A Appendix

A.1 Connections with Tucker/CP decomposition

The two proposed layer varients can be linked to Tucker/CP decomposition. Fig. 2 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 [12]. The Tucker convolution layer with input and output compression ratios s and e, denoted as Tucker layer shown in Fig. 4, has the same structure (modulo nonlinearities) as the Tucker decomposition approximation of a regular convolution [11]. Fused inverted bottleneck layer with an input expansion ratio s, shown in Fig. 3, can also be considered as a variant of the Tucker decomposition approximation.

A.2 Experimental Setup

Architecture Search. Our proposed search spaces are complementary to any neural architecture search algorithms. We employ TuNAS [2] 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 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 in Table 1.

Latency Benchmarking. The simulated latencies in Section 3.1 are obtained using lookup tables similar to those used by NetAdapt [21]. 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 [10] as required. The GPU models are optimized and benchmarked using TensorRT 7.1 converted from an intermediate ONNX format.

A.3 Transferability of Models across Hardware

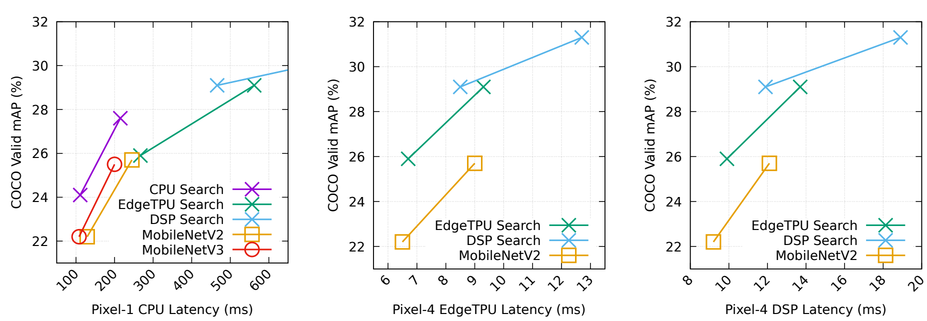

Figure 8: Transferability of architectures (searched wrt different target platforms) across hardware platforms. For each architecture, we report both the original model and its scaled version with channel multiplier $1.5\times$.

Finally, we investigate the transferability of the architectures across hardware platforms. Fig. 8 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.

A.4 Architecture Visualizations

Fig. 9 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.

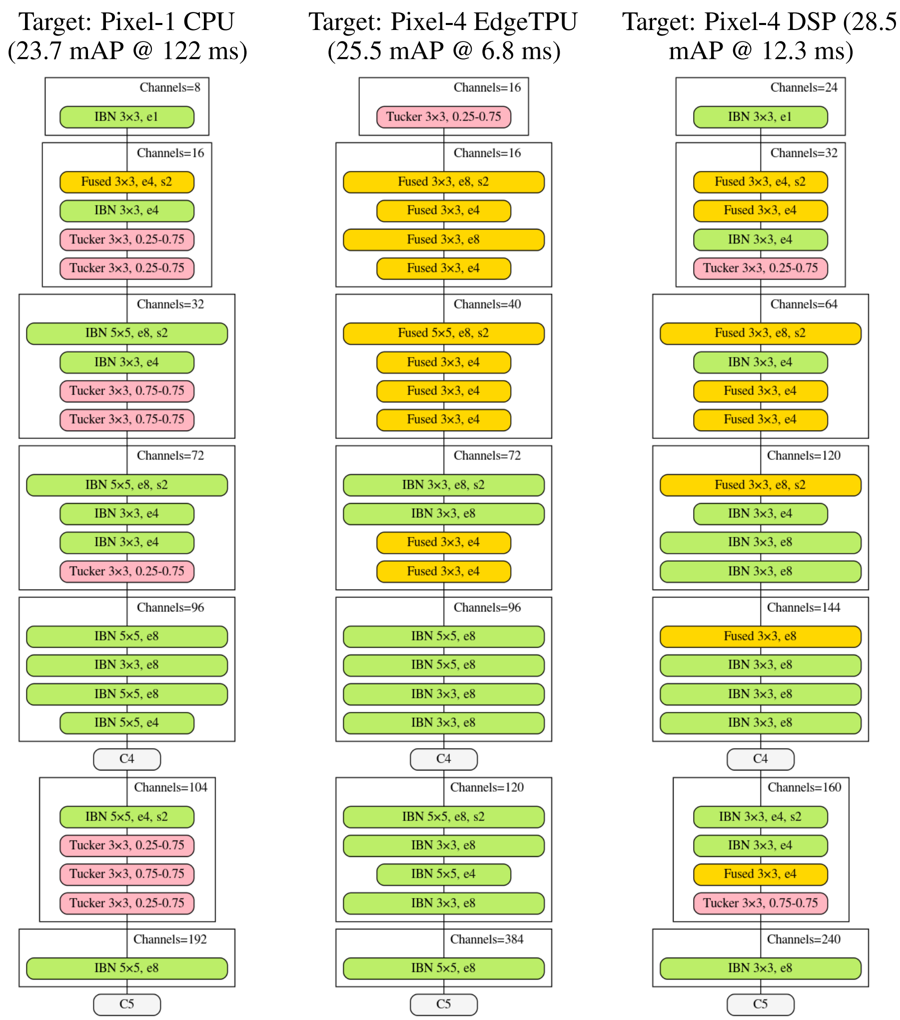

Figure 9: Best architectures searched in the IBN+Fused+Tucker space wrt different mobile accelerators. Endpoints C4 and C5 are consumed by the SSD head.

B Previous Reviews

B.1 Updates and Resubmission Rationale

A key contribution of our submission is the development of new NAS search spaces and models that specifically target hardware accelerators for mobile phones, such as DSPs, EdgeTPUs and edge GPUs. In contrast, most previous NAS research either (i) explicitly optimized for FLOPS or CPU latencies, or (ii) reused search space primitives such as inverted bottlenecks which were originally developed for CPUs. In this work we quantitatively reveal that this limited search space design can be a substantial bottleneck of the performance of NAS for modern mobile accelerators.

However, we believe the above was not communicated well enough in a previous version of our manuscript. For example, R1 from our ECCV submission raised concerns that our models had high FLOPS counts, while R3 and R4 expressed concerns that the paper showed limited performance improvements on CPU.

In addition, the ECCV reviewers were concerned about the novelty of the NAS method (TuNAS) and/or task (object detection), which was the main reason for rejection. These, however, were not the key points which we intended to focus on. In this 6-page submission, we have substantially improved the clarity of the paper to better reflect our main contributions:

New search space family. We propose a new search space family which can substantially boost the performance of NAS methods across a variety of modern mobile accelerators. This is achieved by revisiting full (instead of depthwise) convolutions, which are underexplored in existing mobile NAS works. State-of-the-art models. We deliver a set of lightweight, easy-to-deploy mobile object detection with state-of-the-art quality-latency trade-offs across EdgeTPUs, DSPs, as well as edge GPUs (new results). Models will be released to benefit a wide range of on-device object detection applications.

Besides improved writing, we also provide new results to demonstrate that our search space design is not only performant for CPUs, EdgeTPUs and DSPs, but can also generalize well to a new hardware platform (Nvidia Jetson GPU).

B.2 Original Reviews

B.2.1 Meta Reviewer

Comment: After taking the rebuttal into account and reading each others reviews, there is a consensus among reviewers that while the paper is focused on an important topic and some of the reviewers comments were addressed in the rebuttal, the main limitations remain and the novelty and contribution is not sufficient to warrant acceptance at ECCV 2020.

B.2.2 Reviewer #1

Summary Of Contributions: This paper proposes a NAS method that generates a latency-efficient backbone for object detection. The authors aim to multiple target platforms to reduce the inference speed of a backbone by searching with Inverted Bottleneck, Bottleneck, and their fused block. Experiments are done to support the authors' claim, Strengths: +) The proposed idea of fusing full-convolution into IB is interesting. +) Restricting search space looks good. Weaknesses: -) The novelty is limited. Since the proposed method focuses on finding the backbone for the following detection module SSDLite, there has been a lot of NAS methods that successfully find a classification backbone such as ProxylessNAS targetted to diverse platforms. -) No comparison with other backbones + SSDLite. For example, MNasNet + SSD-lite (not using NAS-FPN head) can be easily evaluated in Table 1. -) Limited performance improvement (especially in CPU) -) The architecture found by the proposed method has large FLOPs and parameters compared to the baselines. -) The authors constrained the search spaces but the method has still a costly training budget of using 4x4 TPU-v2.

Suggestion To Authors: 1. Why the searched models targetted to DSP show better performance than that of EdgeTPU in the transferability test in Fig 9 (b)? Readers may speculate the searched models would not be close to optimal. 2. A shallow search space could be a benefit to quickly get a model but sometimes looks too restricted. Is there any intuition why the searching space is restricted to block-level? TuNAS looks available to run with operation-level search space. 3. How many TPU (or GPU) times searching for architecture? 4. Please specify the latency of MobileNetV3 in EdgeTPU and DSP in Table 1. 5. Does the search space include the strides of a convolution layer? 6. SSD-lite mainly uses depthwise convolutions which are not efficient in some platforms such as GPU. Therefore, it is natural to extend the proposed method by considering SSD-lite as well into the search space.

Preliminary Rating: 3: Borderline reject Preliminary Rating Justification: Searching for a network architecture for SSD-lite detector may be necessary for a fast object detection task. However, the novelty is limited compared to the previous NAS methods, because the method just finds a backbone and use SSD-lite as a detection head. Furthermore, I don't think this method can be easily adopted because the training time is too costly.

B.2.3 Reviewer #4

Summary Of Contributions: The paper proposes to search a lightweight detection network for mobile accelerators by using regular convolutions, instead of just following the inverted bottlenecks (IBN) in MobileNet v3. All the modifications focus on replacing the depthwise convolution with regular convolution, and have achieved certain results. Strengths: The motivation is intuitive, and it has important practical value to search lightweight detection network for Mobile devices. The experiments shows the effectiveness on some specific devices. Weaknesses: 1. Insufficient contributions. The authors should refine and reorganize the contributions, especially in the section of introduction. Moreover, the depthwise convolution will increase the Memory Access Cost (MAC), thus it does not necessarily reduce the inference time. This idea has be demonstrated in ShuffleNet v2 [1], so using regular convolution to replace the depthwise convolution is not that significant. 2. In Fig. 5, the proposed search space (i.e., IBN+Fused and IBN+Fused+Tucker) has no advantages over IBN both in speed and accuracy, the authors should better clarify this. 3. There are no comparisons with the original MnasFPN. 4. The manuscript should be refined carefully, there are many typos and poor sentences.

[1] Ma N, Zhang X, Zheng H T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design, ECCV2018: 116-131.

Suggestion To Authors: Some typos and poor expressions: Line 34: a lot of effort.... Line 113: most performant models... Line 248: modulo non-linearities Line 251: modulo nonlinearities

Preliminary Rating: 3: Borderline reject Preliminary Rating Justification: Although the paper has achieved certain progresses, the contributions are insufficient, please see the weaknesses. The paper should be reorganized to highlight the innovations and enhance the readability.

Confidence: 4: High, published similar work Final Rating: 3: Borderline reject Final Rating Justification: The authors provide some explanations for my previous concerns. However, I think the main weaknesses still exist, I thus keep my rating unchanged.

B.2.4 Reviewer #3

Summary Of Contributions: (1) This paper searches network backbones targeted for object detection on some specific harewares. MobileDets, the searched a family of models, achieve surprising results on different hardware platforms.

(2) A new search space — TDB is proposed by adding two new kind of cell structures (regular convolution based) into IBNs search space. Strengths: (1) The searched models — MobileDets show surprising results on different hareware platforms.

For COCO, MobileDets outperform MobileNetV3+SSDLite by 1.7 mAP at comparable mobile CPU inference latencies. MobileDets also outperform MobileNetV2+SSDLite by 1.9 mAP on mobile CPUs, 3.7 mAP on EdgeTPUs and 3.4 mAP on DSPs while running equally fast. Moreover, MobileDets are comparable with the state-of-the-art MnasFPN on mobile CPUs even without using the feature pyramid, and achieve better mAP scores on both EdgeTPUs and DSPs with up to 2x speedup. Weaknesses: (1) The novelty is limited.

I. Searching for architectures targeted at specific platforms has already been proposed many times, like [4, 30].

II. The proposed TDB search space is obtained by only adding ResNet-like cell structure and MLPConv(Network In Network)-like cell structure. The connections between ResNet cell structure, Inverted bottleneck sturcture and Tucker/CP decomposition have already been demonstrated by previous works shown below.

Min Lin et al. "Network In Network". ICLR 2014. Chen Yunpeng et al. "Sharing Residual Units Through Collective Tensor Factorization in Deep Neural Networks". IJCAI-18. Marcella Astrid et al. "CP-decomposition with Tensor Power Method for Convolutional Neural Networks Compression". BigComp 2017.

III. The adopted NAS method has no differences between previous NAS method (TuNAS).

(2) What most interested me is the searched model. However, I mainly concerns whether the experimental setting is fair between the searched models and the baselines (MobileNetV2, MobileNetV3).

Suggestion To Authors: (1) Show the key novelty of the proposed method.

(2) Make sure the experimental setting is fair between the searched model and baselines (MobileNetV2, MobileNetV3), e.g. the training time. The training time has great effects on two-stage object detection frameworks, like Mask-RCNN. Preliminary Rating: 3: Borderline reject Preliminary Rating Justification: The paper searches architectures for object detection on different platforms. The searched architectures — MobileDets show surprising results.

However, the novelty is limited. Platform aware NAS has been explored by many previous papers.

The proposed TDB search space is obtained by only adding ResNet-like cell structure and MLPConv-like cell structure. And the connections between ResNet cell structure, inverted bottleneck structure and tensor decompostions have already been demonstrated by previous works. Confidence: 4: High, published similar work Final Rating: 3: Borderline reject Final Rating Justification: I read the other reviews and the author responses. I still hold that the novellty is limited. This paper only apply the NAS to search backbone for detection. I will never change my opinion.