 80 classes while having sub-million parameters when detecting more than 20 classes.
3. **End-to-end Real-Time Deployment**: We demonstrate the practical applicability of our proposed system by deploying the TinyissimoYOLO architectures on the smart glasses platform and by predicting images from the real world.
4. **Power Efficiency and Performance Evaluation**: We conduct extensive experiments to validate the power efficiency and performance of our smart glasses system. Comparative analyses against commercial edge vision systems reveal superior energy efficiency and longer battery life for our system.
5. **Open-Source Implementation**: To facilitate reproducibility and encourage further research, we release the source code of our optimized TinyissimoYOLO architecture versions, allowing the community to build upon our work and extend it to new application domains.

The paper is structured as follows: in Sect. 2, we investigate recent related works regarding both the SOTA object detection neural networks and their deployability onto MCU using different frameworks. Then, the extension of the new TinyissimoYOLO versions are described in Sect. 3. Section 4 introduces the prototype design of the smart glasses aiming for latency and energy efficiency during onboard AI execution. The results are presented in Sect. 5, including the detection results and the system evaluation of GAP9 integrated with the proposed smart glasses system. Finally, we conclude our work in Sect. 6.

2 Related Work

The following section provides an overview of SOTA object detection algorithms more specifically for use with edge processors. Additionally, we discuss different deployment frameworks to deploy networks on edge devices. Lastly, the currently available smart glasses on the market, as well as research projects, are summarized and set into perspective while this work is further motivated.

2.1 SOTA Object Detection on Microcontrollers

You Only Look Once (YOLO) [35], is an optimized deep learning algorithm used to perform real-time object detection on GPU-class devices [17]. It utilizes a feature extraction Convolutional Neural Networks (CNN) backbone and detection head to perform localization of the extracted semantic information. Meanwhile, there exist a family of different YOLO versions which differ in backbone, head structure, network size, inter-network connection, and used layer operations [3,18,19,21,36,37,46,47]. However, even the smallest *"nano"* versions of the YOLO-family exhibit roughly 3 million parameters, making them unsuitable for MCUs. As such, TinyissimoYOLO [31] and its successor [32] try to bridge the gap between accuracy and network size [40] while maximizing the available compute acceleration on the milliwatt edge device. Until now, TinyissimoYOLO was not able to detect 20 or more classes while fitting seamlessly on a MCU. The earlier version's output layer scaled linearly with the number of detection classes, quickly reaching 2 million parameters and more. Therefore, this work increases the number of detection classes while maintaining the network size below 1 MB.

Networks such as YOLOX-Nano [11], PP-PicoDet [51] and NanoDet-M[2] achieve higher mean-average precision while utilizing specialized network layers to decrease network parameters. While PP-PicoDet and NanoDet-M incorporate depth-wise convolutional layers and enhanced ShuffleNet [52] blocks, they fail to fully exploit the hardware acceleration built into MUCs such as GAP9. Conversely, YOLOX-Nano encounters memory constraints of MCUs due to its large input resolution of 640 pixels, which RGB image costs more than 1 MB of memory solely for the input image. Consequently, simple convolutional layers with a kernel of 3×3 are preferable for current accelerators built into MCUs class devices [6], which makes TinyissimoYOLO favorable for on-device execution. Nonetheless, deploying a battery-powered device with a camera resolution under 300 pixels to distinguish 80 classes of MS-COCO [26] is impractical. Therefore, we focus our evaluation of the networks on PascalVOC [9] and evaluate a few on MS-COCO for a fair comparison against similar-sized networks. Lastly, we compare them to MCUNet, which—similar to us—reports an end-to-end system latency deployed on MCU devices. For a comprehensive comparison, refer to Table 1. To the best of the author's knowledge, the only work that implemented an end-to-end object detection pipeline on a microcontroller is MCUNet [24]. They claim to have 10 FPS inference execution. When considering the

[2] https://github.com/RangiLyu/nanodet.

additional time needed for capturing an image, MCUNet's end-to-end latency achieves 7.3 FPS.

To achieve lower end-to-end latency this work builds on the results published in the TinyissimoYOLO papers [31,32]. They compared several ARM-Cortex M4 and M7 MCUs from STMicroelectronics, Apollo4b from *Ambiq* (an ultra-low-power MCU using sub-threshold technology), and the MAX78000 MCUs from *Analog Devices*. The work exploits a SOTA hardware accelerator for CNN networks, and a *RISC-V* low-power parallel processor with hardware accelerator, GAP9 from *Greenwaves Technologies*. It showed that MAX78000 is parallelizing the compute workload best. Nevertheless, GAP9 achieves the same energy efficiency despite having two-fold less MAC/cycle. Additionally, the papers show that the increased clocking frequency of GAP9 together with its flexible combination of cluster cores and neural engine, allows for larger, less restricted networks while being able to parallelize the overall workload better.

Therefore, this paper leverages the GAP9's exceptional energy-efficient parallel processing capabilities and integrates it with a low-power *Bluetooth* Low Energy BLE transceiver chip. This combination forms the backbone of our design to facilitate robust object detection in smart glasses with seamless connectivity.

2.2 Smart Glasses

Smart glasses—or in-general Augmented Reality AR—focus on a general computing concept [27] to process the user-device interaction while communicating with cloud or smartphones via Bluetooth and Wi-Fi [20]. Big tech companies such as *Google*[3] or *Apple*[4], further rely on community APPs being developed and processed on the device, requiring the manufacturer to provide an easy-to-develop software stack, to abstract the hardware from the software. This requires an Operating System OS environment to run at reasonably fast speeds, such that user interactions are ensured to run smoothly. In particular, these requirements hinder smart on-device sensor data processing, resulting in the data being processed in the cloud, while data and user privacy are not absolutely contained and guaranteed.

In contrast, several research smart glasses projects aim at the counterpart by running AI algorithms on the device [13,45]. Others investigate the human-machine interaction with smart glasses [41]. However, many publications focus on a single smart glasses application scenario and design the system accordingly, e.g., for visually impaired people [1,29] or smart gadget aid for medical [5,44] or construction work [8]. Nonetheless, to the best of our knowledge, none of the proposed research prototypes are integrated into the thin frames of the passive glasses [42] and will therefore not be non-stigmatizing nor fashionable, in contrast to actual smart glasses products such as the *RayBan-Stories*[5] or the brand-new *RayBan-Meta*[6]. Therefore, this work introduces electronics that retrofit a

[3] https://developers.google.com/glass-enterprise/.

[4] https://developer.apple.com/visionos/.

[5] https://tech.facebook.com/2023/2/the-making-of-ray-ban-stories/.

[6] https://about.fb.com/news/2023/09/new-ray-ban-meta-smart-glasses/.

Table 1. Network Comparison: Overview of similar-sized networks compared to our TinyissimoYOLO versions and evaluated on PascalVOC and MS-COCO.

Model	Image Resolution	Parameters (M)	PascalVOC mAP@(50–95)	MS-COCO mAP@(50–95)
MbV2+CMSIS [24]	128	0.87	32%	
MCUNet [24]	224	1.2	51%	
MCUNetV2-M4 [23]	224	1.01	65%	
MCUNetV2-H7 [23]	224	2.03	68%	
NanoDet-M[a]	320	0.95		21%
YOLOX-Nano [11]	640	0.91		26%
PP-PicoDet [51]	320	0.99		27%
TY-v1 3cls [31]	224	1.66	68%	
TY-v1 10cls [32]	224	2.36	65%	
TY-v1 20cls [32]	224	3.35	60%	
TY-v1.3-Small*	256	0.40	30%	
TY-v1.3-Big*	256	0.96	38%	
YOLO-v5-nano [18]	640	2.66		34%
TY-v5-Small*	256	0.63	35%	
TY-v5-Big*	256	0.89	42%	14%
YOLO-v8-nano [19]	640	3.2		37%
TY-v8-Small*	256	0.71	39%	
TY-v8-Big*	256	0.84	44%	15%
TY-v10*	256	0.85	49%	14%

* This works' TinyissimoYOLO networks deployed and evaluated on GAP9.

[a] https://github.com/RangiLyu/nanodet.

functional and fashionable smart glasses frame with a peak power consumption below 100 mW. This makes our solution to smart glasses not only aesthetically appealing but also capable of directly executing demanding YOLO object detection tasks, showcasing both efficiency and effectiveness in image processing for real-world applications.

3 TinyissimoYOLOs

This section presents a family of sub-million parameter detection algorithms, based on a different version of the YOLO architecture, which we have developed for accelerated MCUs or in general low-power edge processors. These networks are proposed as a trade-off between computational resources and performance, as for example in on-device execution for smart glasses systems. The networks predict multi-object-class probabilities and bounding boxes from 256 × 256 resolution images. However, the networks can be adapted for larger and smaller resolutions.

3.1 Network Architectures

YOLOv5, YOLOv8, and YOLOv10 employ different backbone and head architectures to predict class probabilities and bounding boxes. We accurately evaluated the respective versions to extract their performance under sub-million parameter constraints, see the network specifications in Table 1. This has been conducted to establish a family of networks suitable to be deployed on AIoT devices. In particular, for the deployment of multi-object detection networks on a MCU with ML acceleration, such as the GAP9. All the networks described below have been deployed on the GAP9 and not only will their detection capability be evaluated, but also their deployed inference performance, energy consumption, and their ability to parallelize the inference execution on such a hardware.

The new TinyissimoYOLOv1.3 key differentiator lies in the incorporation of the Detection Block from the V8 architecture into the final prediction layer of TinyissimoYOLOv1.3. In contrast to the originally proposed TinyissimoYOLO versions [31,32] this strategic enhancement imbues the latter with heightened sensitivity and a larger number of detection parameters, without scaling the detection block linearly. Consequently, TinyissimoYOLOv1.3 demonstrates precise multi-class object detection capability on sub-million parameters.

TinyissimoYOLOv5's architecture features a Cross-Stage Partial CSP connection [48] bottleneck module named 'C3'. In C3, the input is duplicated through two separate 1 × 1 convolutions which are then concatenated and processed through a final 1 × 1 convolution to produce the final output. The main distinction between the architecture of YOLOv5 and YOLOv8 lies in the number of convolutions and on the expansion or contraction of the hidden channels in the CSP. Differently from YOLOv5, TinyissimoYOLOv5-big has a layer channel multiple of 0.15 instead of 0.25 (YoloV5-nano) and contains 4 times fewer parameters, while the small version uses a channel multiple of 0.1.

YOLOv8 CSP Block called 'C2F', which powers TinyissimoYOLOv8 starts with a 1 × 1 convolution that expands the input channels to twice the hidden channel size. Then, it splits the output into two equal parts and applies a series of bottleneck layers on each part. Finally, it concatenates the processed outputs and applies a second 1 × 1 convolution. TinyissimoYOLOv8 has a depth multiple of 0.30 instead of the 0.33 in YoloV8-nano, a layer channel multiple of 0.18 instead of 0.25, and contains 5 times fewer parameters than the nano version, while the small version uses a channel multiple of 0.1.

The latest YOLO version v10 [46], removes the need to perform Non-maximum Suppression NMS during training and improves the detection accuracy on established datasets. The enhanced CSP feature extraction backbone, combined with the updated and NMS-free head achieves SOTA detection performance while having faster execution times. Our TinyissimoYOLOv10, utilizes the same backbone, neck, and head however, we use a channel multiple of 0.18 and a depth multiple of 0.15 to achieve sub-million number of parameters.

3.2 Implementation Details

We trained the proposed models on an *NVIDIA GeForce RTX* 4090 for 1000 epochs, with a batch size of 64. The initial learning rate $lr = 1e-3$ was reduced using a cosine learning rate scheduler to $lr = 1e-5$ after a 3 epoch-long warmup phase ($lr = 1e-2$). The training process utilised a Multi-Object Detection Loss [35], and images with a resolution of 256 × 256 pixels. Additionally, several established image augmentations are applied to the images, such as exposure and saturation adjustments in the HSV color space, horizontal image flipping, image translation, scaling as well as image mosaicing. We optimize the weights using Stochastic Gradient Descent SGD (momentum = 0.937) and trained with Automatic Mixed PrecisionAMP. The networks have been integrated into the *Ultralytics* framework [19]. Thereafter, we applied Post-Training Quantization PTQ [12] to quantize the networks to 8-bit integer values, such that the sub-million parameter network fits into a MCU memory of 1 MB.

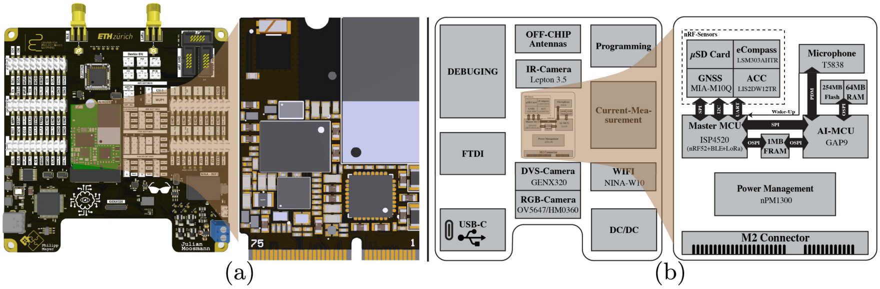


Fig. 2. Development Board: a) The proposed hardware system consists of two boards. The board on the left—development board—is shown, featuring additional power circuitry, multiple camera interfaces, Wi-Fi, and debug possibility. The smart glasses board—zoomed in—, features two MCUs, several sensors, and a power management system for stand-alone operation. b) shows the hardware block diagram for the development and smart glasses board respectively.

4 System Design

To achieve *"truly"* smart and energy-efficient smart glasses, we propose a hardware-software smart glasses solution, eventually eliminating the transmission of private data, and decreasing inference latency while effectively increasing battery run-time. For this, a development board was designed—see Fig. 2a—and utilized for rapid end-to-end evaluations while the new smart glasses PCB has been designed to retrofit existing smart glasses temples for the final system, see Fig. 1a.

4.1 Smart Glasses Design

A modular design was targeted so that reuse of the same platform can be guaranteed for other ultra-low-power all-day battery runtime AIoT applications. All components are tightly integrated onto a custom, miniaturized Printed Circuit Board PCB, to replace existing PCB of commercial smart glasses.

The design features dedicated low-power infrastructure to power cycle energy-hungry sensors while powering the minimal required components. This platform is co-designed with a development platform for the rapid integration of further research sensors.

Figure 2a shows the hardware overview. The development board is shown on the left, while the smart glasses electronics is the zoomed-in PCB (green) on the development board. The right side of the Fig. 2b describes the hardware using a block diagram. The left side shows the development board, while the zoomed-in version is the smart glasses electronics. The M2 connector aims for development and reusability. To further minimize the form factor and fit temples of smart glasses, several on-board sensors are left away, resulting in the design that can be seen in Fig. 1b. The final miniaturized smart glasses PCB fits smoothly inside the temples of smart glasses, see Fig. 1a.

Smart Glasses Electronics: The platform can be broken down into the following three main parts: (i) battery and power management,(ii) dual-MCU architecture leveraging energy-efficiency (iii) various sensors such as accelerometer, multiple cameras, and microphones.

The battery and power management is needed for untethered operation of the smart glasses' electronics. One ultra-low-power MCU is in charge of wireless communication, low-power signal processing (i.e. data from MEMS sensors), and power management of the various power domains by using multiple power switches in series and parallel within the same voltage levels. The aforementioned tasks are controlled by a SoC. In the development board, the ISM4520 has been chosen. It features a low-power nRF52 from *Nordic Semiconductor* with a built-in *ARM* Cortex-M4 processor, on-chip BLE and Long Range modulation technique LoRa. The smart glasses, however, don't need LoRa. Therefore the ISP2053 has been chosen. It features a dual-core nRF53 with an embedded *Bluetooth* 5.2 module. This SoC's footprint is smaller than the one built into the devboard, making it ideal for the smart glasses system.

In addition to the SoC with built-in communication modules, a more power and energy-efficient *RISC-V* parallel processor, GAP9 from *Greenwaves Technologies*, is used for enabling computational intensive on-the-edge AI algorithm inference for image and audio processing using neural networks. GAP9 enables both parallelization and hardware acceleration while the GAP9 achieves the overall best trade-off performance in a few milliwatts envelope, considering latency, energy consumption per inference, and parallelization of the computation. In particular, the GAP9 SoC has a built-in general-purpose neural network accelerator capable of running SOTA algorithms' operations whenever the algorithm fits inside the memory. A FRAM is placed such that both GAP9 and nRF52 can access a shared memory space to share sensor data and system states. Additionally, GAP9 has a big amount of off-chip memory such as 254 MB of Flash and 64 MB DRAM.

The miniaturized smart glasses electronics shown in Fig. 1b, is a minimal electronics design, incorporating 2 cameras—GENX320 from Prophesee and HM0360 from Himax—, an accelerometer—LIS2DW—, a microphone and the above-described power management as well as BLE 5.2. The miniaturized design requires a high-precision PCB design with 8-impedance matched layers for the camera's CSI-2 protocol connection and the OctoSPI interface between GAP9 and external RAM and FLASH. However, for system development, the M2 PCB was utilized which features additional sensors, such as the eCompass—LSM303—, GNSS module—MIA-M10Q—, Microphones with wake-up capabilities—T5838—and an on-board micro-SD card storage for gigabytes of data collection. Removing these additional sensors of the M2 PCB allows all the electronics design to be shrunk into a trapezoidal form factor, Fig. 1b of 14 mm x 56 mm with a height of 3 mm to fit the temples, see Fig. 1a.

The Development Board—see Fig. 2—is designed for modular, rapid design and integration of various new sensors and system development. The main focus is on the design of GAP9's interfacing with different cameras using the CSI-2 protocol. The single-line CSI-2 interface is multiplexed to interface two cameras. Once a connector for interfacing commercial of the shelf COTS RGB cameras, designed for use with Raspberry Pis has been added. This interface is shared with a second camera connector, specifically designed for the new event-based camera GENX320 from Prophesee. As a third camera option, a Lepton3.5 is added to the development board for adding infrared camera capabilities and interfaced via SPI.

4.2 End-to-End System and Experimental Settings

First, we describe which hardware was used for system deployment, followed by the deployment procedure of the developed TinyissimoYOLO networks. Lastly, we describe the end-to-end system and how the experiments are conducted.

To showcase onboard intelligence and longer battery runtime, an end-to-end firmware has been designed and deployed on the GAP9 hardware. For the power evaluation, the development board together with the M2 PCB is used. Since the M2 system features the same hardware used for the end-to-end system, power measurements have been conducted with it. To maximize similarity, the sensors and peripherals not fitted in the miniaturized smart glasses PCB are switched off.

The deployment of the networks is done using *Greenwaves Technologies*' NNTool. Starting with the unquantized exported .onnx file from the *Ultralytics* framework, NNTool has an integrated PTQ flow. As such, the networks were deployed using 8-bit integer precision. Once quantized, the network can be auto-tiled by *Greenwaves Technologies*' Autotiler. This tool is used for automatic code generation, in particular, to generate C-code for the deployment onto the target cores or accelerator (NE16) of GAP9. Further, the user can specify the amount of L1, L2, and L3 memory to be used and the Autotiler tool will generate the network according to the constraints given such that the network is executed with minimal memory wait stalls, depending on each layer's size and operation used. Once the C-code has been generated, the network can be integrated into a custom user-specific project.

We implemented the end-to-end pipeline which is visualized in Fig. 5. First, a raw image in a Bayern pattern is captured. The image gets demosaiced on the Fabric Controller FC of GAP9. TinyissimoYOLO is used to predict objects while its network output is post-processed to extract the bounding boxes of the detected objects. The image-capturing process uses double-buffering, to decrease latency arising from the image sensor data acquisition. The demosaic process as well as the post-processing of the network's output utilizes the FC, while the object detection network runs on the cluster and on the neural engine.

The measurements of the networks are done by measuring the current consumption over 10 consecutive network inferences and calculating the average current consumption. Knowing the voltage level at which GAP9 is operating, the power as well as the energy consumption is calculated. We also calculate the number of Multiply and Accumulate MAC the network has to conduct one inference.

Lastly, the full end-to-end system is measured on board level, to incorporate the camera, both microcontrollers, the Power Management Integrate Ciruit PMIC, and all the voltage converter's inefficiencies.

5 Results

This chapter summarizes the results achieved in this work. First, the networks' detection accuracy is presented. Second the networks are deployed on the GAP9 and the achieved performance is reported. Lastly, the full end-to-end system results are provided including an always-on battery runtime estimation for the end-to-end system.

5.1 Detection Results

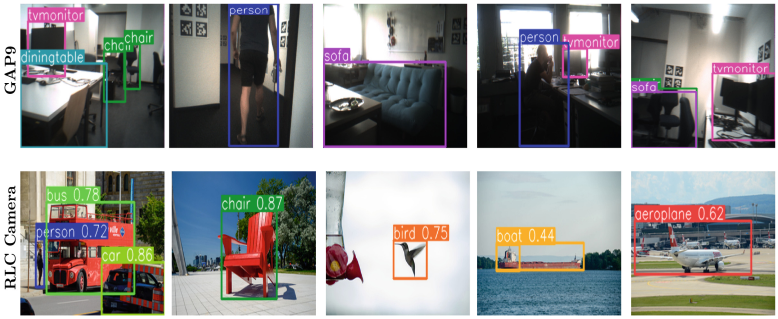
In this section, the performance evaluation of the various TinyissimoYOLO network configurations on the PascalVOC test dataset is presented. The results are summarized in Table 1 and visualized in Table 2 using images taken with GAP9—first row of Table 2—, and an Single-Lens ReflexSLR camera—second row. The evaluation of the newly proposed TinyissimoYOLO network with the different network configurations on the PascalVOC dataset reveals a noteworthy improvement in different network configurationsmAP scores. Specifically, transitioning from TinyissimoYOLOv1.3-Small to TinyissimoYOLOv8-Big or the TinyissimoYOLOv10 architecture results in a substantial mAP enhancement, showcasing the effectiveness of the latter in accurately detecting objects across various classes.

5.2 Deployed Networks on GAP9

The networks are 8-bit quantized and deployed on the NE16 accelerator of GAP9 and compared in terms of latency—Fig. 4b-b—, inference efficiency—Fig. 4b-c—, and energy per inference—Fig. 4b-d—for each TinyissimoYOLO network version v1.3, v5, v8-small/big and v10. TinyissimoYOLOv1.3 outperforms the other networks in terms of latency with only 16.9 ms execution time. V5 and both v8 variants need 32.7 ms, 34 ms, and 36.6 ms, respectively. In terms of inference efficiency, v1.3 is best parallelizable with up to 43.37 MAC/cycle. The v5 and v8 versions have 16.28 MAC/cycle, 14.98 MAC/cycle, 15.27 MAC/cycle, respectively. Since v1.3 performs fastest and most parallelized, it's also the most energy efficient consuming only 1.27 mJ, followed by v5 consuming 2.34 mJ, v8-small with 2.48 mJ and v8-big consuming 2.62 mJ.

Deploying the networks at different clocking frequencies of the NE16 core at different voltage levels leads to a Pareto Optima curve for running the networks quantized on the SoC of GAP9. Figure 4a shows the TinyissimoYOLOv1.3, v5, and v8 being deployed in both 'small' and 'big' versions. It shows that, while

Table 2. Qualitative Results: TinyissimoYOLOv8 running on images captured with our system (row 1) and with images taken using an SLR camera (row 2).



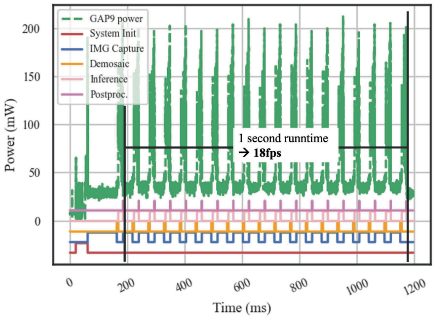
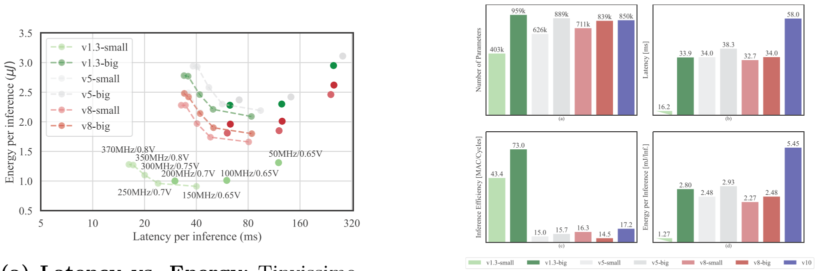


Fig. 3. Full System Power Measurement: The lines described in the legend show if the mentioned system process is running or not running. We show 18fps GAP9 on-device image-capturing, demosaicing, network inference and postprocessing execution.

v1.3-Small has the lowest detection robustness, as seen in Tab. 1, it benefits from the lowest inference latency and has the lowest energy consumption. Additionally, the frequency sweep for all the TinyissimoYOLO versions proposed can be used as a look-up table for finding the ideal network's execution frequency. Depending if the application allows for low energy consumption and low accuracy, running TinyissimoYOLOv1.3-small at 150 MHz would be the ideal choice or if the application needs the fastest inference time and most accuracy, running TinyissimoYOLOv8-Big at 370 MHz is the ideal choice.



(a) Latency vs. Energy: TinyissimoYOLOv1.3, v5 and v8 quantized deployed on GAP9's NE16 running at different voltage levels and different clocking frequencies.

(b) Deployed TinyissimoYOLO Versions (@0.8V; 370MHz): This comparison shows a) the number of parameters of each TinyissimoYOLO version b) the latency of each network, c) the inference efficiency, or energy consumption per inference and d) the MAC/cycle, or parallelization workload.

Fig. 4. Evaluation of the Deployed TinyissimoYOLO Versions on GAP9

5.3 System Results

Running the end-to-end solution on the GAP9 requires the system to initialize all required processing units of the SoC. Including the NE16 accelerator of GAP9, initializing the CSI-2 interface for the HM0360 camera from Himax, and creating all the needed memory buffers for image capturing and demosaicing/de-bayering. Once initialized, the system can run the system application loop, which consists of image capturing (double-buffered), demosaicing of the image, AI inference execution (TinyissimoYOLOv1.3), and post-processing of the network output. Figure 3 shows the power consumption during the full operation of the system, running the loop ten consecutive times. For the following power measurements, the loop was executed 100x and the average is reported. Capturing an image 100 times took on average 34.69 ms while consuming 1.17 mJ or 18.79 mA. Demosaicing needed only 4.87 ms while consuming 23.82 mA resulting in 0.209mJ of energy. Running the small TinyissimoYOLOv1.3 network on NE16 consumed 52.27 mA of current during 16.86 ms resulting in 1.59 mJ of Energy. The rest of the TinyissimoYOLO version is evaluated in Sect. 5.2. Table 3 summarizes the current, power, and energy measurements for the corresponding demonstration process. Last, the post-processing took 27 μ s, consuming 28.3 mA resulting in

1.38 μ J. With that, the average loop execution consumes 3.28 mJ, i.e. 29.55 mA for 61.67 ms.

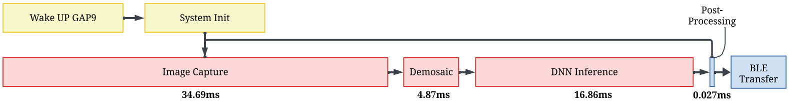


Fig. 5. End-to-End System Overview: The image shows the flow chart of the demonstrator firmware, including the execution latency for the corresponding task. The box sizes are in relative size to the execution time.

Table 3. System demonstrator: energy consumption measurement.

	Event	Current [mA]	Power [mW]	Energy [mJ]	Time [ms]
	System quiescent*	0.75	2.48	2.48	
	Init system	11.96	21.54	0.89	41.44
Loop	Capture image	18.78	33.82	1.17	34.69
	Demosaic non-cluster	23.82	42.88	0.21	4.87
	Run Tinyissimoyolov1.3	52.27	94.10	1.59	16.86
	Post processing	28.26	50.86	0.001	0.03
	Loop AVG.	30.0	54.0	3.05	56.45

* nRF wakes up from sleep mode, changes power settings, and sleeps again.

Battery Runtime Estimation. The smart glasses are charged with a battery, which fits inside the opposite temple of the smart-glasses' electronics, see Fig. 1a, with a maximal energy content of up to 154 mAh and 585 Wh with a nominal voltage level of around 3.8 V. The continuous execution of the presented end-to-end system consumes 30.0 mA at 1.8 V continuously and draws 54 mW from the battery. Including the HM0360 image sensor and the nRF MCU, the total system power consumption is 62.9 mW, resulting in a battery runtime of 9.3 h. In perspective the newly released *RayBan-Meta* smart glasses, with the same sized battery capacity, claim to last 4 h with moderate usage or up to 3 h of continuous audio streaming and voice assistance[7].

6 Conclusion

This paper proposed a novel smart glasses platform and demonstrated the system's capabilities. We perform image capturing and demosaicing, before running

[7] https://www.meta.com/ch/en/legal/ray-ban-meta/disclosures/.

AI inference and post-processing the networks' output to get bounding boxes. The end-to-end processing loop takes 56 ms and consumes 62.9 mW resulting in 18fps of continuous end-to-end execution for 9.3 h on a battery with 154 mAh. This sets a notable achievement for image processing on MCU class devices.

Further, this paper proposed a family of new TinyissimoYOLO versions, using the YOLOv3 detection layer and the YOLO version-specific head, while evaluating the architectures proposed in TinyissimoYOLOv1, YOLOv5, YOLOv8, and YOLOv10. The networks contain 50x to 100x fewer parameters than the initial YOLOv1 version and have been evaluated and compared on the PascalVOC and MS-COCO test datasets. The networks achieve sub-million parameters for up to 20 classes and fit quantized on MCUs. TinyissimoYOLOv8 with 840 k parameters, achieves 44% mAP while being executed within and 34 ms on the GAP9. The fastest small TinyissimoYOLOv1.3 is executed within 16.2 ms consuming only 1.27 mJ of energy for one inference and achieves 30% mAP. As such, this paper presents a highly generalized multi-class detection family of networks running with near SOTA performance detection accuracy in real-time (>18fps) on the GAP9 MCU.

Acknowledgment. The authors would like to thank Dominik Müller for his commitment during his semester's project. As well as armasuisse Science & Technology, and the Swiss National Science Foundation (SNSF) project 200021E_219943 Neuromorphic Attention Models for Event Data (NAMED) for funding this research.

References

1. Ali Hassan, E., Tang, T.B.: Smart glasses for the visually impaired people. In: Computers Helping People with Special Needs: 15th International Conference, ICCHP 2016, Linz, Austria, July 13-15, 2016, Proceedings, Part II 15, pp. 579–582. Springer (2016). https://doi.org/10.1007/978-3-319-41267-2_82
2. Bian, S., Wang, X., Polonelli, T., Magno, M.: Exploring automatic gym workouts recognition locally on wearable resource-constrained devices. In: 2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC), pp. 1–6 (2022https://doi.org/10.1109/IGSC55832.2022.9969370
3. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020).https://doi.org/10.48550/arXiv.2004.10934
4. Bonazzi, P., RÃegg, T., Bian, S., Li, Y., Magno, M.: Tinytracker: Ultra-fast and ultra-low-power edge vision in-sensor for gaze estimation. In: 2023 IEEE SENSORS, pp. 1–4 (2023https://doi.org/10.1109/SENSORS56945.2023.10325167
5. Chang, W.J., Chen, L.B., Hsu, C.H., Chen, J.H., Yang, T.C., Lin, C.P.: Medglasses: a wearable smart-glasses-based drug pill recognition system using deep learning for visually impaired chronic patients. IEEE Access **8**, 17013–17024 (2020). https://doi.org/10.1109/ACCESS.2020.2967400
6. Conti, F., et al.: Marsellus: A Heterogeneous RISC-V AI-IoT End-Node SoC With 2–8 b DNN Acceleration and 30%-Boost Adaptive Body Biasing. IEEE J. Solid-State Circ. **59**(1), 128–142 (2024). https://doi.org/10.1109/JSSC.2023.3318301

7. Crespo, J., Castillo, J.C., Mozos, O.M., Barber, R.: Semantic information for robot navigation: a survey. Appl. Sci. **10**(2) (2020). https://doi.org/10.3390/app10020497
8. Danielsson, O., Holm, M., Syberfeldt, A.: Augmented reality smart glasses for operators in production: Survey of relevant categories for supporting operators. Procedia CIRP **93**, 1298–1303 (2020). https://doi.org/10.1016/j.procir.2020.04.099, 53rd CIRP Conference on Manufacturing Systems 2020
9. Everingham, M., Eslami, S., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The Pascal visual object classes challenge: a retrospective. Int. J. Comput. Vision **111**(1), 98–136 (2014). https://doi.org/10.1007/s11263-014-0733-5
10. Feng, X., Jiang, Y., Yang, X., Du, M., Li, X.: Computer vision algorithms and hardware implementations: a survey. Integration **69**, 309–320 (2019). https://doi.org/10.1016/j.vlsi.2019.07.005
11. Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.: Yolox: exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430 (2021).https://doi.org/10.48550/arXiv.2107.08430
12. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint arXiv:1510.00149 (2015). http://arxiv.org/abs/1510.00149
13. Hong, I., et al.: A 2.71 nj/pixel gaze-activated object recognition system for low-power mobile smart glasses. IEEE J. Solid-State Circ. **51**(1), 45–55 (2016). https://doi.org/10.1109/JSSC.2015.2476786
14. Huang, J., Huang, G., Zhu, Z., Ye, Y., Du, D.: Bevdet: high-performance multi-camera 3d object detection in bird-eye-view. arXiv preprint arXiv:2112.11790 (2021). https://doi.org/10.48550/arXiv.2112.11790
15. Iqbal, M.Z., Campbell, A.G.: Adopting smart glasses responsibly: potential benefits, ethical, and privacy concerns with ray-ban stories. AI and Ethics **3**(1), 325–327 (2023). https://doi.org/10.1007/s43681-022-00155-7
16. Islamoglu, G., et al.: Ita: an energy-efficient attention and softmax accelerator for quantized transformers. In: 2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp. 1–6 (2023).https://doi.org/10.1109/ISLPED58423.2023.10244348
17. Jiang, P., Ergu, D., Liu, F., Cai, Y., Ma, B.: A review of yolo algorithm developments. Procedia Computer Science **199**, 1066–1073 (2022). https://doi.org/10.1016/j.procs.2022.01.135, the 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021): Developing Global Digital Economy after COVID-19
18. Jocher, G.: YOLOv5 by Ultralytics (May 2020). https://doi.org/10.5281/zenodo.3908559, https://github.com/ultralytics/yolov5
19. Jocher, G., Chaurasia, A., Qiu, J.: YOLO by Ultralytics (Jan 2023). https://github.com/ultralytics/ultralytics
20. Lee, L.H., Hui, P.: Interaction methods for smart glasses: a survey. IEEE Access **6**, 28712–28732 (2018). https://doi.org/10.1109/ACCESS.2018.2831081
21. Li, C., et al.: Yolov6: A single-stage object detection framework for industrial applications. arXiv preprint arXiv:2209.02976 (2022). https://doi.org/10.48550/arXiv.2209.02976
22. Li, Z., et al.: Bevformer: learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In: European Conference on Computer Vision, pp. 1–18. Springer (2022). https://doi.org/10.1007/978-3-031-20077-9_1
23. Lin, J., Chen, W.M., Cai, H., Gan, C., Han, S.: MCUNetV2: Memory-efficient patch-based inference for tiny deep learning. arXiv preprint arXiv:2110.15352 (2021). https://doi.org/10.48550/arXiv.2110.15352

24. Lin, J., Chen, W.M., Lin, Y., Cohn, J., Gan, C., Han, S.: MCUNet: tiny deep learning on IoT devices. In: Advances in Neural Information Processing Systems (2020). http://arxiv.org/abs/2007.10319
25. Lin, J.Y., Chiang, C.L., Wu, M.J., Yao, C.C., Chen, M.C.: Smart glasses application system for visually impaired people based on deep learning. In: 2020 Indo–Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN), pp. 202–206 (2020). https://doi.org/10.1109/Indo-TaiwanICAN48429.2020.9181366
26. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
27. Liu, C., Chen, S., Tsai, T.H., de Salvo, B., Gomez, J.: Augmented reality - the next frontier of image sensors and compute systems. In: 2022 IEEE International Solid-State Circuits Conference (ISSCC). vol. 65, pp. 426–428 (202https://doi.org/10.1109/ISSCC42614.2022.9731584
28. Mccormac, J., Clark, R., Bloesch, M., Davison, A., Leutenegger, S.: Fusion++: Volumetric object-level slam. In: 2018 International Conference on 3D Vision (3DV), pp. 32–41 (2018). https://doi.org/10.1109/3DV.2018.00015
29. Miah, M.R., Hussain, M.S.: A unique smart eye glass for visually impaired people. In: 2018 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE), pp. 1–4 (2018).https://doi.org/10.1109/ICAEEE.2018.8643011
30. Moin, A., et al.: A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition. Nature Electron. **4**(1), 54–63 (2021). https://doi.org/10.1038/s41928-020-00510-8
31. Moosmann, J., Giordano, M., Vogt, C., Magno, M.: Tinyissimoyolo: a quantized, low-memory footprint, tinyml object detection network for low power microcontrollers. In: 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS), pp. 1–5 (2023). https://doi.org/10.1109/AICAS57966.2023.10168657
32. Moosmann, J., Müller, H., Zimmerman, N., Rutishauser, G., Benini, L., Magno, M.: Flexible and fully quantized lightweight tinyissimoyolo for ultra-low-power edge systems. IEEE Access **12**, 75093–75107 (2024). https://doi.org/10.1109/ACCESS.2024.3404878
33. Nahavandi, D., Alizadehsani, R., Khosravi, A., Acharya, U.R.: Application of artificial intelligence in wearable devices: Opportunities and challenges. Comput. Methods Programs Biomed. **213**, 106541 (2022). https://doi.org/10.1016/j.cmpb.2021.106541
34. Raj, V., Vigneswaran, E., Deshnaa, M., RajPrasanth, K.: Virtual smart glass for blind using object detection. In: 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 1419–1424 (2022).https://doi.org/10.1109/ICSSIT53264.2022.9716412
35. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788 (2016). https://doi.org/10.1109/CVPR.2016.91
36. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6517–6525 (2017). https://doi.org/10.1109/CVPR.2017.690
37. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018). https://doi.org/10.48550/arXiv.1804.02767

38. Reuther, A., Michaleas, P., Jones, M., Gadepally, V., Samsi, S., Kepner, J.: Survey of machine learning accelerators. In: 2020 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–12 (2020). https://doi.org/10.1109/HPEC43674.2020.9286149
39. Rosinol, A., Abate, M., Chang, Y., Carlone, L.: Kimera: an open-source library for real-time metric-semantic localization and mapping. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 1689–1696 (2020). https://doi.org/10.1109/ICRA40945.2020.9196885
40. Saha, S.S., Sandha, S.S., Srivastava, M.: Machine learning for microcontroller-class hardware: A review. IEEE Sens. J. **22**(22), 21362–21390 (2022). https://doi.org/10.1109/JSEN.2022.3210773
41. Salvaro, M., Benatti, S., Kartsch, V., Guermandi, M., Benini, L.: A wearable device for brain–machine interaction with augmented reality head-mounted display. In: EAI International Conference on Body Area Networks, pp. 339–351. Springer (2018https://doi.org/10.1007/978-3-030-29897-5_29
42. Scherer, M., Sidler, F., Rogenmoser, M., Magno, M., Benini, L.: Widevision: A low-power, multi-protocol wireless vision platform for distributed surveillance. In: 2022 18th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 394–399 (2022). https://doi.org/10.1109/WiMob55322.2022.9941670
43. Spandonidis, C., et al.: Development of visual sensors and augmented reality based smart glasses for enhancement of business sustainability and wellbeing of the aging workforce. IEEE Instrument. Measure. Mag. **26**(6), 21–27 (2023). https://doi.org/10.1109/MIM.2023.10217030
44. Subhan, F., et al.: Ai-enabled wearable medical internet of things in healthcare system: a survey. Appl. Sci. **13**(3) (2023). https://doi.org/10.3390/app13031394
45. Sun, H., Kim, K.: Design of glasses products based on artificial intelligence. In: The International Conference on Cyber Security Intelligence and Analytics, pp. 1051–1058. Springer (2022).https://doi.org/10.1007/978-3-030-96908-0_130
46. Wang, A., et al.: Yolov10: Real-time end-to-end object detection. arXiv preprint arXiv:2405.14458 (2024). https://doi.org/10.48550/arXiv.2405.14458
47. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: Yolov7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7464–7475 (2023). https://doi.org/10.1109/CVPR52729.2023.00721
48. Wang, C.Y., Mark Liao, H.Y., Wu, Y.H., Chen, P.Y., Hsieh, J.W., Yeh, I.H.: Cspnet: a new backbone that can enhance learning capability of cnn. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1571–1580 (2020). https://doi.org/10.1109/CVPRW50498.2020.00203
49. Wang, X., Hersche, M., Magno, M., Benini, L.: Mi-bminet: an efficient convolutional neural network for motor imagery brain–machine interfaces with eeg channel selection. IEEE Sens. J. **24**(6), 8835–8847 (2024). https://doi.org/10.1109/JSEN.2024.3353146
50. Wang, X., et al.: Embedded classification of local field potentials recorded from rat barrel cortex with implanted multi-electrode array. In: 2018 IEEE Biomedical Circuits and Systems Conference (BioCAS), pp. 1–4 (2018).https://doi.org/10.1109/BIOCAS.2018.8584830
51. Yu, G., et al.: Pp-picodet: a better real-time object detector on mobile devices. arXiv preprint arXiv:2111.00902 (2021). https://doi.org/10.48550/arXiv.2111.00902

52. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: an extremely efficient convolutional neural network for mobile devices. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6848–6856 (2018). https://doi.org/10.1109/CVPR.2018.00716
53. Zhao, W.X., et al.: A survey of large language models. arXiv preprint arXiv:2303.18223 (2023). https://doi.org/10.48550/arXiv.2303.18223