This paper has been accepted for publication at the
IEEE Conference on Artificial Intelligence Circuits and Systems (AICAS), Hangzhou, 2023.

TinyissimoYOLO: A Quantized, Low-Memory Footprint, TinyML Object Detection Network for Low Power Microcontrollers

Julian Moosmann, Marco Giordano, Christian Vogt, Michele Magno
Center for Project Based Learning - ETH Zürich
julian.moosmann, marco.giordano, christian.vogt, michele.magno@pbl.ee.ethz.ch

***Abstract*—This paper introduces a highly flexible, quantized, memory-efficient, and ultra-lightweight object detection network, called TinyissimoYOLO. It aims to enable object detection on microcontrollers in the power domain of milliwatts, with less than $0.5\,$MB memory available for storing convolutional neural network (CNN) weights. The proposed quantized network architecture with $422\,$k parameters, enables real-time object detection on embedded microcontrollers, and it has been evaluated to exploit CNN accelerators. In particular, the proposed network has been deployed on the MAX78000 microcontroller achieving high frame-rate of up to $180\,$fps and an ultra-low energy consumption of only $196\,\mu$J per inference with an inference efficiency of more than $106\,$MAC/Cycle. TinyissimoYOLO can be trained for any multi-object detection. However, considering the small network size, adding object detection classes will increase the size and memory consumption of the network, thus object detection with up to 3 classes is demonstrated. Furthermore, the network is trained using quantization-aware training and deployed with 8-bit quantization on different microcontrollers, such as STM32H7A3, STM32L4R9, Apollo4b and on the MAX78000's CNN accelerator. Performance evaluations are presented in this paper.**

***Index Terms*—YOLO, ML, computer vision, object detection, CNN accelerator, microcontroller, quantization, quantization-aware training**

I. Introduction

Object detection on edge devices such as microcontroller units (μCs) have the capability of reducing detection latency and increasing the overall energy efficiency by running on-device network inferences [1], [2]. In addition, the sensitive transmission of sensor data is reduced or substituted, reducing privacy issues to a minimum. Furthermore, emerging dedicated hardware accelerators for machine learning (ML) models are enabling edge processing and edge artificial intelligence (AI) reducing the energy consumption for inference and enabling real-time on-board processing on resource-constrained microcontrollers [3]. On such constrained devices, computational power is significantly reduced and does not allow for the deployment of classical object detection deep neural networks (DNNs) such as you only look once (YOLO) [4], region-based convolutional neural networks (R-CNNs) [5] or single shot detectors (SSDs) [6] because their memory requirements are significantly exceeding the available memory of few kilobytes typically available in such devices. However, their fundamental ideas are of importance for designing new DNNs, especially for the emerging field of edge AI.

The authors would like to thank *armasuisse Science & Technology* for funding this research.

In fact, to keep the power consumption in the order of a few milliwatts, the computational power on μCs is significantly lower compared to CPUs and GPUs. Thus, object detection networks need to be carefully designed and optimized for such tiny devices in particular to achieve a small memory footprint and a high number of operations processed per cycle [7]. To overcome the memory constraints, several different methods have been recently reported in literature, such as pruning [8], [9], quantization [10], [11], new frameworks developed for memory efficient inference [12], patch-based inference scheduling [13] or neural architecture searches including search spaces specialized for memory-constrained devices [14], [15]. These techniques are used to reduce the memory and complexity of existing state-of-the-art networks to deploy on a tiny device while keeping the original network's structure and still achieving similar inference accuracy while performance (especially inference speed) is often neglected [16]. Among other techniques, quantization is one of the most promising and popular, as it reduces both the memory requirements and increases the number of operations per second that μCs can perform. Li et al. (2019) [17] have shown that quantizing an object detection network to 4-bit, they achieve state-of-the-art prediction accuracy with a mean average precision (mAP) loss of only $2\,\%$ to $5\,\%$ compared to its full precision counterpart while having 8x less memory occupation.
To the best of our knowledge, there is still no previous work that adopts those techniques to achieve a generalized object detection network, ready for deployment on edge devices and μCs with less than $0.5\,$MB of weight memory, that are able to achieve similar accuracy performance of larger networks.

This paper proposes a quantized and highly accurate object detection convolutional neural network (CNN) based on the architecture of YOLO [4] suitable for edge processors with limited memory and computational resources. The proposed network is composed of quantized convolutional layers with 3x3 kernels and a fully connected layer at the output. It is designed for having a low memory footprint of less than $0.5\,$MB. The proposed network is trained and evaluated on the WiderFace dataset [18]. Furthermore, to showcase multi-object detection capability, while keeping the network small, it has been trained and evaluated on a sub-set of the PascalVOC [19] dataset (3 out of the 20 classes, namely: person, chair and car). Finally, the network is deployed quantized and memory-efficient on different μC architectures, such as the STM32H7A3 and STM32L4R9 from STMicroelectronics, Ambiq's Apollo4b and on a novel microcontroller, MAX78000 from Analog Devices, which has a built-in CNN accelerator. The performance of the different architectures is compared and it will be shown, how the MAX78000 outperforms the other μCs. Furthermore, this paper investigates the effect of mAP against relative object size dimensions within images. This evaluation helps to understand which object size should be chosen when training the generalized object detection network.

II. TinyissimoYOLO

The ultra-lightweight object detection network designed in this paper for operation on μC, named TinyissimoYOLO, is a general object detection network and can be seen in Figure 1. It can be deployed on any hardware with at least $0.5\,\mathrm{MB}$ of network parameter memory, i.e. flash, available. Not only is it capable of being deployed on any ARM Cortex-M microcontrollers, but it can also be deployed on μCs with built-in hardware accelerators, such as the MAX78000 from Analog Devices, GAP9 [20] from Greenwaves, Sony' Spresense [21] or Syntiant TinyML [22], among others. The network can be trained on any object detection class and supports multi-class object detection with the cost of minimal increasing memory.

To facilitate small size and to be deployable in an efficient way on all existing microcontrollers, as well as exploiting CNN emerging hardware accelerators that are starting to be embedded in recent microcontrollers, TinyissimoYOLO has been designed to consist of only convolutional layer-operations and fully connected linear layers, which are largely optimized in hardware and software toolchains. As we mentioned, the goal of the design is to reach outstanding accuracy performance with a memory envelope of $0.5\,\mathrm{MB}$ and with a minimal number of operations and yet a high number of operations per cycle.

A. Network design

TinyissimoYOLO is inspired by the YOLOv1 [4] architecture. Major design decisions of the TinyissimoYOLO network encompass the following 4 points:

- input image size,
- hidden convolutional layers,
- fully connected linear layer before the output layer,
- output layer size.

Input image size has been chosen such that the network runs on all the previously mentioned μCs. Limiting factor is the CNN accelerator of the MAX78000, which does not support CNN inputs greater than 90x91 without using a specialized mode. Thus the input of 88x88 is chosen because it is a tread-off between maximizing the image size and being able to do pooling on the input dimensions without rounding the dimensions down to the 4th pooling layer. The network's **hidden convolutional layer** parameter memory scales by the number of input channels times the number of output channels times the filter size. Even though in the original YOLOv1 network high channel count layers are preferred, the small memory size on μCs and accelerators will not allow such layers. For example, one convolutional layer with 256 input and output channels and a kernel of 3x3, consumes already more than $0.5\,\mathrm{MB}$ of quantized 8-bit weights. To ensure staying below $0.5\,\mathrm{MB}$ of network memory, only convolutional layers with 3x3 filters are used, since it scales the weight memory only by a factor of 9x, instead of 25x (for 5x5 kernel) or 49x (for a 7x7 kernel). Even though it would be possible to use bigger filter sizes (e.g. YOLOv1 used a filter of 7x7 in the input) we focused on using convolutional layers with 3x3 filters and keeping the network as simple as possible. Furthermore, we increase the number of channels with increasing network depth, starting with 16 output channels in the CNN's input layer and ending with 128 output channels at the CNN's output. Max pooling with a stride of two decreases the channel resolution every second layer. Thereafter, the fully connected layer predicts the output probabilities and coordinates out of the features extracted by the former convolutional layers. As such, the **fully connected linear layer which connects the CNN to the output layer**, is chosen to be 256 and therefore is larger than the output layer size of 4x4(2xBbox+C) for mapping the CNN's features onto the regression, classification output. The network's **output layer** uses the YOLOv1 output of the size SxS(BxBbox+C). Where SxS is the grid, within each cell B bounding box (Bbox) (with the objects width, height, centre in x- and y-axis, relative to the grid cell size and prediction certainty) of possible detections are calculated by the network as well as the C possibly detected classes. As the network's input image size is set to 88x88 pixels a smaller output grid size of 4x4 and $B = 2$ is used. The value of B was found as a trade-off between significantly more memory consumption and having less prediction accuracy. $B = 3$, increased the output layer memory consumption by a factor of 1.5x but did not increased the mAP significantly. Therefore, $B = 2$ has been used for training the network on WiderFace. To ensure deployability on the CNN accelerator of MAX78000 and training a multi-object detection network for 3 classes of PascalVOC, $B = 1$ has been chosen. The single-class TinyissimoYOLO with $B = 2$ is $106\,\%$ bigger than the multi-class TinyissimoYOLO with $B = 1$. For training the TinyissimoYOLO network, we use the YOLO loss function introduced in the YOLOv1 paper [4] by Redmon et al. (2016) to train the network. Evaluation is done by using the mAP as it has been used by Everingham et al. (2015) [19].

B. Training and Dataset

For training, testing and validation of the TinyissimoYOLO network, the WiderFace dataset [18] was used. $90\,\%$ of WiderFace training dataset was used to train the network, while $10\,\%$ was used for validation. Testing of the network was done with the corresponding test dataset of WiderFace. Due to TinyissimoYOLO's small input size of 88x88x3, and the 4x4 network output grid predictions, we evaluated the network's

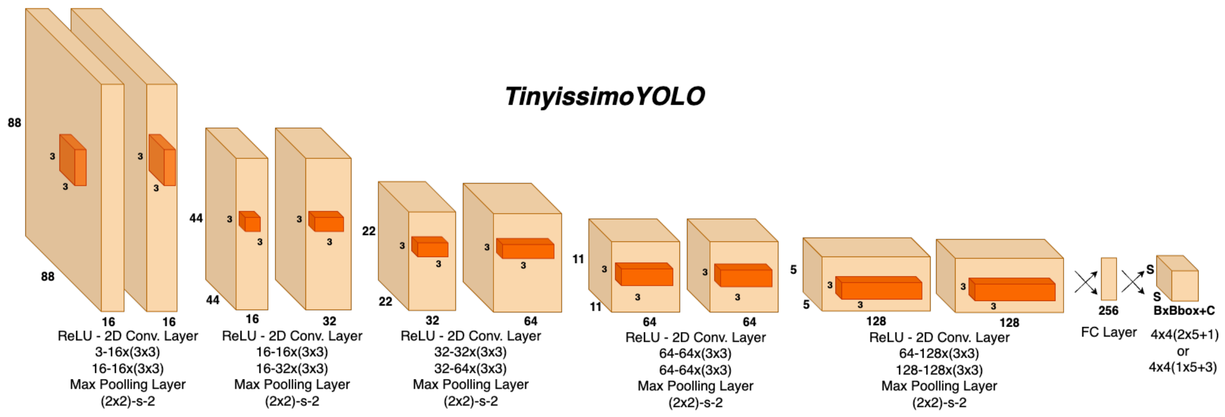

Fig. 1. TinyissimoYOLO proposed by this paper.

accuracy by restricting the training data to images containing less or equal to 10 and 5 objects. As such, the objects are ensured to be visible in the downscaled 88x88 pixel images. With the restriction made on the training dataset of WiderFace, we evaluate the influence of the number of objects within an image on mAP. Additionally, for showing the network's ability for general multi-object detection, the network is trained and evaluated on the PascalVOC dataset [19] with a restriction of maximal three objects per image, due to the increased difficulty of the multi-class object detection problem. To not only deploy the network on the STM32s and Apollo4b but also to fit in the MAX78000 CNN accelerator's memory, the multi-object TinyissimoYOLO is trained and evaluated using only 3 of the 20 classes, namely person, chair and car and thus stays bellow the sub-0.5 MB memory size restriction set by us. The 3 classes were chosen because of their object occurrences within the dataset and thus to train the network with a balanced dataset. Deploying the network on real hardware, experimental tests can be conducted with many different sceneries inside and outside of buildings.

Because the network is designed to be deployable on μCs and on CNN accelerators in the most efficient way, the network is quantized from 32-floating point to 8-bit integers. This further reduces computation cycles and memory requirements. In order to optimize the network's fine-tuning, the training is already conducted aware of this quantization. The network is trained 350 epochs with floating-point precision and thereafter switches to another 300 epochs trained quantization-aware. The starting learning rate as well as the weight decay is set to be $5 * 10^{-4}$. The used optimizer is SGD [23]. The learning rate is scheduled with a multi-step learning rate for improved learning speed [24].

C. μC implementation

By experimental evaluation, we demonstrate that TinyissimoYOLO is efficiently deployed on several low-power devices, such as STM32L4R9 with an ARM Cortex-M4F core, STM32H7 with an ARM Cortex M7 core, Apollo4b the world's more energy-efficient ARM Cortex-M4F core, and MAX78000 (using its built-in CNN accelerator). Deploying the designed and quantized network on the μCs's ARM Cortex M cores, is straightforward, as we designed the network to be compatible with TensorFlow-lite micro [25]. Next to the μC only deployment, it is shown that the network can also operate well on a μC with a built-in CNN accelerator (MAX78000) by deploying it with the provided deployment tools from Analog Devices [26]. TinyissimoYOLO fits the MAX78000's CNN accelerator's limitations. The most important considerations for the MAX78000 are that the allowed network memory footprint of its weights can only be 442kB. Considering the data passed through the architecture, the input size is constrained to <90x91 input width and height, when using the non-streaming mode of the CNN accelerator. All network operations needed by TinyissimoYOLO are covered by the design of the hardware accelerator and its software deployment tools. A full pipeline demonstrator with MAX78000FTHR μC and on-device image sensor CameraCubeChip®is built and used for validating the system with real-life data.

III. Experimental Results

The networks have been trained on a subset of the WiderFace [18] and the PascalVOC datasets as explained in section *B. Training and Dataset*. They have been evaluated on the whole test dataset of WiderFace as well as on the restricted datasets. Table I lists the evaluation of the trained TinyissimoYOLO networks and compares the mAP [19] when trained with a different number of max. objects allowed per image. On WiderFace, TinyissimoYOLO achieves a mAP [19] of 45 % when evaluated on the whole dataset and up to 73 % mAP when restricting the evaluation dataset of the network to max. 5 objects per image. Restricting the dataset during training to max. 10 or 5 objects per image, a mAP of 46 % and 44 % mAP is achieved when evaluating on the whole dataset. Evaluating the restricted trained networks on the restricted datasets up to 75 % mAP is achieved with an increase of 2 % compared to training the network on the whole dataset. The TinyissimoYOLO network trained and evaluated on the PascalVOC dataset for the object detection classes person, chair and car achieves mAP of up to 57 %, see Table II. Here a global restriction to maximal three objects per image is made because of the increased difficulty of the multi-class object detection problem.

TABLE I
TINYISSIMOYOLO NETWORK TRAINED AND EVALUATED ON WIDERFACE (FACE DETECTION). MAXIMAL ALLOWED OBJECTS PER IMAGE HAVE BEEN RESTRICTED FOR TRAINING AND EVALUATION.

	mAP		
WiderFace	**no restriction**	**max 10 obj.**	**max 5 obj.**
no restriction	45.3 %	64.1 %	73.5 %
max 10 obj.	46.2 %	65.7 %	75.4 %
max 5 obj.	43.5 %	62.4 %	72.7 %

TABLE II
TINYISSIMOYOLO NETWORK TRAINED AND EVALUATED ON PASCCALVOC WITH CLASSES: PERSON, CHAIR AND CAR. MAXIMAL ALLOWED OBJECTS PER IMAGE HAVE BEEN RESTRICTED TO 3 OBJECTS FOR TRAINING AND EVALUATION.

	mAP			
Dataset	**person**	**chair**	**car**	**overall**
PascalVOC[a]	57.4 %	30.2 %	65.1 %	58.5 %

[a] training dataset with person / chair / car classes

Given the TiniyssimoYOLO's performance on the datasets, the TinyissimoYOLO networks (trained on WiderFace and PascalVOC) are now compared when deployed quantized to 8-bits on the different μCs. Since both networks yield the exact same performance results, no network distinction is made for the results on the following metrics:

- power efficiency [μW/MHz],
- inference efficiency [MAC/Cycles] (which tells how well the device can parallelise the network execution),
- inference latency [ms],
- energy per inference [mJ/Inf.].

The metrics are measured by deploying TinyissimoYOLO on the different target devices and measuring the power consumption of the μCs only with the following μC configurations: MAX78000 @ (1.2 V; 50 MHz), STM32H7A3 @ (3.3 V; 192 MHz), STM32L4R9 @ (3.3 V; 120 MHz) and Apollo4b @ (1.8 V; 192 MHz). The metrics are chosen as proposed by Giordano et al. (2022) [27] to evaluate the architecture's latency, power and energy efficiency as well as the hardware's capability of parallelising the network's computational workload to its processor(s). Figure 2 a), compares the latency of the network being executed on the different μCs. Due to the usage of the custom CNN hardware accelerator, MAX78000 runs one inference within 5.5 ms while the second-fastest is STM32H7A3 with 359 ms. Therefore, MAX78000 outperforms the others by a factor of >65x. Figure 2 b) shows the inference efficiency of the different architectures. The accelerated CNN hardware of MAX78000 showcases its ability to parallelise the CNN workload with 107 MAC/Cycle while all the others need at least 2-4 cycles to execute one MAC. In Figure 2 c) Apollo4b outperforms all the others by only consuming 59 μW/MHz and thus being the most power efficient μC within this comparison. Figure 2 d) shows the energy efficiency of the devices. Despite the fact that the Apollo4b has best-in-class power efficiency, the MAX78000 hardware accelerator's fast and inference-efficient execution manages to be the overall, most energy efficient with only 196 μJ per Inference and outperforming by a factor of 32x compared to the second best, being the Apollo4b with 6.1 mJ per Inference. Last but not least the TinyissimoYOLO network consumes 422 kB of memory when trained on 1 object class, e.g. WiderFace and 398 kB when trained on 3 object classes, e.g. PascalVOC (person, chair and car). The input image of 88x88x3 pixels consumes an additional 25 kB of memory space, while 350 kB of RAM (when deployed on the ARM Cortex-M cores) is sufficient for executing an inference.

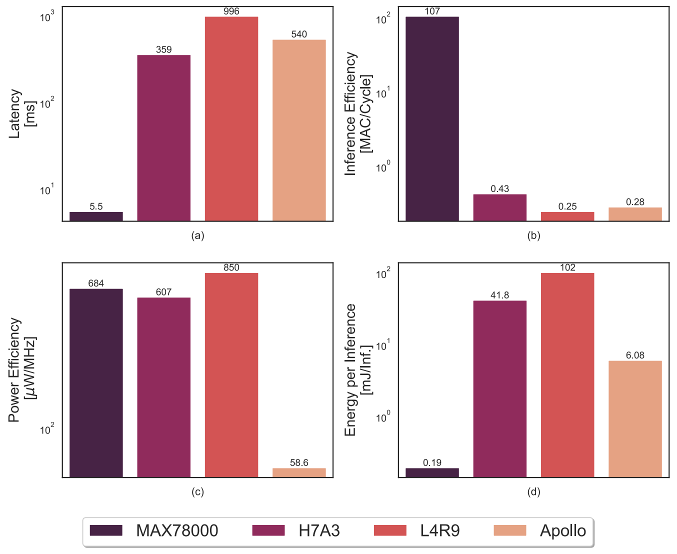

Fig. 2. TinyissimoYOLO performance comparison when deployed quantized to 8-bit on the different architectures. The CNN accelerated MAX78000 μC outperforms the other architectures in terms of latency, inference efficiency and energy per inference.

IV. CONCLUSION

This work presented TinyissomoYOLO, a multi-object detection network showcased to be used for edge applications with small amount of objects to be detected simultaneously. The network can be deployed on any μC with a minimal required flash of less than 0.5 MB for its network parameters. Evaluations showed, training the network on input images with objects adequate for its input size (restricting the dataset to maximal 10 or 5 objects per image) increases not only network performance on a restricted evaluation but also when evaluating the network on the original validation dataset, achieving up to 45 % mAP for WiderFace. Evaluating the network on a restricted PascalVOC dataset, the network achieves 59 % mAP with 57 % / 30 % / 65 % on the classes person / chair / car respectively. Comparing low power μC devices when running the TinyissimoYOLO network on those, reveals, the MAX78000's CNN accelerated hardware is the most energy efficient among the evaluated devices and is able to achieve fast inference times running in real-time on μCs with speeds up to 180 fps.