Single-Path NAS: Device-Aware Efficient ConvNet Design

Dimitrios Stamoulis [1 *] **Ruizhou Ding** [1] **Di Wang** [2] **Dimitrios Lymberopoulos** [2] **Bodhi Priyantha** [2] **Jie Liu** [3] **Diana Marculescu** [1]

Abstract

Can we automatically design a Convolutional Network (ConvNet) with the highest image classification accuracy under the latency constraint of a mobile device? Neural Architecture Search (NAS) for ConvNet design is a challenging problem due to the combinatorially large design space and search time (at least 200 GPU-hours). To alleviate this complexity, we propose *Single-Path NAS*, a novel differentiable NAS method for designing device-efficient ConvNets in **less than 4 hours**. 1. **Novel NAS formulation**: our method introduces a **single-path**, over-parameterized ConvNet to encode all architectural decisions with shared convolutional kernel parameters. 2. **NAS efficiency**: Our method decreases the NAS search cost down to **8 epochs** (30 TPU-hours), *i.e.*, up to **5,000**$\times$ **faster** compared to prior work. 3. **On-device image classification**: *Single-Path NAS* achieves 74.96% top-1 accuracy on ImageNet with 79ms inference latency on a Pixel 1 phone, which is state-of-the-art accuracy compared to NAS methods with similar latency ($\leq 80ms$).

1. Introduction

"*Is it possible to reduce the NAS search cost down to only few hours?*" NAS methods have revolutionized the design of ConvNets (Zoph et al., 2017), yielding state-of-the-art results in deep learning applications (Real et al., 2018). NAS has a profound impact on the design of hardware-efficient ConvNets for on-device computer vision, *e.g.*, under inference latency constraints on a mobile device (Tan et al., 2018). However, NAS remains an intrinsically costly problem with a combinatorially large search space: *e.g.*, searching for a ConvNet with 22 layers and five candidate operations per layer yields $5^{22} \approx 10^{15}$ possible networks.

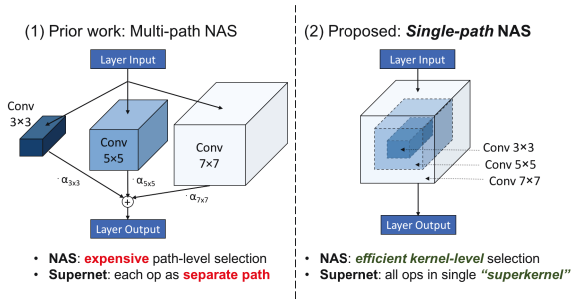

Figure 1. Single-Path NAS directly optimizes for the subset of convolution weights of an over-parameterized "**superkernel**" in each ConvNet layer (right). Our **novel view** of the design space eliminates the need for maintaining separate paths for each candidate operation, as in previous *multi-path* approaches (left).

Inefficiencies of *multi-path* NAS: Recent methods use one-shot formulations (Liu et al., 2018; Pham et al., 2018) by viewing the NAS problem as an operation/path selection problem: first, an over-parameterized, *multi-path* supernet is constructed, where, for each layer, every candidate operation is added as a *separate* trainable path (Figure 1, left). Next, NAS searches for the paths of the *multi-path* supernet that yield the optimal architecture. As expected, naively branching out all paths is inefficient, since the number of trainable parameters during the search grows linearly with respect to the number of candidate operations per layer (Bender et al., 2018). To tame the memory explosion due to the *multi-path* supernet, current methods employ "workaround" solutions: *e.g.*, searching on a proxy dataset (Wu et al., 2018), or employing a memory-wise scheme where only few paths are updated during search (Cai et al., 2019). Nevertheless, these methods remain considerably costly, with total computational demand of at least 200 GPU-hours.

In this paper, we propose *Single-Path NAS*, a novel NAS method for designing hardware-efficient ConvNets in **less than 4 hours**. Our **key insight** is illustrated in Figure 1 (right). Our key observation is that different candidate convolutional operations can be viewed as subsets of the weights of an over-parameterized **single "superkernel"**. Instead of choosing from different paths/operations as in *multi-*

*Extended abstract of ODML-CDNNR 2019 presentation (required non-archival arxiv.org version). Full paper can be found in (Stamoulis et al., 2019). [1]Department of ECE, Carnegie Mellon University, Pittsburgh, PA, USA [2]Microsoft, Redmond, WA, USA [3]Harbin Institute of Technology, Harbin, China. Correspondence to: Dimitrios Stamoulis <dstamoul@andrew.cmu.edu>.

Joint Workshop on *On-Device Machine Learning & Compact Deep Neural Network Representations* (ODML-CDNNR 2019). ICML 2019 Workshop.

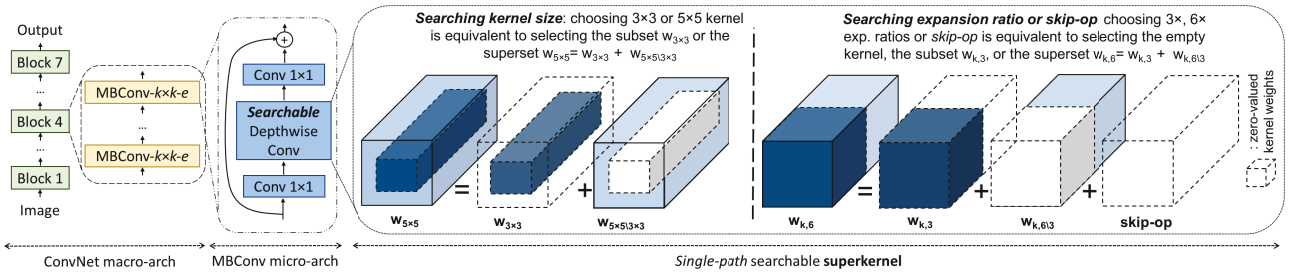

Figure 2. Single-path NAS builds upon *hierarchical* MobileNetV2-like search spaces (Tan et al., 2018) to identify the mobile inverted bottleneck convolution (MBConv) per layer (left). Our *one-shot supernet* encapsulates all possible NAS architectures in the search space, *i.e.*, different kernel size (middle) and expansion ratio (right) values, without the need for appending each candidate operation as a separate path. *Single-Path* NAS directly searches over the weights of the per-layer **searchable "superkernel"** that encodes all MBConv types.

path methods, we view the NAS problem as *finding which subset of the per-layer **"superkernel"** weights to use*, hence searching across a **single-path** one-shot NAS supernet.

2. Proposed Method: *Single-Path* NAS

Search space: We build upon the MobileNetV2-based macro-architecture (Figure 2, left), where layers are grouped into blocks based on their filter sizes (Sandler et al., 2018). Each layer is a mobile inverted bottleneck convolution (conv) MBConv-$k \times k$-e (Sandler et al., 2018), (*i.e.*, a point-wise 1×1 conv with expansion ratio e, a $k \times k$ depthwise conv, and a linear 1×1 conv; Figure 2, left). Our goal is to identify the MBConv-$k \times k$-e type per layer.

"Superkernel"-based formulation: Our key insight is that the candidate operations can be viewed as subsets of the **"superkernel"** weights. Without loss of generality, we denote the weights of two candidate operations, *e.g.*, 3×3 conv or 5×5 conv, as $\mathbf{w}_{3\times3}$ and $\mathbf{w}_{5\times5}$, respectively. We observe that $\mathbf{w}_{3\times3}$ can be viewed as the *inner* core of the $\mathbf{w}_{5\times5}$ kernel, while "zeroing" out the "*outer*" shell $\mathbf{w}_{5\times5\setminus3\times3}$ (Figure 2, middle). Thus, we can write the NAS decision as:

$$\mathbf{w}_k = \mathbf{w}_{3\times3} + \mathbb{1}(\left\|\mathbf{w}_{5\times5\setminus3\times3}\right\|^2 > t_{k=5}) \cdot \mathbf{w}_{5\times5\setminus3\times3} \quad (1)$$

where $\mathbb{1}(\cdot)$ is the indicator function that encodes the architectural NAS choice and $t_{k=5}$ is a latent variable (*e.g.*, a threshold value) that controls the decision.

Trainable NAS decisions: Drawing inspiration from quantization decisions (Ding et al., 2019), we use the *group Lasso term* in the $\mathbb{1}(\cdot)$ condition. Instead of picking the thresholds (*e.g.*, $t_{k=5}$) by hand, we seamlessly treat them as trainable parameters to learn via gradient descent. To compute the gradients for thresholds, we relax the indicator function $g(x,t) = \mathbb{1}(x > t)$ to a sigmoid function $\sigma(\cdot)$, when computing gradients, *i.e.*, $\hat{g}(x,t) = \sigma(x > t)$.

Searching for expansion ratio or skip-op: Since the kernel-based $\mathbf{w}_k$ result (Equation 1) is a kernel itself, we can in turn apply our formulation to encode expansion ratio decisions, where $e = 3$ or $e = 6$ correspond to using half or all the channels of an MBConv-$k \times k$-6 layer, respectively (Figure 2, right). Finally, by "zeroing" out all channels, we encode the NAS decision of dropping the entire layer:

$$\begin{aligned}\mathbf{w} = \mathbb{1}(\|\mathbf{w}_{k,3}\|^2 > t_{e=3}) \cdot (\mathbf{w}_{k,3} + \\ \mathbb{1}(\left\|\mathbf{w}_{k,6\setminus3}\right\|^2 > t_{e=6}) \cdot \mathbf{w}_{k,6\setminus3})\end{aligned} \quad (2)$$

Hence, our **searchable superkernel** can sufficiently capture any MBConv type in the MobileNetV2-based design space (Figure 2). For input $\mathbf{x}$, the output of the i-th MBConv layer of the network is $o^i(\mathbf{x}) = \text{conv}(\mathbf{x}, \mathbf{w}^i | t^i_{k=5}, t^i_{e=6}, t^i_{e=3})$.

Differentiable NAS: To account for both the accuracy and inference latency of the searched ConvNet, we use a latency-aware formulation for the NAS problem (Wu et al., 2018):

$$\min_{\mathbf{w}} CE(\mathbf{w}|\mathbf{t}_k, \mathbf{t}_e) + \lambda \cdot \log(R(\mathbf{w}|\mathbf{t}_k, \mathbf{t}_e)) \quad (3)$$

where CE is the cross-entropy loss of the single-path model and R is the runtime in milliseconds (ms) of the searched NAS model on the target device. The coefficient λ modulates the trade-off between cross-entropy and runtime.

Runtime model: Prior art has showed that the on-device ConvNet runtime can be modeled as the sum of each i-th layer's runtime $R(\mathbf{w}|\mathbf{t}_k, \mathbf{t}_e) = \sum_i R^i(\mathbf{w}^i|\mathbf{t}^i_k, \mathbf{t}^i_e)$ (Wu et al., 2018; Cai et al., 2017; Stamoulis et al., 2018a). To preserve the differentiability of the objective, we formulate the per-layer R^i as a function of the NAS decisions. We profile the target mobile device (Pixel 1 smartphone) and we record the runtime for each candidate kernel operation per layer i. As a function of the expansion ratio decisions, we write:

$$\begin{aligned}R^i_e = \mathbb{1}(\|\mathbf{w}_{k,3}\|^2 > t_{e=3}) \cdot (R^i_{5\times5,3} + \\ \mathbb{1}(\left\|\mathbf{w}_{k,6\setminus3}\right\|^2 > t_{e=6}) \cdot (R^i_{5\times5,6} - R^i_{5\times5,3}))\end{aligned} \quad (4)$$

By incorporating the kernel size decision, the runtime is:

$$\begin{aligned}R^i = \frac{R^i_{3\times3,6}}{R^i_{5\times5,6}} \cdot R^i_e + \\ R^i_e \cdot (1 - \frac{R^i_{3\times3,6}}{R^i_{5\times5,6}}) \cdot \mathbb{1}(\left\|\mathbf{w}_{5\times5\setminus3\times3}\right\|^2 > t_{k=5})\end{aligned} \quad (5)$$

Table 1. Single-Path NAS achieves state-of-the-art image classification accuracy (%) on ImageNet for similar on-device latency setting compared to previous NAS methods ($\leq 80ms$ on Pixel 1), with up to $5,000\times$ reduced search cost in terms of number of epochs. *The search cost in epochs is estimated based on the claim (Cai et al., 2019) that ProxylessNAS is $200\times$ faster than MnasNet. ‡ChamNet does not detail the model derived under runtime constraints (Dai et al., 2018) so we cannot retrain or measure the latency.

METHOD	TOP-1 ACC (%)	TOP-5 ACC (%)	RUNTIME (MS)	SEARCH COST (EPOCHS)
MOBILENETV2 (SANDLER ET AL., 2018)	72.00	91.00	75.00	-
MOBILENETV2 (OUR IMPL.)	73.59	91.41	73.57	
RANDOM SEARCH	73.78 ± 0.85	91.42 ± 0.56	77.31 ± 0.9 MS	-
MNASNET (TAN ET AL., 2018)	74.00	91.80	76.00	40,000
MNASNET (OUR IMPL.)	74.61	91.95	74.65	
CHAMNET-B (DAI ET AL., 2018)	73.80	–	–	240‡
PROXYLESSNAS-R (CAI ET AL., 2019)	74.60	92.20	78.00	200*
PROXYLESSNAS-R (OUR IMPL.)	74.65	92.18	77.48	
FBNET-B (WU ET AL., 2018)	74.1	-	-	90
FBNET-B (OUR IMPL.)	73.70	91.51	78.33	
Single-Path NAS (**PROPOSED**)	**74.96**	**92.21**	79.48	**8 (3.75 HOURS)**

As in Equation 1, we relax the indicator function to a sigmoid function $\sigma(\cdot)$ when computing gradients. To evaluate the prediction accuracy of the runtime model, we generate 100 random ConvNets and we measure their runtime on the device. Our model can accurately predict the actual runtimes: the Root Mean Squared Error (RMSE) is $1.32ms$, which corresponds to an average 1.76% prediction error.

3. Experiments

Experimental Setup: We select Pixel 1 as the target device since it allows for a representative comparison with prior work that optimizes for this platform. We run our framework using `TensorFlow (TF)` on TPUs-v2 (Jouppi et al., 2017). We deploy the ConvNets on the device with `TF TFLite`. We profile on-device runtime using the Facebook AI Performance Evaluation Platform (FAI-PEP, 2018). We implement our trainable "superkernels" on `Keras`.

We apply our method to design ConvNets for image classification on ImageNet (Deng et al., 2009) running on Pixel 1 with an overall target latency of $80ms$. We train the derived *Single-Path* NAS model for 350 epochs. We summarize the results in Table 1. To enable a representative comparison of the search cost per method, we directly report the number of epochs per method, hence canceling out the effect of different hardware systems (GPU vs TPU hours).

State-of-the-art on-device image classification: *Single-Path* NAS achieves top-1 accuracy of **74.96%**, which is the new state-of-the-art ImageNet accuracy among hardware-efficient NAS methods. More specifically, **our method achieves better top-1 accuracy than ProxylessNAS by +0.31%**, while maintaining on par target latency of $\leq 80ms$ on the same target mobile phone. *Single-Path* NAS outperforms methods in this mobile latency range, *i.e.*, better than MnasNet (+0.35%), FBNet-B (+0.86%), and MobileNetV2 (+1.37%).

Reduced search cost: *Single-Path* NAS has a total search cost of 8 epochs, which is **5,000**$\times$ faster than MnasNet, **25**$\times$ faster than ProxylessNAS, and **11**$\times$ faster than FBNet. Specifically, MnasNet reports a total of 40k train epochs. In turn, ChamNet trains an accuracy predictor on 240 samples. ProxylessNAS reports $200\times$ search cost improvement over MnasNet, hence the overall cost is the TPU-equivalent of 200 epochs. Finally, FBNet reports 90 epochs. Overall, we search for $\sim 10k$ **steps** (8 epochs with a batch size of 1024), which corresponds to total wall-clock time of **3.75 hours** on a TPUv2. In particular, given than a TPUv2 has 2 chips with 4 cores each, this corresponds to a total of 30 TPU-hours.

4. Discussion & Future Work

Novel idea: The key insight behind our work is to revisit the one-shot **supernet** NAS design space with a *single-path* view, by formulating the NAS problem as *finding which subset of kernel weights to use* in each ConvNet layer. While concurrent works consider relaxed convolution formulations (Shin et al., 2018; Hundt et al., 2019; Guo et al., 2019), they either use design spaces and objectives that have been shown to be hardware inefficient (*e.g.*, cell-based space, FLOP count), or they do not intrinsically relax the kernel over both kernel-size and channels dimensions.

Future work: The efficiency of our *single-path* design space could enable future work beyond our differentiable NAS formulation and based on reinforcement learning or evolutionary methods. Moreover, our methodology can be flexibly extended to other hardware design goals, *e.g.*, power, memory, energy, and communication constraints (Dong et al., 2018; Stamoulis et al., 2018b;b). To this end and to foster reproducibility, unlike all recent hardware-efficient NAS methods which release pretrained models only, we open-source and fully document our method at: `https://github.com/dstamoulis/single-path-nas`.

References

Bender, G., Kindermans, P.-J., Zoph, B., Vasudevan, V., and Le, Q. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pp. 549–558, 2018.

Cai, E., Juan, D.-C., Stamoulis, D., and Marculescu, D. Neuralpower: Predict and deploy energy-efficient convolutional neural networks. In *Asian Conference on Machine Learning*, pp. 622–637, 2017.

Cai, H., Zhu, L., and Han, S. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.

Dai, X., Zhang, P., Wu, B., Yin, H., Sun, F., Wang, Y., Dukhan, M., Hu, Y., Wu, Y., Jia, Y., et al. Chamnet: Towards efficient network design through platform-aware model adaptation. *arXiv preprint arXiv:1812.08934*, 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Ding, R., Liu, Z., Chin, T.-W., Marculescu, D., and Blanton, R. Flightnns: Lightweight quantized deep neural networks for fast and accurate inference. In *2019 Design Automation Conference (DAC)*, 2019.

Dong, J.-D., Cheng, A.-C., Juan, D.-C., Wei, W., and Sun, M. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. *arXiv preprint arXiv:1806.08198*, 2018.

FAI-PEP. Facebook AI Performance Evaluation Platform. https://github.com/facebook/FAI-PEP, 2018.

Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019.

Hundt, A., Jain, V., and Hager, G. D. sharpdarts: Faster and more accurate differentiable architecture search. *arXiv preprint arXiv:1903.09900*, 2019.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12. IEEE, 2017.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Shin, R., Packer, C., and Song, D. Differentiable neural network architecture search. *OpenReview*, 2018.

Stamoulis, D., Cai, E., Juan, D.-C., and Marculescu, D. Hyperpower: Power-and memory-constrained hyperparameter optimization for neural networks. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018a.

Stamoulis, D., Chin, T.-W. R., Prakash, A. K., Fang, H., Sajja, S., Bognar, M., and Marculescu, D. Designing adaptive neural networks for energy-constrained image classification. In *Proceedings of the International Conference on Computer-Aided Design*. ACM, 2018b.

Stamoulis, D., Ding, R., Wang, D., Lymberopoulos, D., Priyantha, B., Liu, J., and Marculescu, D. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. *arXiv preprint arXiv:1904.02877*, 2019.

Tan, M., Chen, B., Pang, R., Vasudevan, V., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*, 2018.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *arXiv preprint arXiv:1812.03443*, 2018.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6), 2017.