MobileDets: Searching for Object Detection Architecture for Mobile Accelerators (Supplementary Material)

Yunyang Xiong[1,2*], Hanxiao Liu[2*], Suyog Gupta[2], Berkin Akin[2], Gabriel Bender[2], Yongzhe Wang[2], Pieter-Jan Kindermans[2], Mingxing Tan[2], Vikas Singh[1], and Bo Chen[2]

[1]University of Wisconsin-Madison
[2]Google

1 Summary

This is the supplementary material for our submission. In this document, we describe further details of the connections with Tucker/CP decomposition, and experimental results.

1.1 Connections with Tucker/CP decomposition

The two proposed layer varients can be linked to Tucker/CP decomposition. Fig. 1 shows the graphical structure of an inverted bottleneck with input expansion ratio s, modulo nonlinearities. This structure is equivalent to the sequential structure of approximate evaluation of a regular convolution by using CP decomposition [4]. The Tucker convolution layer with input and output compression ratios s and e, denoted as Tucker layer shown in Fig. 3, has the same structure (modulo nonlinearities) as the Tucker decomposition approximation of a regular convolution [3]. Fused inverted bottleneck layer with an input expansion ratio s, shown in Fig. 2, can also be considered as a variant of the Tucker decomposition approximation.

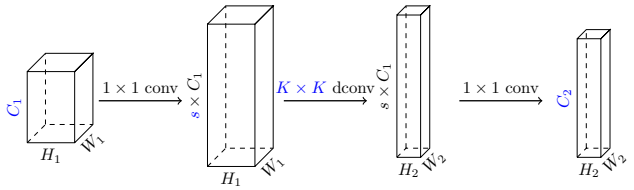

Figure 1: Inverted bottleneck layer: 1×1 pointwise convolution transforms the input channels from C_1 to $s \times C_1$ with input expansion ratio $s > 1$, then $K \times K$ *depthwise* convolution transforms the input channels from $s \times C_1$ to $s \times C_1$, and the last 1×1 pointwise convolution transforms the channels from $s \times C_1$ to C_2. The highlighted C_1, s, K, C_2 in IBN layer are searchable.

1.2 Experimental Setup

Architecture Search. Our proposed search spaces are complementary to any neural architecture search algorithms. We employ TuNAS [1] for its scalability and its reliable improvement over random baselines. To avoid overfitting the true validation dataset, we split out 10% of the COCO training data to evaluate the models and compute rewards during search. Hyperparameters for training the shared weights follow those in standalone training. As for reinforcement learning, we use Adam optimizer with an initial learning rate of 5×10^{-3}, $\beta = (0, 0.999)$ and $\epsilon = 10^{-8}$. We search for 50K steps to

*Equal contribution.

4th Workshop on Meta-Learning at NeurIPS 2020, Vancouver, Canada.

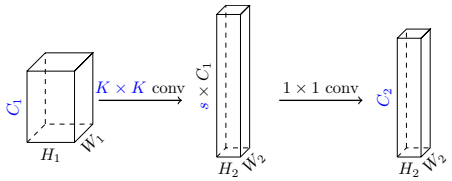

Figure 2: Fused inverted bottleneck layer: $K \times K$ *regular* convolution transforms the input channels from C_1 to $s \times C_1$ with input expansion ratio $s > 1$, and the last 1×1 pointwise convolution transforms the channels from $s \times C_1$ to C_2. The highlighted C_1, K, s, C_2 in the fused inverted bottleneck layer are searchable.

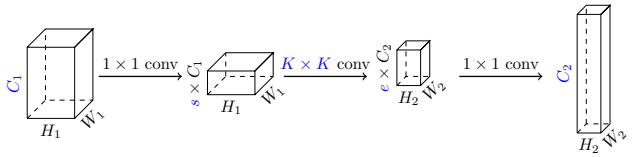

Figure 3: Tucker layer: 1×1 pointwise convolution transforms the input channels C_1 to $s \times C_1$ with input compression ratio $s < 1$, then $K \times K$ *regular* convolution transforms the input channels from $s \times C_1$ to $e \times C_2$ with output compression ratio $e < 1$, and the last 1×1 pointwise convolution transforms the channels from $e \times C_2$ to C_2. The highlighted C_1, s, K, e, C_2 in Tucker layer are searchable.

obtain the architectures in ablation studies and search for 100K steps to obtain the best candidates in the main results table.

Architecture Evaluation via Retraining. The training is carried out over 32 synchronized replicas on a 4x4 TPU-v2 pod. For fair comparison with existing models, we use standard preprocessing in Tensorflow object detection API without additional enhancements such as drop-block or auto-augment. We use SGD with momentum 0.9 and weight decay 5×10^{-4}. The learning rate is warmed up in the first 2000 steps and then follows cosine decay. The training setting is the same between our searched model and baselines for fair comparison and all models are trained from scratch without any ImageNet pre-trained checkpoint. We consider two different training schedules: (a) *Short-schedule*: Each model is trained for 50K steps with a batch size of 1024 and an initial learning rate of 4.0; (b) *Long-schedule*: Each model is trained for 400K steps with a batch size of 512 and an initial learning rate of 0.8. The short schedule is about $4\times$ as fast as the long schedule but would result in slightly inferior quality. Unless otherwise specified, we use the short schedule for ablation studies and the long schedule for the final results.

Latency Benchmarking. The simulated latencies are obtained using lookup tables similar to those used by NetAdapt [5]. We report on-device latencies for all of our main results. We benchmark using TF-Lite for CPU, EdgeTPU and DSP, relying on NNAPI to delegate computations to accelerators. All benchmarks use single-thread and a batch size of 1. In Pixel 1 CPU, we use only a single large core. For Pixel 4 EdgeTPU and DSP, the models are fake-quantized [2] as required. The GPU models are optimized and benchmarked using TensorRT 7.1 converted from an intermediate ONNX format.

1.3 Transferability of Models across Hardware

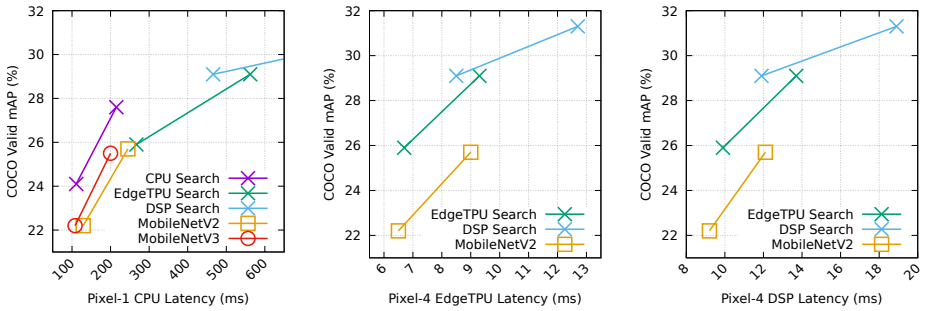

Figure 4: Transferability of architectures (searched wrt different target platforms) across hardware platforms. For each architecture, we report both the original model and its scaled version with channel multiplier $1.5\times$.

Finally, we investigate the transferability of the architectures across hardware platforms. Fig. 4 compares MobileDets (obtaind by targeting at different accelerators) wrt different hardware platforms. Our results indicate that architectures searched on EdgeTPUs and DSPs are mutually transferable. In fact, both searched architectures extensively leveraged regular convolutions. On the other hand, architectures specialized wrt EdgeTPUs or DSPs (which tend to be FLOPs-intensive) do not transfer well to mobile CPUs.

1.4 Architecture Visualizations

Fig. 5 visualizes our searched object detection architectures, MobileDets, by targeting at CPU, EdgeTPU, and DSP, using our TDB search space. We observe that MobileDets use regular convolutions extensively on EdgeTPU and DSP, especially in the early stage of the network where depthwise convolutions tend to be less efficient. These results demonstrate that IBN-only search space is not optimal for these mobile accelerators.

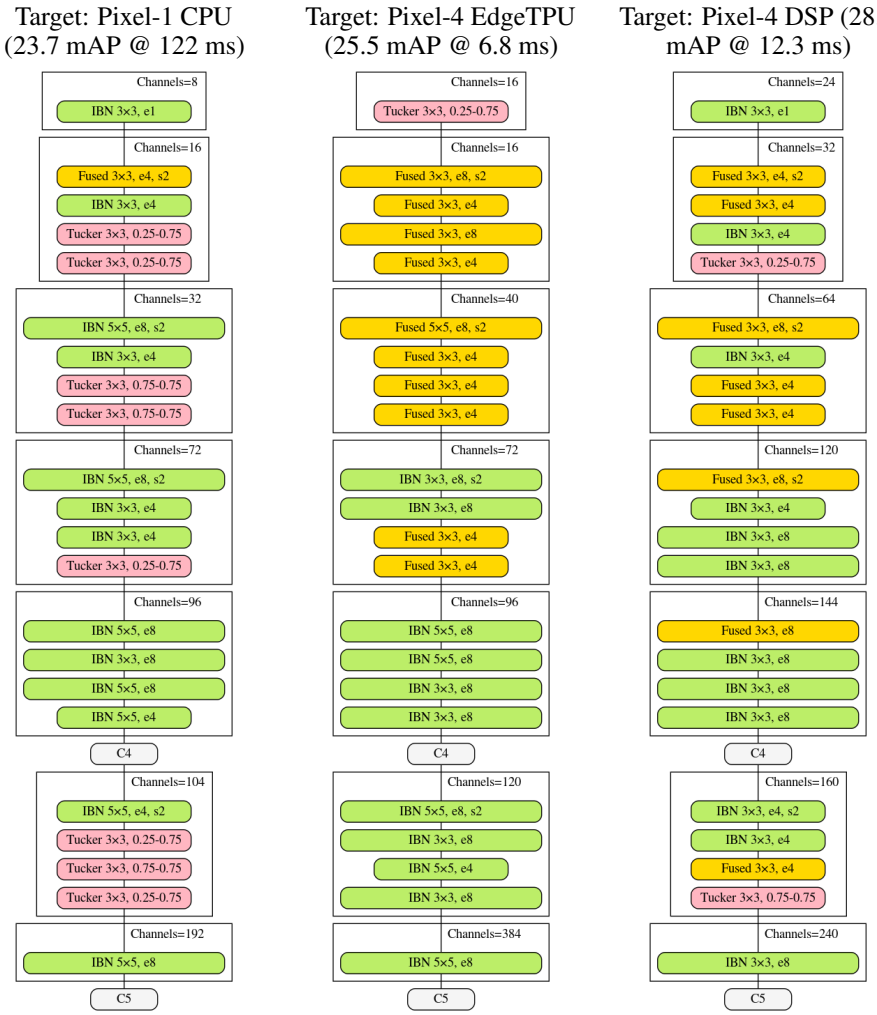

Figure 5: Best architectures searched in the IBN+Fused+Tucker space wrt different mobile accelerators. Endpoints C4 and C5 are consumed by the SSD head.

References

[1] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V Le. Can weight sharing outperform random architecture search? an investigation with tunas. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14323–14332, 2020.

[2] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[3] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

[4] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

[5] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision*, pages 285–300, 2018.