[Supplementary Material]
RaScaNet: Learning Tiny Models by Raster-Scanning Images

1. Peak Memory

1.1. Peak Memory Calculation for RaScaNet

The peak memory is the maximum required memory to store activation maps during inference. The required memory of a layer is calculated by the sum of both input and output activation maps. For the l-th layer of L-layer in RaScaNet, let $I^{(l)} \in \mathbb{R}^{c_i^{(l)} \times h_i^{(l)} \times w_i^{(l)}}$ and $O^{(l)} \in \mathbb{R}^{c_o^{(l)} \times h_o^{(l)} \times w_o^{(l)}}$ be the input and output activation, respectively. Then, the required memory $M^{(l)}$ for the l-th layer is computed as follows:

$$M^{(l)} = (c_i^{(l)} \times h_i^{(l)} \times w_i^{(l)}) + (c_o^{(l)} \times h_o^{(l)} \times w_o^{(l)}). \tag{1}$$

And the peak memory is:

$$M_{\text{peak}} = \max_{1 \le l \le L} M^{(l)}. \tag{2}$$

Table 1 shows the dimension of weight and activation, along with the required memory for storing activation maps of each layer (Eq. 1). The peak memory usage of RaScaNet is 7.92 KB, which is 1-2 orders of magnitude smaller than that of other tiny models. The memory requirements decreased dramatically because of its unique property coming from raster-scanning.

Table 1. Required memory for each layer in RaScaNet with 210×240 input. The peak memory usage appears in the first layer, 'CNN-Conv2d-1' with 7.92 KB where the input height is 5 rather than 210 because of the raster-scanning approach. The dimensions of W, I, and O are $(c_i \times c_o \times h \times w)$, $(c_i \times h \times w)$, and $(c_o \times h \times w)$, respectively. W is the model weight.

Module	Layer	W	I	O	Memory (KB)
CNN	Conv2d-1	$3 \times 6 \times 3 \times 3$	$3 \times 5 \times 240$	$6 \times 3 \times 240$	**7.92**
	Pool-1	-	$6 \times 3 \times 240$	$6 \times 3 \times 120$	6.48
	Conv2d-2	$6 \times 11 \times 1 \times 3$	$6 \times 3 \times 120$	$11 \times 3 \times 120$	6.12
	Pool-2	-	$11 \times 3 \times 120$	$11 \times 3 \times 60$	5.94
	Conv2d-3	$11 \times 21 \times 3 \times 3$	$11 \times 3 \times 60$	$21 \times 1 \times 60$	3.24
	Pool-3	-	$21 \times 1 \times 60$	$21 \times 1 \times 30$	1.89
	Conv2d-4	$21 \times 42 \times 1 \times 3$	$21 \times 1 \times 30$	$42 \times 1 \times 30$	1.89
	Pool-4	-	$42 \times 1 \times 30$	$42 \times 1 \times 15$	1.89
Spatial Attention	Key-Conv1d	42×48	$42 \times 1 \times 15$	$48 \times 1 \times 15$	1.35
	Value-Conv1d	42×42	$42 \times 1 \times 15$	$42 \times 1 \times 15$	1.26
Channel Attention	Squeeze-Conv1d	$(42 + 48) \times 12$	$90 \times 1 \times 15$	$12 \times 1 \times 15$	1.53
	Excite-Conv1d	12×42	$12 \times 1 \times 15$	$42 \times 1 \times 15$	0.81
	Projection-Conv1d	42×48	$42 \times 1 \times 15$	$48 \times 1 \times 15$	1.35
RNN	Reset-Conv1d	$(48 + 48) \times 48$	$96 \times 1 \times 15$	$48 \times 1 \times 15$	2.16
	Update-Conv1d	$(48 + 48) \times 48$	$96 \times 1 \times 15$	$48 \times 1 \times 15$	2.16
	Candidate-Conv1d	$(48 + 48) \times 48$	$96 \times 1 \times 15$	$48 \times 1 \times 15$	2.16

1.2. Peak Memory Calculation Trick for MobileNetV2

As stated in [2], an inverted residual block is formulated as $\mathbb{F}(x) = (S \circ D \circ E)(x)$, where E is a pointwise expansion layer $E : \mathbb{R}^{i \times i \times k} \to \mathbb{R}^{i \times i \times n}$, D is a per-channel depthwise layer $D : \mathbb{R}^{i \times i \times n} \to \mathbb{R}^{o \times o \times n}$, and S is a pointwise squeeze

layer $S : \mathbb{R}^{o\times o\times n} \rightarrow \mathbb{R}^{o\times o\times k'}$. Because the function can operate on a per-channel basis where the final output is represented as an accumulation of it, $\mathbb{F}(x)$ can be formulated as follows:

$$\mathbb{F}(x) = \Sigma_{i=1}^{t}(S_i \circ D \circ E_i)(x), \tag{3}$$

where t represents the number of channel groups. By using such a t-way split trick, the required memory for computing $\mathbb{F}(x)$ can be as low as $|i^2k| + |o^2k'| + O(max(i^2, o^2))$. When $t = 6$, the peak memory usage of MobileNetV2 (0.35×) with 224×224 input, is reported as 250 KB [1], which comes from the first convolution layer. However, such a trick is known to hurt the runtime performance due to increased cache miss [2]. Therefore, instead of using a t-way split trick, we calculate the memory requirements of an inverted residual block using Eq. (1). Without the trick, the peak memory usage of MobileNetV2 (0.35×) increases to 752.64 KB (Table 2).

Table 2. The peak memory usage of MobileNetV2 (0.35×) without the trick [2] explodes in the second inverted residual block.

Second inverted residual block	W	I	O	Peak Memory (KB)
Pointwise-Expand	$8 \times 48 \times 1 \times 1$	$8 \times 112 \times 112$	$48 \times 112 \times 112$	702.46
Depthwise (stride: 2)	$48 \times 48 \times 3 \times 3$	$48 \times 112 \times 112$	$48 \times 56 \times 56$	**752.64**
Pointwise-Squeeze	$48 \times 8 \times 1 \times 1$	$48 \times 56 \times 56$	$8 \times 56 \times 56$	175.62

1.3. Effect of Input Resolution

We demonstrate how the peak memory usage changes according to various input sizes (Fig. 1). We first define n_{diag}=$\sqrt{h^2 + w^2}$. As n_{diag} gets larger, the peak memory usage of MobileNetV2 (0.35×) increases quadratically (Fig. 1 blue) as standard CNNs operate on the entire input (h×w). On the other hand, RaScaNet sequentially processes k number of rows at a time (k=5), and the peak memory usage is proportional to 5×w. Thus, the proposed method requires $\mathcal{O}(w) \approx \mathcal{O}(N)$ memory, whereas standard CNNs require $\mathcal{O}(hw) \approx \mathcal{O}(N^2)$ memory. When input size increases from QVGA (240×320) to VGA (480×640), the peak memory usage of MobileNetV2 (0.35×) increases by 4× (1152 KB → 4608 KB, Fig. 1 blue arrow), whereas the peak memory usage of RaScaNet increases by 2× (10.6 KB → 21.1 KB, Fig. 1 red arrow).

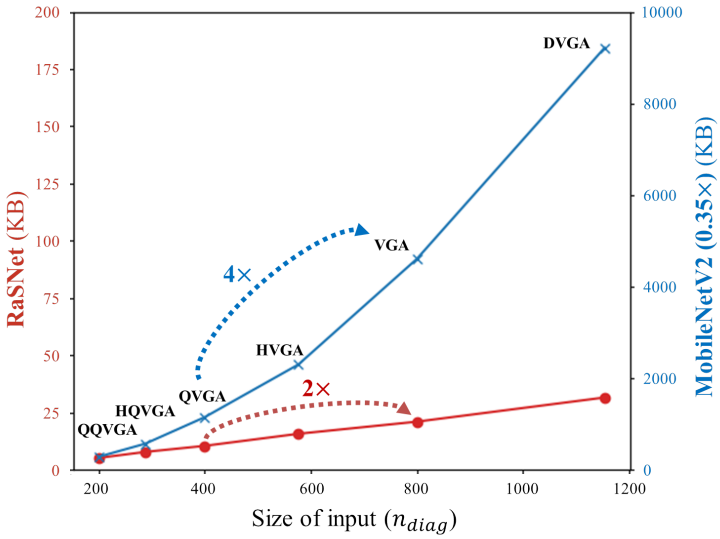

Figure 1. Peak memory usage vs. input resolution. As the input resolution gets larger, the peak memory usage of RaScaNet increases linearly (red), whereas the peak memory usage of MobileNetV2 (0.35×) increases quadratically (blue). Note that each y-axis has a different scale (50× larger scale for MobileNetV2).

2. Early Termination

2.1. Determination of Threshold τ

As described in our paper, we can early terminate the inference when a confidence score of the current scan-line is greater than the threshold τ. For the robustness of the scheme, we early terminate when a confidence score of two consecutive scan-line exceeds τ.

The early termination threshold τ affects on both accuracy and multiply-accumulate operations (MACs), but not on peak and weight memory. Comparing with the model, which does not use early termination, the MACs reduce more as we use lower τ at the cost of accuracy. In the experiment, τ is set to 0.99 to avoid the accuracy drop.

2.2. Analysis of Early Termination

In this section, we analyze the distribution of how early the inference stops using visual wake words dataset (VWW) [1]. For 210×240 input, RaScaNet sequentially processes 42 ($=\frac{210}{5}$) number of sub-images (steps). As shown in Fig. 2, 69.6% ($=\frac{2646}{3800}$) of the positive images (class "person") are early terminated and the average termination step ($\bar{N}_{\text{ET}}$) is 26.9. In other words, we can skip 15.1 (=42−26.9) steps and MACs reduce by 35.9% on average. Considering both positive and negative images, MACs reduce by 17.1% for 210×240 input.

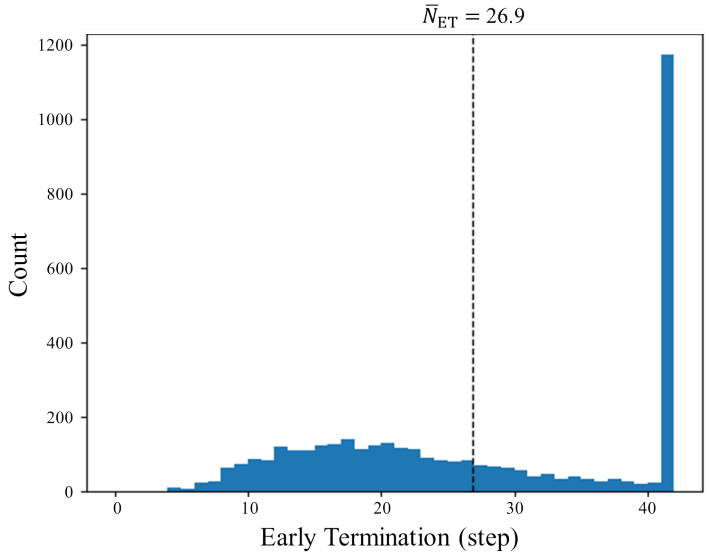

Figure 2. The distribution of early termination in positive images. The average termination step ($\bar{N}_{\text{ET}}$) is 26.9, and MACs reduce by 35.9%. The count at the last scan-line (step=42) is 1154 (30.4% of total positive images), which represents the number of samples not terminated early.

Fig. 3 shows early termination examples where inference stops in the middle of the scanning. RaScaNet can terminate even earlier when it gains strong confidence by looking at the distinctive feature. In most cases, the frontal face is a good indication of a person for early termination.