RaScaNet: Learning Tiny Models by Raster-Scanning Images

Jaehyoung Yoo[1,2] Dongwook Lee[1] Changyong Son[1] Sangil Jung[1]
ByungIn Yoo[1] Changkyu Choi[1] Jae-Joon Han[1] Bohyung Han[2]

[1]Samsung Advanced Institute of Technology (SAIT), South Korea
[2]Computer Vision Lab., Seoul National University, South Korea

Abstract

*Deploying deep convolutional neural networks on ultra-low power systems is challenging due to the extremely limited resources. Especially, the memory becomes a bottleneck as the systems put a hard limit on the size of on-chip memory. Because peak memory explosion in the lower layers is critical even in tiny models, the size of an input image should be reduced with sacrifice in accuracy. To overcome this drawback, we propose a novel **Ra**ster-**Sca**nning **Net**work, named RaScaNet, inspired by raster-scanning in image sensors. RaScaNet reads only a few rows of pixels at a time using a convolutional neural network and then sequentially learns the representation of the whole image using a recurrent neural network. The proposed method operates on an ultra-low power system without input size reduction; it requires 15.9–24.3× smaller peak memory and 5.3–12.9× smaller weight memory than the state-of-the-art tiny models. Moreover, RaScaNet fully exploits on-chip SRAM and cache memory of the system as the sum of the peak memory and the weight memory does not exceed 60 KB, improving the power efficiency of the system. In our experiments, we demonstrate the binary classification performance of RaScaNet on Visual Wake Words and Pascal VOC datasets.*

1. Introduction

With recent advances in deep learning, complex computer vision applications run on edge devices, which results in various benefits, including improving privacy, reducing power consumption, and personalizing predictions. There has been emerging interest in expanding the scope of machine learning to ultra-low power systems, called TinyML [2]. The goal of TinyML is to perform various always-on use-cases, typically in the *mW* range and below, powered by general purpose microcontroller units (MCUs) or application specific integrated circuits (ASICs) [1].

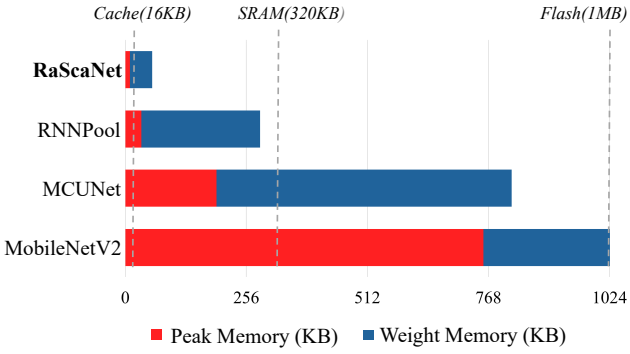

Figure 1. Comparison between RaScaNet and existing tiny models in terms of memory usage. RaScaNet not only fits into SRAM but also takes advantage of cache memory.

Despite many successes of deep neural networks in computer vision applications [13, 20, 38], it is still challenging to deploy a convolutional neural network (CNN) on an MCU due to the on-chip memory restrictions and the problems related to power consumption and latency [12, 21]. Typical microcontrollers only have limited on-chip memory of 100–320 KB (SRAM), and the peak memory of TinyML models should not exceed the on-chip SRAM. Moreover, it is highly recommended to maintain the sum of the peak memory and the weight memory below the size of on-chip SRAM. Otherwise, we have to fetch the partial model weights layer by layer from the flash storage (256 KB–1 MB), which increases both read access time and cache miss ratio.

While previous works [22, 30] focused only on reducing the peak memory below the on-chip SRAM (320 KB), our method reduces both peak and weight memory significantly even below 60 KB (Fig. 1). Such ultra-low memory footprint is crucial for low-power platforms, as it reduces the number of accesses to memory.

Considering the objective of running TinyML with the limited on-chip memory and power, the conventional CNNs fetching a full-frame image at once would be inappropriate.

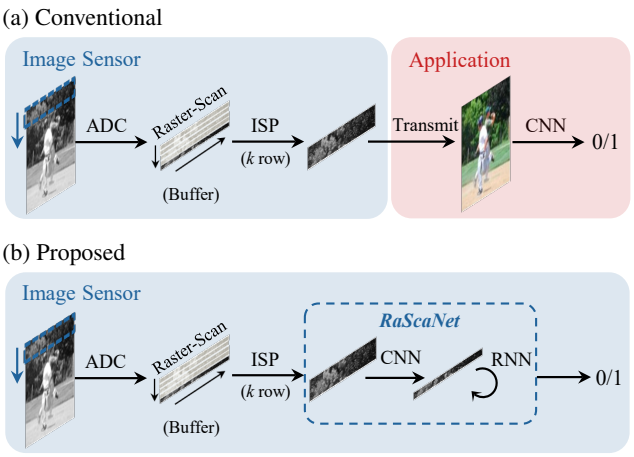

Figure 2. The pixels in image sensor are first digitized by an analog-to-digital converter (ADC) in a raster-scan order. Then, the converted pixel rows are stored in the buffer that can hold only k rows for image signal processing (ISP) algorithms. To process conventional CNNs, (a) the processed rows are transmitted to the applications, which have to wait until the whole image is received. On the other hand, (b) RaScaNet can operate on sensor with minimum peak memory by exploiting the data stream of an image sensor.

Even a small model is not free from the peak memory explosion due to the quadratic increase of memory requirements with respect to image size.

To tackle this problem, we propose a novel architecture, named RaScaNet, which processes non-overlapping sub-images sequentially instead of processing full-frame images as a whole. This aligns well with the modern image sensors that read a small number of rows at a time sequentially. By exploiting the data stream of an image sensor, RaScaNet can operate on sensor with minimum peak memory (Fig. 2).

To be specific, the proposed algorithm scans an input image from top to bottom using a CNN. Next, a series of the features obtained from the CNN are sequentially fed into a recurrent neural network (RNN) to learn the final representation of the image. In this model, the RNN learns semantic representations based on the low-level information encoded by the CNN.

Our contributions are summarized as below:

1. We propose a novel deep neural network architecture to fit in the ultra-low power systems, called RaScaNet, which processes input images in a raster-scan manner.

2. RaScaNet achieves state-of-the-art Pareto efficiency in accuracy vs. memory; it requires 15.9–24.3× smaller peak memory and 5.3–12.9× smaller weight memory than the state-of-the-art tiny models while still presenting competitive accuracy.

3. We design three components in RaScaNet—multi-head CNN, attention mechanisms, and confidence loss—for its effectiveness on practical computer vision tasks. We also introduce an early termination scheme for further acceleration.

2. Related Work

Recent studies on efficient deep learning can be categorized into the following four groups. First, some approaches incorporate efficient operations, such as depthwise separable convolution [6], to reduce computational cost without sacrificing accuracy. Xception [6], MobileNet [15, 16, 31], and EfficientNet [33] are well-known models that use depthwise separable convolutions. Second, model compression techniques, such as quantization [5, 18, 39] and pruning [26, 27], are employed to reduce the model size. Third, neural architecture search (NAS) techniques identify optimized models [24, 25, 29] via automated learning procedure without hand-crafted heuristics. Several NAS approaches [3, 14, 32] aimed to learn stronger models through the hardware feedback in the search loop. Last, there is a stream of work for improving the computational cost by using small image patches. GFNet [35] performs efficient image classification by processing a sequence of relatively small input patches, which are selected from the original image dynamically. ViT [10] utilizes fewer computational resources to train but achieves comparable performance to convolutional networks using Transformer [34] with 16x16 image patches as an input.

Although the aforementioned techniques achieve significant advances in efficient deep learning, deploying deep neural networks on an MCU is much more challenging, especially because of their extremely limited memory capacity. To reduce the peak memory requirements of networks, Saha *et al*. [30] proposed to use RNN pooling (RNNPool), which reduces the size of the activation maps in the first few layers by 4–8× using RNNs. This is based on the observation that the peak memory usage depends mostly on the layers in the early stages of CNNs. However, RNNPool still suffers from peak memory explosion when dealing with high resolution images because it employs conventional convolutions after the RNN pooling layer. Contrary to RNNPool, our approach adopts an RNN for sequential processing of CNN features by scanning an image from top to bottom.

Lin *et al*. [22] have introduced a co-design framework, referred to as MCUNet, based on a NAS method for resource constrained systems (TinyNAS) along with a lightweight inference engine (TinyEngine). MCUNet achieves the state-of-the-art accuracy with smaller peak memory than the model learned by ProxylessNAS [3] and better efficiency than MobileNetV2. MCUNet has advanced the Pareto optimal frontier in terms of accuracy vs.

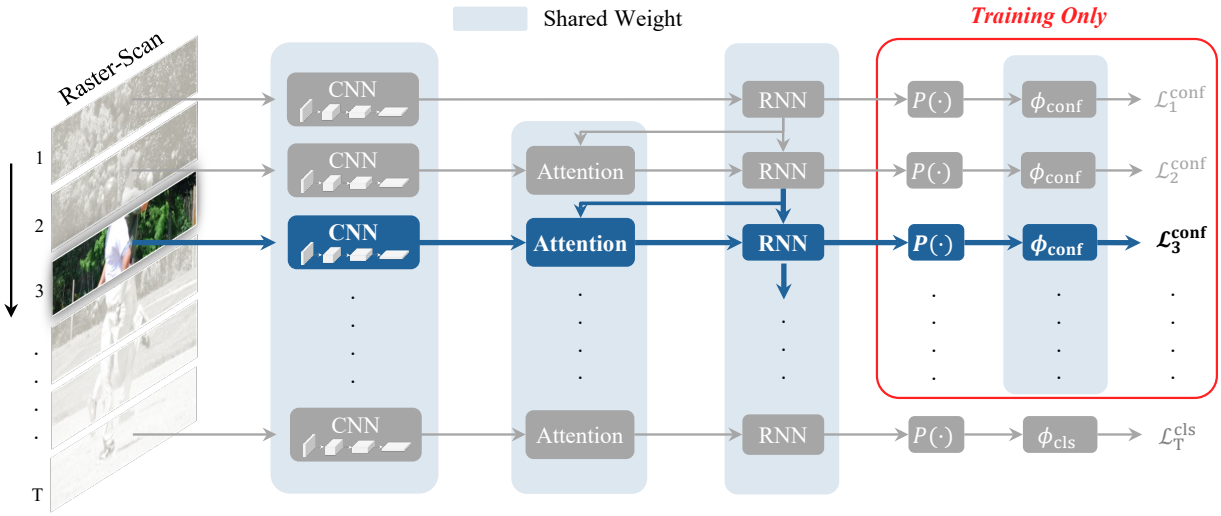

Figure 3. Illustration of the RaScaNet architecture. Inspired by the raster-scanning ISP in image sensors, our approach operates on T sub-images sequentially. RNN captures the visual context by scanning the image from top to bottom. A fully connected layer takes the last hidden state of the RNN to compute logits for classification. In addition, attention is adopted to enhance the quality of the features extracted from a CNN using the RNN hidden state as a query.

peak memory compared with the previous methods. However, it still suffers from the peak memory expansion in the first few layers of the network. Conventional CNNs based on full-frame inputs put inevitably hard limitation on their input sizes, and MCUNet is not an exception.

All existing methods for peak memory reduction do not optimize the model size because MCUs are provided with extra 1–2 MB of flash storage to store network models. However, minimizing the number of network parameters is also important in TinyML because loading the full model into on-chip SRAM reduces the read access time to the flash storage and the number of cache misses significantly.

Most prior works, except MCUNet [22], use the tricks proposed in MobileNetV2 [31] for calculating the peak memory of inverted residual blocks. However, such a trick is not desirable for MCUs because it increases cache misses and consequently degrades runtime performance. Without the trick, the peak memory of MobileNetV2 ($0.35\times$) increases by approximately three times (from 250 KB to 750 KB) in the first inverted residual block. The proposed method is also free from this limitation.

CIFAR10 [19] and ImageNet [8] datasets are frequently used for benchmarking the mobile-friendly models. However, both datasets are inappropriate for representing the microcontroller use-cases because of their small image resolution and an excessive number of classes [7]. Chowdhery *et al*. [7] have presented a new dataset, referred to as Visual Wake Words, which is parsed from COCO2014 [23] for serving a binary classification problem: whether a person is present in an image. The performances of several existing mobile-friendly models [16, 31, 32] have been reported to achieve 85%–90% accuracy under strict limitation of peak memory to fit in 250 KB as well as the restriction of multiply-accumulate operations (MACs) below 60M.

3. Method

The standard CNNs require more peak memory in the lower layers, and memory requirements gradually decrease as the feature passes to the upper layers. Peak memory at the initial stage of inference depends heavily on the input size, and feeding full-frame inputs to CNNs inevitably causes a bottleneck in TinyML models. RaScaNet aims to overcome this inherent limitation of CNNs by processing a group of horizontal lines at a time using a combination of CNN and RNN. A CNN extracts the features of each group sequentially which are then encoded using an RNN to learn the context from the entire image.

This section describes the architecture, loss function, and early termination scheme of the proposed model. The overall architecture of RaScaNet is illustrated in Fig. 3. We first present the CNN architecture for feature extraction from a set of rows. Next, we explain how to process the features using an RNN sequentially, from top to bottom, to learn the vertical context of the input image. Then, we introduce attention modules for boosting the discriminativeness of the features. Finally, we discuss the loss function and early termination scheme employed in our model.

3.1. CNN for Raster-Scanning Images

Let an input image tensor $\mathbf{X} \in \mathbb{R}^{3\times h\times w}$ be divided into a set of multiple rows as $\{\mathbf{x}_t\}_{t=1}^{T}$, where $\mathbf{x}_t \in \mathbb{R}^{3\times k\times w}$ denotes a group of rows and $T = \lceil \frac{h}{k} \rceil$ is the number of

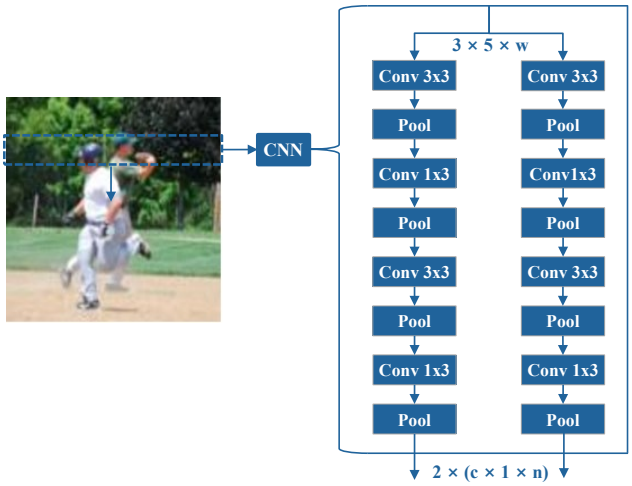

Figure 4. Detailed architecture of a CNN in RaScaNet (e.g., 2-head). For each input split into $3\times5\times w$, the network computes two stream features with c channels. The height dimension is reduced to 1 by convolutional layers without padding. Two features are then concatenated in the channel dimension and projected onto a latent space of the hidden state in the RNN.

groups. We read a sub-image $\mathbf{x}_t$ $(t = 1, \ldots, T)$ using a multi-head CNN, as illustrated in Fig. 4. Let the i^{th} head of the CNN be $\mathbf{P}_{i,t} \in \mathbb{R}^{c\times1\times n}$. Then, with a set of features from M heads, $\{\mathbf{P}_{i,t}|i = 1, \ldots, M\}$, we first concatenate and then project the resulting feature onto a latent space of the RNN hidden state to obtain $\mathbf{F}_t \in \mathbb{R}^{d\times1\times n}$, which is given by

$$\mathbf{F}_t = \text{conv}([\mathbf{P}_{1,t} : \mathbf{P}_{2,t} : \cdots : \mathbf{P}_{M,t}]; \Theta^f), \tag{1}$$

where : denotes the concatenation in a channel direction, and $\Theta^f \in \mathbb{R}^{(c\times M)\times d\times1}$ is the projection convolution weight.

The integration of a multi-head CNN facilitates learning more diverse representations than traditional single-head counterparts. In addition, using multiple heads based on shallow CNNs is advantageous for resource-constrained real-time applications compared with a single deep network because the parallelized implementation of multiple heads would improve processing speed; such a benefit is more prominent in on-sensor systems based on ASICs.

Our CNN architecture is based on the standard convolution operators instead of the depthwise separable convolutions [6], which show better trade-off between accuracy and computational cost. This is because 1) we are more interested in the main idea of RaScaNet rather than its architectural variations, and 2) we focus on the benefits of shallow (with four layers) CNNs. The use of the shallow model improves latency and reduces the model size by 5.3–12.9× compared with other tiny models.

3.2. Convolutional RNN

RaScaNet employs an RNN to capture the vertical scene context using the raster-scanned features from the CNN. We choose a variant of GRU [4] for the sequential modeling, which adopts 1D convolutions rather than fully connected layers. In the network, the weights for the update and forget gates are shared across pixels to facilitate learning universal rules. The use of the convolutional RNN enables RaScaNet to hold both temporal (vertical) and translational (horizontal) invariance, which are inherent properties of RNN and CNN, respectively.

We are given the projected feature $\mathbf{F}_t \in \mathbb{R}^{d\times1\times n}$ by Eq. (1), and the hidden state of the preceding RNN cell, $\mathbf{H}_{t-1} = [\mathbf{h}^1_{t-1}, \mathbf{h}^2_{t-1}, ..., \mathbf{h}^n_{t-1}] \in \mathbb{R}^{d\times1\times n}$, where d is the size of hidden state, and n is the width (in other words, the number of pixels). At the current timestep, the update of the hidden state in the convolutional RNN, $\mathbf{H}_t$, is given by

$$\mathbf{R}_t = \sigma(\text{conv}_r([\mathbf{F}_t : \mathbf{H}_{t-1}])), \tag{2}$$
$$\mathbf{U}_t = \sigma(\text{conv}_u([\mathbf{F}_t : \mathbf{H}_{t-1}])), \tag{3}$$
$$\bar{\mathbf{H}}_t = \tanh(\text{conv}_n([\mathbf{F}_t : \mathbf{H}_{t-1} \odot \mathbf{R}_t])), \tag{4}$$
$$\mathbf{H}_t = (1 - \mathbf{U}_t) \odot \bar{\mathbf{H}}_t + \mathbf{U}_t \odot \mathbf{H}_{t-1}, \tag{5}$$

where σ denotes a sigmoid function, $\odot$ denotes the Hadamart product, and $\mathbf{R}_t$ and $\mathbf{U}_t$ are the outputs of the forget gate and the update gate in GRU, respectively.

3.3. Attention

RaScaNet incorporates spatial and channel attention, using the preceding RNN cell as a query for boosting the discriminativeness of the CNN features. The integration of both attention modules effectively improves accuracy with marginal increase of memory usage and computational cost.

Spatial Attention The goal of the spatial attention module is to extract spatially distinct feature representations based on the similarity to the hidden state in the preceding RNN cell, $\mathbf{H}_{t-1}$. This approach assumes that the informative locations in the feature map at timestep $t-1$ are also likely to be so in the next scan-line because of the spatial smoothness property. We apply the spatial attention to each of multiple heads in the CNN.

Specifically, we first apply a 1D convolution to $\mathbf{P}_{i,t}$ and obtain $\mathbf{K}_{i,t} \in \mathbb{R}^{d\times1\times n}$ as follows:

$$\mathbf{K}_{i,t} = \text{conv}(\mathbf{P}_{i,t}; \Theta^k_i), \tag{6}$$

where $\Theta^k_i \in \mathbb{R}^{c\times d\times1}$ is a convolution weight parameter. Next, we compute the correlation between the spatial locations using a matrix multiplication and obtain the similarity matrix $\mathbf{S}_{i,t} \in \mathbb{R}^{n\times n}$, which is given by

$$\mathbf{S}_{i,t} = \mathbf{H}^\top_{t-1}\mathbf{K}_{i,t}. \tag{7}$$

Then, we apply a softmax function to $\mathbf{S}_{i,t}$ and obtain the attention map $\bar{\mathbf{A}}_{i,t} \in \mathbb{R}^{n\times n}$. The representation encoded with the spatial attention, denoted by $\bar{\mathbf{P}}_{i,t} \in \mathbb{R}^{c\times 1\times n}$, is given by

$$\mathbf{V}_{i,t} = \text{conv}(\mathbf{P}_{i,t}; \Theta_i^v), \tag{8}$$

$$\bar{\mathbf{P}}_{i,t} = \mathbf{P}_{i,t} + \mathbf{V}_{i,t}\bar{\mathbf{A}}_{i,t}, \tag{9}$$

where $\Theta_i^v \in \mathbb{R}^{c\times c\times 1}$ and $\mathbf{V}_{i,t} \in \mathbb{R}^{c\times 1\times n}$ are the corresponding convolution weight and the feature map given by another 1D convolution to $\mathbf{P}_{i,t}$, respectively.

Channel Attention The goal of the channel attention is to capture various discriminative information from our multi-head CNNs effectively during the top-down scanning process. The RNN hidden state from the preceding scan-line is used to guide which channel to focus on, in the current t^{th} scan-line. Instead of using the conventional channel attention mechanisms [17, 36], which employs the global pooling for squeezing spatial information, we apply a 1D convolution to learn the convolutional channel attention.

Given a set of features encoded with spatial attention, $\{\bar{\mathbf{P}}_{i,t}|i = 1,\dots,M\}$, we concatenate the feature tensors in a channel direction and obtain $\mathbf{C}_t \in \mathbb{R}^{(c\times M)\times 1\times n}$. Then, we further concatenate $\mathbf{C}_t$ and $\mathbf{H}_{t-1}$, which constructs $\tilde{\mathbf{C}}_t \in \mathbb{R}^{(c\times M+d)\times 1\times n}$. The channel attention weight $\tilde{\mathbf{A}}_t \in \mathbb{R}^{(c\times M)\times 1\times n}$ is derived from $\tilde{\mathbf{C}}_t$ by integrating typical squeeze-and-excitation convolutions, which is given by

$$\mathbf{C}_t = [\bar{\mathbf{P}}_{1,t} : \bar{\mathbf{P}}_{2,t} : \cdots : \bar{\mathbf{P}}_{M,t}], \tag{10}$$

$$\tilde{\mathbf{C}}_t = [\mathbf{C}_t : \mathbf{H}_{t-1}], \tag{11}$$

$$\tilde{\mathbf{A}}_t = \sigma(\text{conv}_e(\text{conv}_s(\tilde{\mathbf{C}}_t))). \tag{12}$$

By incorporating the channel attention weights via a 1D shared convolution, we assign a different attention weight on each spatial locations in $\tilde{\mathbf{A}}_t$. The final representation with the channel attention, $\tilde{\mathbf{F}}_t \in \mathbb{R}^{d\times 1\times n}$, is given by a Hadamard product of $\mathbf{C_t}$ and $\tilde{\mathbf{A}}_t$ followed by its projection onto the latent space of $\mathbf{H}_{t-1}$ as mentioned in Eq. (1).

$$\tilde{\mathbf{F}}_t = \text{conv}(\mathbf{C}_t \odot \tilde{\mathbf{A}}_t; \Theta^f). \tag{13}$$

3.4. Loss Function

RaScaNet maintains meaningful information, whereas less critical features are forgotten via the RNN in the middle of the top-down scanning process. When the raster-scanning reaches the bottom of the image, the last RNN hidden state $\mathbf{H}_T$ is employed to predict the class label as follows:

$$\hat{y}^{\text{cls}} = \phi_{\text{cls}}(P(\mathbf{H}_T)), \tag{14}$$

where ϕ_{cls} is the last fully connected layers for binary classification and $P(\cdot)$ is the global pooling operator.

Figure 5. Visualization of the confidence pseudo-label y^{conf} of class "person" for negative (top) and positive (bottom) images. The color bar at the right-hand side of each image denotes the target class presence score (pseudo-label) ranging from 0 (blue) to 1 (red).

Our loss function is composed of two terms: classification loss and confidence loss. The classification loss, $\mathcal{L}^{\text{cls}}$, is given by the cross-entropy between a predicted score vector and a one-hot ground-truth label. We introduce an additional classifier ϕ_{conf} to calculate the confidence loss $\mathcal{L}_t^{\text{conf}}$ in each scan-line, where ϕ_{conf} is applied to the hidden state of each RNN cell to predict whether the corresponding t^{th} scan-line contains the target or not. Note that ϕ_{conf} is used only for training and not in the inference stage.

In practice, the confidence label for each scan-line is easy to acquire if the detection bounding box annotation for each target object is available. However, without such a supervision, we provide an alternative approach: a pseudo-label for each scan-line, denoted by y_t^{conf}, based only on image-level class label. To obtain the pseudo-label, we apply classifier ϕ_{cls} to $\mathbf{H}_t$ in the t^{th} scan-line of a positive image, whereas the pseudo-label is set to 0 for all scan-lines in a negative image. Then, y_t^{conf} is stated as

$$y_t^{\text{conf}}(\mathbf{x}_t, \mathbf{H}_{t-1}) = \begin{cases} \phi_{\text{cls}}(P(\mathbf{H}_t)), & \text{if } y^{\text{cls}} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

where $\mathbf{x}_t$ is the input for the t^{th} scan-line and y^{cls} is the binary label of an input image $\mathbf{X}$. We expect that ϕ_{cls} estimates more reliable pseudo-labels as training progresses. Fig. 5 visualizes the estimated pseudo-label for six different examples.

The total loss $\mathcal{L}$ is defined as follows:

$$\mathcal{L} = \mathcal{L}^{\text{cls}} + \alpha \sum_{t=1}^{T} \mathcal{L}_t^{\text{conf}}, \tag{16}$$

where α is a coefficient balancing two terms. Note that $\mathcal{L}^{\text{cls}}$ is given by the cross-entropy loss between the ground-truth label and the image-level prediction of the network while $\mathcal{L}_t^{\text{conf}}$ is derived from the mean squared error between the

Figure 6. Visualization of the class activation map (CAM) of class "person" for negative (top) and positive (bottom) images in the VWW dataset. The activation of class "person" starts to be salient in the positive images when the uppermost parts of target are raster-scanned. High intensity is maintained until the bottom of the images since the RNN tends to summarize all the information in the past for classification at each scan-line.

pseudo-label and the predicted confidence score of the t^{th} scan-line.

3.5. Early Termination

In conventional CNNs, the whole image must be processed for image-level classification. On the other hand, RaScaNet can acquire the intermediate classification results in the middle of the scanning process before seeing an entire image. Fig. 5(bottom) presents that the confidence scores of positive images increase monotonically as the scan progresses. To take advantage of the unique property, RaScaNet can terminate the inference procedure if it gains a sufficient confidence score, greater than or equal to τ at an early stage as illustrated in Fig. 7(b). For implementation, we apply the final classifier ϕ_{cls} to each hidden state $\mathbf{H}_t$, which contains the information of the sub-images scanned until the t^{th} scan-line. The early termination scheme (RaScaNet-ET) reduces MACs by skipping the remaining parts of the input image, which is specified in Fig. 7(c).

(a) Score$_{t_a} < \tau$ (b) Score$_{t_b} = \tau$ (c) Score$_T > \tau$

Figure 7. Visualization of the early termination scenario with three timesteps, $t_a < t_b < T$. (a) If RaScaNet is not sure of target presence, the scan continues. (b) If RaScaNet is sufficiently confident about the presence, it stops the process to (c) avoid the unnecessary computation.

4. Experiments

We evaluate RaScaNet on the Visual Wake Words (VWW) [7] and Pascal VOC [11] datasets. To demonstrate the effectiveness of our approach, we compare our method with recent state-of-the-art tiny models [16, 22, 30, 31]. Moreover, following the design constraints of tiny vision models in the microcontroller use-case [7], we set a hard limit of 60M MACs per inference.

Implementation Details Unlike in conventional image classification tasks, we first split an image into groups of five-line images, which are then fed to RaScaNet sequentially. We adopt a batch-normalization and a ReLU layer after each convolutional layer in a multi-head CNN. In all experiments, the images for training and validation are resized to four difference scales, 210×240, 175×200, 140×160, and 105×120. We perform data augmentation using color-jittering, random cropping, and random horizontal flipping. Training is based on the AdamW [28] optimizer with warmup and a cosine learning rate scheduler. Moreover, we quantize both weight and activation in our network to INT8 [37, 39] while the classifier with size about 0.6KB uses floating point numbers.

4.1. Visual Wake Words

The VWW dataset focuses on the binary classification task: whether a person is present in an image. It is constructed by re-labeling the images in the publicly available COCO dataset [23]; each image assigned a label, 1 or 0, where 1 corresponds to a person. Note that VWW has 115K

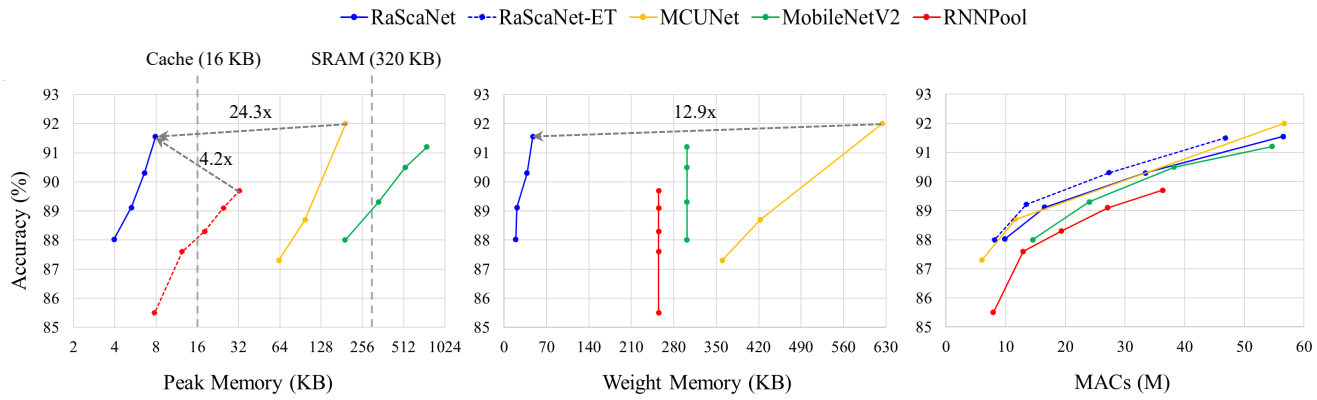

Figure 8. Comparison between RaScaNet and other state-of-the-art tiny models on the VWW dataset. The peak memory is plotted in a **log-scale**. RaScaNet requires 15.9–24.3× smaller peak memory (left) and 5.3–12.9× smaller weight memory (middle) compared with the state-of-the-art tiny model with a competitive accuracy. RaScaNet has less MACs than MobileNetV2 and RNNPool but comparable to MCUNet (right), which is partly because RaScaNet employs plain convolutions instead of depthwise separable convolutions. MACs of RaScaNet can be further reduced by adopting an early termination scheme (RaScaNet-ET). The peak memory for RNNPool [30] is calculated using the trick from MobileNetV2 [31]. The peak memory and weight memory of RaScaNet-ET is identical to RaScaNet.

training images, and 47% of them have the "person" label. The validation set contains 8K images in total.

Comparison with other models We compare RaScaNet to other state-of-the-art tiny models in terms of memory usage and accuracy (Fig. 8). RaScaNet requires 15.9–24.3× smaller peak memory with an accuracy comparable to MCUNet. The peak memory of RaScaNet is 4.2× smaller than RNNPool despite its higher accuracy by 1.9% points. Moreover, RaScaNet achieves 5.3–12.9× smaller model size compared with other methods. By reducing the model size dramatically, below 50 KB, RaScaNet can be fully loaded in the commercial on-chip SRAM.

Applying RaScaNet-ET, the early termination model introduced in Sec. 3.5, the MACs reduce by 17.1% on average for 210×240 images without accuracy drop compared with the basic RaScaNet model. RaScaNet-ET achieves state-of-the-art Pareto efficiency on accuracy vs. MACs.

Qualitative analysis The Fig. 6 illustrates the class activation maps (CAMs) of the RNN hidden state for some examples in the VWW dataset. The CAMs for class "person" ($\text{CAM}_{\text{person}}$) are typically plane in negative images (Fig. 6 top). In contrast, the $\text{CAM}_{\text{person}}$ in positive images identifies proper locations of the target objects (Fig. 6 bottom). The $\text{CAM}_{\text{person}}$ starts to be salient when the uppermost parts of the target are raster-scanned and maintain the high values until the bottom of the images. This property is different from the CAMs in the standard CNNs, where the highlighted areas focus only on the target. This is because RaScaNet carries the information of the target observed in the previous and current steps all the way to the last scan-line, where the final classification is performed. For negative images, some regions might be activated, however, they vanish in following scan-lines because they do not acquire strong belief to classify the inputs as positive.

4.2. Pascal VOC

The Pascal VOC [11] dataset is widely used for benchmarks in various computer vision tasks. The dataset is composed of 16K training images (2007-trainval, 2012-trainval) and 5K validation images (2007-test). The original dataset has 20 classes, but we relabeled it so that it only contain two classes: 1 ("person") or 0 ("not a person"). Following the setting in the VWW dataset, the image containing a person is labeled as 1 only when the bounding box for a person is larger than 0.5% of the image area.

RaScaNet and RaScaNet-ET significantly advance the Pareto optimal frontiers in all the three experiments compared with other tiny models (Fig. 9). RaScaNet requires 95.6× and 7.1× smaller peak and weight memory compared with MobileNetV2, respectively.

4.3. Ablation Study

Number of heads in CNN Adding CNN heads improves the accuracy at the cost of the model size and MACs. In this experiment, we aim to compare accuracy with similar number of parameters and MACs while changing number of heads. Hence, we reduce the channel size of CNN when increasing the head number. Table 1 summarizes the effect of multi-head CNN on accuracy. By using a 2-head CNN, we achieve 91.6% accuracy, which is 0.9% points higher than

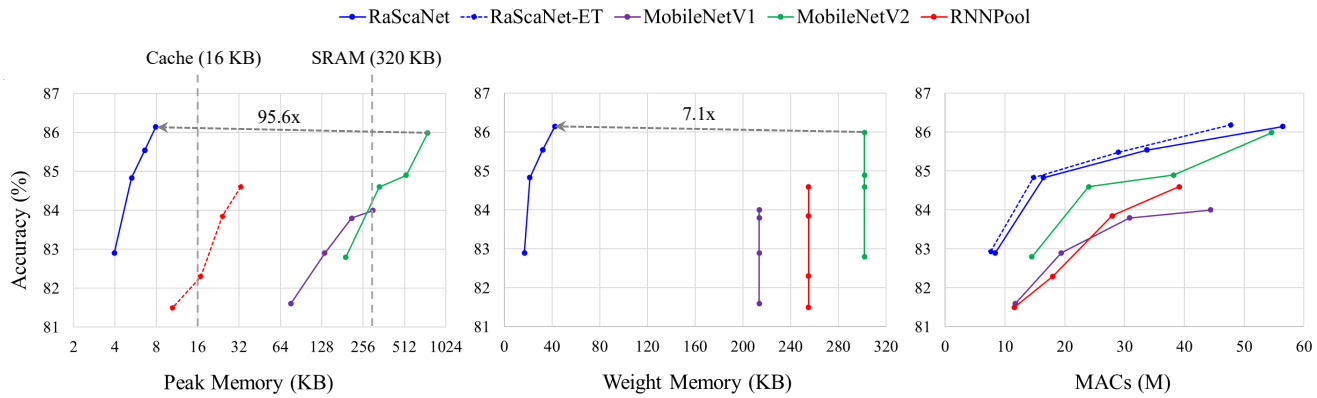

Figure 9. Comparison between RaScaNet and other tiny models on the Pascal VOC dataset. The peak memory is plotted in a **log-scale**. RaScaNet and RaScaNet-ET advance the Pareto optimal frontiers of accuracy vs. three evaluation criteria, i.e., peak memory, weight memory, and MACs. RNNPool [30] is reproduced using the code from [9]. The peak memory and weight memory of RaScaNet and RaScaNet-ET are same.

1-head counterpart. Increasing the number of heads to 3, while maintaining MACs below 60M leads to the saturation in accuracy.

Table 1. Impact of multi-head CNN of RaScaNet on the VWW dataset.

Method	MAC	Parameters	Accuracy
1-head CNN	56.2M	47.0KB	90.7%
2-head CNN	56.7M	47.6KB	91.6%
3-head CNN	57.5M	48.4KB	91.5%

Effect of confidence loss The loss function of RaScaNet consists of two terms: $\mathcal{L}^{\text{cls}}$ is derived from the entire image context, and $\mathcal{L}^{\text{conf}}$ is based on raster-scanned sub-images. Owing to the pseudo-label estimation per sub-image given by Eq. (15), the proposed method gains 0.8% points in accuracy compared with the network trained without $\mathcal{L}^{\text{conf}}$ as presented in Table 2. When the pseudo-label is given by a *trivial* way, where all raster-scanned sub-images have the same confidence labels with the image-level label, i.e., $y_t^{\text{conf}} = y^{\text{cls}}$, there is no performance gain by the use of the trivial pseudo-labels because it induces many false-positive confidence labels.

Table 2. Impact of confidence loss, $\mathcal{L}^{\text{conf}}$, given by the estimated pseudo-labels y^{conf} on the VWW dataset.

Method	Accuracy
$\mathcal{L}^{\text{cls}}$	90.8%
$\mathcal{L}^{\text{cls}} + \mathcal{L}^{\text{conf}}$ (trivial)	90.8%
$\mathcal{L}^{\text{cls}} + \mathcal{L}^{\text{conf}}$ (proposed)	91.6%

Attention modules RaScaNet incorporates two attention modules, which adopt the preceding RNN hidden state as a query. Table 3 shows the impact of attention modules; when the attention is applied, our model improves accuracy by 0.7% points. Although the integration of the attention modules increases MACs and model size by 11.1% points and 24.8% points, respectively, our network is still 5.3–12.9× smaller than other tiny models.

Table 3. Impact of attention module on the VWW dataset.

Method	MAC	Parameters	Accuracy
w/o attention	50.8M	37.8KB	90.9%
w/ attention	56.7M	47.6KB	91.6%

5. Conclusion

We propose a novel tiny architecture, referred to as RaScaNet, which raster-scans input images for running a deep neural network on ultra-low power systems. RaScaNet extracts features from the raster-scanned sub-images using a CNN and captures vertical context by applying an RNN sequentially from top to bottom. The proposed method incorporates multi-head CNN, attention mechanism, and confidence loss to improve accuracy. Also, we introduce early termination scheme to reduce MACs by preventing redundant computation.

RaScaNet achieves the state-of-the-art Pareto efficiency in accuracy vs. memory. It requires 15.9–24.3× less peak memory and 5.3–12.9× smaller weight memory than the state-of-the-art tiny models while maintaining the competitive accuracy. Furthermore, RaScaNet can operate on the current image sensor architecture, which has a common image signal processing algorithms running on the buffer that can hold only k rows of an image. We expect that the proposed raster-scan-based architecture extends to various on-sensor AI tasks, bringing deep learning closer to image sensors.

References

[1] TinyML summit 2021. https://www.tinyml.org/. 1

[2] Colby R Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, et al. Benchmarking TinyML systems: Challenges and direction. *arXiv preprint arXiv:2003.04821*, 2020. 1

[3] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 2

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 4

[5] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 2

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, pages 1251–1258, 2017. 2, 4

[7] Aakanksha Chowdhery, Pete Warden, Jonathon Shlens, Andrew Howard, and Rocky Rhodes. Visual Wake Words Dataset. *arXiv preprint arXiv:1906.05721*, 2019. 3, 6

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, pages 248–255. Ieee, 2009. 3

[9] Don Kurian Dennis, S Gopinath, C Gupta, A Kumar, A Kusupati, SG Patil, and HV Simhadri. EdgeML machine learning for resource-constrained edge devices. https://github.com/Microsoft/EdgeML, 2020. 8

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[11] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 6, 7

[12] Igor Fedorov, Ryan P Adams, Matthew Mattina, and Paul Whatmough. Sparse: Sparse architecture search for CNNs on resource-constrained microcontrollers. In *Advances in Neural Information Processing Systems*, pages 4977–4989, 2019. 1

[13] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision.*, pages 1440–1448, 2015. 1

[14] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision.*, pages 784–800, 2018. 2

[15] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobileNetV3. In *Proceedings of the IEEE International Conference on Computer Vision.*, pages 1314–1324, 2019. 2

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 3, 6

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, pages 7132–7141, 2018. 5

[18] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, pages 4350–4359, 2019. 2

[19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. 3

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1

[21] Tom Lawrence and Li Zhang. IoTNet: An efficient and accurate convolutional neural network for IoT devices. *Sensors*, 19(24):5541, 2019. 1

[22] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. MCUNet: Tiny deep learning on IoT devices. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2, 3, 6

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision.*, pages 740–755. Springer, 2014. 3, 6

[24] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision.*, pages 19–34, 2018. 2

[25] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 2

[26] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision.*, pages 2736–2744, 2017. 2

[27] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018. 2

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[29] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018. 2

[30] Oindrila Saha, Aditya Kusupati, Harsha Vardhan Simhadri, Manik Varma, and Prateek Jain. RNNPool: efficient non-linear pooling for RAM constrained inference. *arXiv preprint arXiv:2002.11921*, 2020. 1, 2, 6, 7, 8

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, pages 4510–4520, 2018. 2, 3, 6, 7

[32] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, pages 2820–2828, 2019. 2, 3

[33] Mingxing Tan and Quoc V Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 2

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 2

[35] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[36] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision.*, pages 3–19, 2018. 5

[37] Xilinx. Training-aware quantization in PyTorch. https://github.com/Xilinx/brevitas, January 2020. 6

[38] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. 1

[39] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 2, 6