InstantNet: Automated Generation and Deployment of Instantaneously Switchable-Precision Networks

Yonggan Fu
Rice University
yf22@rice.edu

Zhongzhi Yu
Rice University
zy42@rice.edu

Yongan Zhang
Rice University
yz87@rice.edu

Yifan Jiang
University of Texas at Austin
yifanjiang97@utexas.edu

Chaojian Li
Rice University
cl114@rice.edu

Yongyuan Liang
Sun Yat-sen University
liangyy58@mail2.sysu.edu.cn

Mingchao Jiang
Rice University
mj33@rice.edu

Zhangyang Wang
University of Texas at Austin
atlaswang@utexas.edu

Yingyan Lin
Rice University
yingyan.lin@rice.edu

***Abstract*—The promise of Deep Neural Network (DNN) powered Internet of Thing (IoT) devices has motivated a tremendous demand for automated solutions to enable fast development and deployment of efficient (1) DNNs equipped with instantaneous accuracy-efficiency trade-off capability to accommodate the time-varying resources at IoT devices and (2) dataflows to optimize DNNs' execution efficiency on different devices. Therefore, we propose InstantNet to automatically generate and deploy instantaneously switchable-precision networks which operate at variable bit-widths. Extensive experiments show that the proposed InstantNet consistently outperforms state-of-the-art designs. Our codes are available at: https://github.com/RICE-EIC/InstantNet.**

***Index Terms*—switchable-precision networks, NAS, dataflow**

I. Introduction

Powerful deep neural networks (DNNs)' prohibitive complexity calls for hardware efficient DNN solutions [1]–[3]. When it comes to DNNs' hardware efficiency in IoT devices, the model complexity (e.g., bit-widths), dataflows, and hardware architectures are major performance determinators. Early works mostly provide *static* solutions, i.e., once developed, the algorithm/dataflow/hardware are fixed, whereas IoT applications often have dynamic time/energy constraints over time. Recognizing this gap, recent works [4], [5] have attempted to develop efficient DNNs with instantaneous accuracy-cost trade-off capability. For example, switchable-precision networks (SP-Nets) [4], [5] can maintain a competitive accuracy under different bit-widths without fine-tuning under each bit-width, making it possible to allocate bit-widths on the fly for adapting IoT devices' instant resources over time.

Despite SP-Nets' great promise [4], [5], there are still major challenges in enabling their deployment into numerous IoT devices. First, existing SP-Nets are manually designed, largely limiting their extensive adoption as *each application* would require a *different* SP-Net. Second, while the best dataflow for SP-Nets under *different bit-widths* can be different and is an important determinator for their on-device efficiency [6], there is still a lack of a generic and publicly available framework that can be used to suggest optimal dataflows for SP-Nets under *each of their bit-widths* on *different IoT devices*. Both of the aforementioned hinder the fast development and deployment of SP-Nets powered DNN solutions for *diverse* hardware platforms of IoT devices. To tackle the aforementioned challenges, we make the following contributions:

- We propose InstantNet, an end-to-end framework that automates the development (i.e., the generation of SP-Nets given a dataset and target accuracy) and deployment (i.e., the generation of the optimal dataflows) of SP-Nets. To our best knowledge, InstantNet is **the first** to simultaneously target both development and deployment of SP-Nets.
- We develop switchable-precision neural architecture search (SP-NAS) that integrates an novel cascade distillation training to ensure that the generated SP-Nets under all bit-widths achieve the same or better accuracy than both *NAS generated* DNNs optimized for individual bit-widths and SOTA *expert-designed* SP-Nets.
- We propose AutoMapper, which integrates a generic dataflow space and an evolutionary algorithm to navigate over the discrete and large mapping-method space and automatically search for optimal dataflows given a DNN (e.g., SP-Nets under a selected bit-width) and target device.
- Extensive experiments based on real-device measurements and hardware synthesis validate InstantNet's effectiveness in consistently outperforming SOTA designs, e.g., achieving 84.68% real-device Energy-Delay-Product improvement while boosting the accuracy by 1.44%, over the most competitive competitor under the same settings.

II. Related works

Static and switchable-precision DNNs. DNN quantization aims to compress DNNs at the most fine-grained bit-level [7], [8]. To accommodate constrained and time-varying resources on IoT devices, SP-Nets [4], [5] aim for instantaneously switchable accuracy-efficiency trade-offs at the bit-level. However, designing such DNNs and the corresponding mapping methods for every scenario can be engineering-expensive and time consuming, considering the ever-increasing IoT devices with diverse hardware platforms and application requirements. As such, techniques that enable fast development and deployment of SP-Nets are highly desirable for expediting the deployment of affordable DNNs into numerous IoT devices.

978-1-6654-3274-0/21/$31.00 ©2021 IEEE

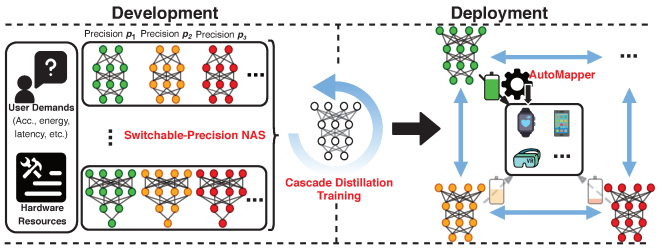

Fig. 1: Overview of InstantNet, which first generates SP-Nets with high accuracy under all bit-widths, and then suggests dataflows to maximize the generated SP-Nets' execution efficiency under different bit-widths on the target device.

Neural Architecture Search for efficient DNNs. To release human efforts from laborious manual design, NAS [9], [10] have been introduced to enable the automatic search for efficient DNNs with both competitive accuracy and hardware efficiency given the datasets. [11]–[13] incorporate quantization bit-widths into their search space and search for mixed-precision networks. However, all these NAS methods search for quantized DNNs with only one *fixed* bit-width, lacking the capability to instantly adapt to other bit-widths without fine-tuning.

Mapping DNNs to devices/hardware. When deploying DNNs into IoT devices with diverse hardware architectures, one major factor that determines hardware efficiency is the dataflow [6]. For devices with application-specific integrated circuit (ASIC) or FPGA hardware, various innovative dataflows [1], [14]–[16] have been developed to maximize the reuse opportunities. Recently, MAGNet has been proposed to automatically identify optimal dataflows and design parameters of a tiled architecture. However, its highly template-based design space, e.g., a pre-defined set of nested loop-orders, can restrict the generality and result in sub-optimal performance. Despite its promising performance, the exploration to automatically identify optimal mapping methods for DNNs with different bit-widths has not yet been considered.

III. The Proposed InstantNet Framework

Here we present our InstantNet framework, starting from an overview and then its key enablers including cascade distillation training (CDT), SP-NAS, and AutoMapper.

A. InstantNet overview

Fig. 1 shows an overview of InstantNet. Specifically, given the target application and device, it automates the development and deployment of SP-Nets. Specifically, InstantNet integrates two key enablers: (1) SP-NAS and (2) AutoMapper. SP-NAS incorporates an innovative cascade distillation to search for SP-Nets, providing IoT devices' desired instantaneous accuracy-efficiency trade-off capability. AutoMapper adopts a generic dataflow design space and an evolution-based algorithm to automatically search for optimal dataflows of SP-Nets under different bit-widths on the target device.

B. InstantNet training: Bit-Wise Cascade Distillation

Unlike generic quantized DNNs optimized to maximize accuracy under one individual bit-width, InstantNet aims to generate SP-Nets of which the accuracy *under all bit-widths* are the same or even higher than that of DNNs customized for individual bit-widths. The key challenge is to ensure high accuracy for lower bit-widths, which is particularly difficult for compact DNN models whose accuracy is more sensitive to quantization. For example, SOTA SP-Nets [5] fails to work on lower bit-widths when being applied to MobileNetV2 [17]. The above challenge has motivated InstantNet's CDT method, which takes advantage of the fact that the quantization noises of SP-Nets under adjacent or closer bit-widths are smaller. Our hypothesis is that distillation between adjacent and closer bit-widths will help to more smoothly enforce the accuracy (or activation distribution) of SP-Nets under low bit-widths to approach their full-precision counterparts. In this way, CDT can simultaneously boost accuracy of SP-Nets under all bit-widths by enforcing SP-Nets under each bit-width to have distillation from **all higher bit-widths**:

$$\begin{aligned} L_{total} &= \frac{1}{N}\sum_{i=0}^{N-1} L_{train}^{cas}(Q_i(\omega)) \\ where\ L_{train}^{cas}(Q_i(\omega)) &= L_{ce}(Q_i(\omega), label) \\ &+\beta\sum_{j=i+1}^{N-1} L_{mse}(Q_i(\omega), SG(Q_j(\omega))) \end{aligned} \tag{1}$$

where L_{total} is SP-Nets' average loss under all the N candidate bit-widths, L_{ce} and L_{mse} are the cross-entropy and mean square error losses, respectively, $Q_i(\omega)$ is the SP-Net characterized with weights ω under the i-th bit-width, β is a trade-off parameter, and SG is the stopping gradient function, i.e., gradient backpropagation from higher bit-widths is prohibited when calculating the distillation loss [5].

To verify the effectiveness of CDT, we visualize the prediction distribution (classification probability after softmax) of MobileNetV2 on CIFAR-100 under the bit-width set of 4, 8, 12, 16, 32 (quantized by SBM [18]) trained using different strategies in Fig. 2. We show the prediction distribution of the following three cases using a random sampled image from the test dataset to verify and visualize the effectiveness of our CDT: (1) 4-bit trained using vanilla distillation, i.e., only consider the distillation with 32-bit width, (2) 4-bit trained using our CDT technique and (3) the 32-bit trained network. We can observe that vanilla distillation fails to narrow the gap between 32-bit and the lowest 4-bit due to the large quantization error gap. This is actually a common phenomenon among efficient models with depthwise layers which are sensitive to low precision on all the considered test datasets, e.g., we observe that the validation accuracy of the 4-bit network with only the aforementioned vanilla distillation is around 1%, indicating the failure of vanilla distillation for tackling the bit-width set with a large dynamic range. In contrast, our CDT notably helps the prediction distribution of the 4-bit network smoothly evolve to that of the 32-bit one, and also boost its accuracy to 71.21%, verifying CDT's effectiveness.

C. InstantNet search: Switchable-Precision NAS

Here we introduce another key enabler of InstantNet, SP-NAS. To our best knowledge, InstantNet is **the first** to address

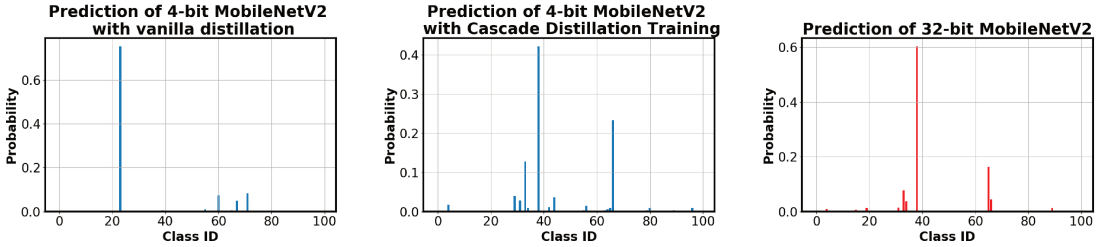

Fig. 2: Visualizing the prediction distribution of MobileNetV2 on CIFAR-100 under **(left)**: 4-bit training with vanilla distillation, **(middle)** 4-bit training with the proposed CDT, and **(right)** 32-bit training.

how to automatically generate networks which naturally favor working under various bit-widths. In addition, to resolve the performance bottleneck in SOTA SP-Nets (manually designed) [4], [5], i.e., large accuracy degradation under the lowest bit-width, we develop a heterogeneous scheme for updating the weights and architecture parameters. Specifically, we update the weights based on our CDT method (see Eq. 1) which explicitly incorporates switchable-bit property into the training process; and for updating the architecture parameters of SP-Net, we adopt *only the weights under the lowest bit-width*, for generating networks forced to inherently tackle SP-Nets' bottleneck of high accuracy loss under the lowest bit-width:

$$\begin{aligned} &\min_{\alpha} L_{val}(Q_0(\omega^*), \alpha) + \lambda L_{eff}(\alpha) \\ &s.t. \quad \omega^* = \arg\min_{\omega} \frac{1}{N} \sum_{i=0}^{N-1} L_{train}^{cas}(Q_i(\omega), \alpha) \end{aligned} \tag{2}$$

where ω and α are the supernet's weights [19] and architecture parameters, respectively, L_{eff} is an efficiency loss (e.g., energy cost), and $Q_0(\omega)$ is the SP-Net under the lowest bit-width. Without loss of generality, here we adopt SOTA differentiable NAS [19] and search space [20].

D. InstantNet deploy: Evolution-based AutoMapper

This subsection introduces InstantNet's AutoMapper, of which an overview is shown in Fig. 3. Motivated by the fact that different mapping methods can have orders-of-magnitude difference in hardware efficiency [6], AutoMapper aims to accept (1) DNNs (e.g., SP-Nets generated by our SP-NAS), (2) the target device, and (3) target hardware efficiency, and then generate mapping methods that maximize both the task accuracy and hardware efficiency of the given SP-Nets under all bit-widths when being executed on the target device.

Generic Dataflow Design Space. A generic dataflow design space is a prerequisite for effective algorithmic exploration and optimization of on-device dataflows, yet is challenging to develop. There are numerous choices for how to temporally and spatially schedule all the DNN's operations to be executed in the target accelerators. Specifically, as there are many more operations in DNNs than the number of operations (e.g., $19.6E+9$ [21] vs. 900 MACs [22] assuming a 16-bit precision) an IoT device can execute in each clock cycle, numerous possible dataflows exist for running DNNs on a device.

To tackle the aforementioned challenge, we propose a generic design space for on-device dataflows, which (1) covers all design choices for generalization and (2) is easy to understand for ease of adoption. Our proposed space leverages commonly used nested *for-loop* descriptions [1], [23]. For better illustration, here we describe the high-level principles. From a nested *for-loop* description, our dataflow space extracts all possible choices characterized by the following factors:

loop-order: the processing order of each dimension within each memory hierarchy, and can be derived from all possible permuted choices without overlap.

loop-size: the no. of operations in one iteration of a specific dimension, which can not be easily determined. We design a simple analytical algorithm to derive all possible choices.

Pipeline/multi-cycle: use pipeline or multi-cycle. The former processes a small chunk of each layer in a pipeline manner, while the latter processes all the layers sequentially.

Considering AlexNet [24] and six layers of nested loops, there are over 10^{27} **total number of discrete mapping-method choices**, posing a great need for developing efficient and effective search algorithms.

Evolutionary Search Algorithm. To navigate the large and discrete space of mapping methods, we adopt an evolutionary based search algorithm, considering that evolutionary algorithms have more exploitation than random search and are better suited for the highly discrete space [25], [26]. Specifically, we will keep track of the hardware efficiency ranking of the current sampled mapping methods at each iteration. Afterwards, if the pool size of current samples is smaller than a specified value, we select a few of the best performing sampled mapping methods and randomly perturb a small number of their features associated with the aforementioned design factors to generate new mapping methods to be evaluated in the next iteration; otherwise, new mapping methods with completely randomly selected design factors will be generated. We summarize our Evolutionary AutoMapper in Alg.1.

IV. Experiment Results

We first describe our experiment setup and then evaluate each enabler of InstantNet, i.e., CDT, SP-NAS, and AutoMapper. After that, we benchmark InstantNet over SOTA SP-Nets on SOTA accelerators [1], [15], [27].

A. Experiment setup

1) Algorithm experiment setup: **Datasets & Baselines.** We consider three datasets (CIFAR-10/CIFAR-100/ImageNet), and evaluate InstantNet over (1) all currently published SP-Nets (AdaBits [4] and SP [5]) with the DoReFa [28] quantizer and (2) a SOTA quantized DNN method SBM [18] to train a SOTA compact DNN MobileNetV2 [17] under individual bits.

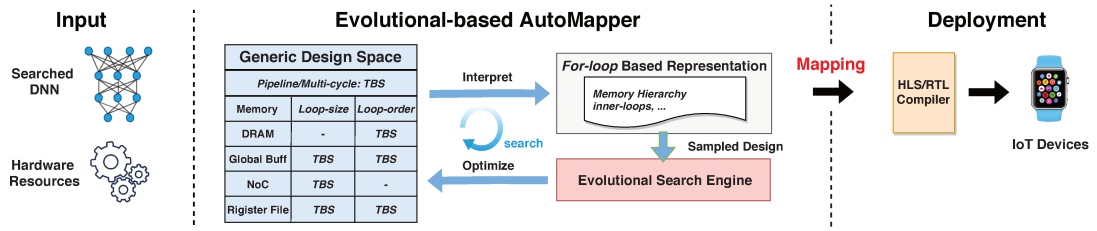

Fig. 3: Overview of the goal, generic dataflow space, and InstantNet's AutoMapper, where TBS denotes "to be searched".

TABLE I: InstantNet's CDT over SBM [18] (SOTA training for quantized DNNs) and SOTA SP-Nets (SP [5] and AdaBits [4]) on **MobileNetV2** and CIFAR-100 in terms of test accuracy (%), where the values in the bracket represent the accuracy drop of the baseline methods compared to our CDT.

Bit-widths	SBM [18]	SP [5]	AdaBits [4]	CDT (Proposed)
4	70.55 (**-0.60**)	66.75 (**-4.40**)	68.07 (**-3.08**)	**71.15**
8	74.40 (**-0.72**)	71.69 (**-3.43**)	73.86 (**-1.26**)	**75.12**
12	74.87 (**-0.16**)	74.16 (**-0.87**)	73.65 (**-1.38**)	**75.03**
16	75.03 (**-0.19**)	74.23 (**-0.99**)	73.87 (**-1.35**)	**75.22**
32	75.23 (**+0.25**)	74.11 (**-0.87**)	74.51 (**-0.47**)	**74.98**
4	70.55 (**-0.53**)	67.63 (**-3.45**)	68.37 (**-2.71**)	**71.08**
5	74.13 (**-0.32**)	72.95 (**-1.50**)	73.52 (**-0.93**)	**74.45**
6	74.69 (**-0.33**)	74.15 (**-0.87**)	74.60 (**-0.42**)	**75.02**
8	74.40 (**-0.64**)	74.99 (**-0.05**)	75.02 (**-0.02**)	**75.04**

Search and training on CIFAR-10/100 and ImageNet. Search space: we adopt the same search space as [20] except the stride settings for each group to adapt to the resolution of the input images in CIFAR-10/100. Search settings. On CIFAR-10/100, we search for 50 epochs with batch size 64. In particular, we (1) update the supernet weights with our cascade distillation technique as in Eq.(2) on half of the training dataset using an SGD optimizer with a momentum of 0.9 and an initial learning rate (LR) 0.025 at a cosine decay, and (2) update network architecture parameters with the lowest bit-width as in Eq.(2) on the other half of the training dataset using an Adam optimizer with a momentum of 0.9 and a fixed LR 3e-4. We apply gumbel softmax on the architecture parameters as the contributing coefficients of each option to the supernet (following [20]), where the initial temperature is 3 and then decayed by 0.94 at each epoch. On ImageNet, we follow the same hyper-parameter settings for the network search as [20]. Evaluate the derived networks: for training the derived networks from scratch using our CDT, on CIFAR-10/100 we adopt an SGD optimizer with a momentum of 0.9 and an initial LR 0.025 at a cosine decay. Each network is trained for 200 epochs with batch size 128. On ImageNet, we follow [20].

Algorithm 1: Evolutionary AutoMapper

Input: Efficiency Goal, DNN, Design Space (DS)
Output: Optimal algorithm-to-device mapping
Build a *pool* with n random samples from DS
while *Efficiency Goal not met* **do**
  if $size(pool) \leq n$ **then**
    for m *iterations* **do**
      Random Pick $p \in pool$
      Random Perturb k features of p
      Add p to *pool*
    end
  else
    Rank the samples in *pool* with the given DNN
    Remove the worst m samples from *pool*
  end
end
return *optimal mapping in pool*

2) Hardware experiment setup: **Implementation methodology.** We consider two commonly used IoT hardware platforms, i.e., ASIC and FPGA, for evaluating our AutoMapper. Specifically, for FPGA, we adopt the Vivado HLx design toolflow where we first synthesize the mapping-method design in C++ via Vivado HLS, and then plug the HLS exported IPs into a Vivado IP integrator to generate the corresponding bit streams, which are programmed into the FPGA board for on-board execution and measurements; for ASIC, we synthesize the Verilog designs based on the generated dataflows using a Synopsys Design Compiler on a commercial CMOS technology, and then place and route using a Synopsys IC Compiler for obtaining the resulting design's actual area.

Baselines. We evaluate AutoMapper over expert/tool generated SOTA dataflows for both FPGA and ASIC platforms, including DNNBuilder [15] and CHaiDNN [27] for FPGA, and Eyeriss [1] and MAGNet [6] for ASIC. For DNNBuilder [15], MAGNet [6] and CHaiDNN [27], we use their reported results; For Eyeriss [1], we use their own published and verified simulator [29] to obtain their results.

B. Ablation study of InstantNet: CDT

Experiment settings. For evaluating InstantNet's CDT, we benchmark it over an SOTA quantized DNN training method (independently train DNNs at each bit-width) and two SP-Nets (AdaBits [4] and SP [5]). In light of our IoT application goal, we consider MobileNetV2 [17] (an SOTA efficient model balancing task accuracy and hardware efficiency) with CIFAR-100, and adopt two different bit-width sets with both large and narrow bit-width dynamic ranges. Without losing generality, our CDT is designed with SOTA quantizer SBM [18] and switchable batch normalization as in SP [5].

Results and analysis. From Tab. I, we have three observations: (1) our CDT consistently outperforms the two SP-Net baselines under all the bit-widths, verifying CDT's effectiveness and our hypothesis that progressively distilling from all higher bit-widths can help more smoothly approach accuracy of the full-precision; (2) CDT is particularly capable of boosting

TABLE II: CDT over independently trained SBM [18] on **ResNet-38**, where the values in the bracket represent CDT's accuracy gain over SBM (the higher, the better)

Dataset	CIFAR-10		CIFAR-100	
Bit-widths	SBM	**CDT (Proposed)**	SBM	**CDT (Proposed)**
4	90.91	**91.45 (+0.54)**	63.82	**64.18 (+0.36)**
8	92.78	**93.03 (+0.25)**	66.71	**67.45 (+0.74)**
12	92.75	**93.06 (+0.31)**	67.13	**67.42 (+0.29)**
16	92.90	**93.09 (+0.19)**	67.17	**67.50 (+0.33)**
32	92.5	**93.08 (+0.58)**	67.18	**67.47 (+0.29)**
4	90.91	**91.88 (+0.97)**	63.82	**64.12 (+0.30)**
5	92.35	**92.56 (+0.21)**	66.20	**66.68 (+0.48)**
6	92.80	**92.93 (+0.13)**	66.48	**66.55 (+0.07)**
8	92.78	**93.02 (+0.24)**	66.71	**66.88 (+0.17)**

TABLE III: CDT over independently trained SBM [18] on **ResNet-74**, where the values in the bracket represent CDT's accuracy gain over SBM (the higher, the better).

Dataset	CIFAR-10		CIFAR-100	
Bit-widths	SBM	**CDT (Proposed)**	SBM	**CDT (Proposed)**
4	91.82	**92.34 (+0.52)**	66.31	**67.35 (+1.04)**
8	93.22	**93.56 (+0.34)**	69.85	**69.98 (+0.13)**
12	93.26	**93.53 (+0.27)**	69.97	**69.99 (+0.02)**
16	93.40	**93.51 (+0.11)**	69.92	**70.01 (+0.09)**
32	93.38	**93.49 (+0.11)**	69.46	**69.98 (+0.52)**
4	91.82	**92.51 (+0.69)**	66.31	**67.34 (+1.03)**
5	92.98	**93.54 (+0.56)**	68.66	**69.49 (+0.83)**
6	93.19	**93.47 (+0.28)**	69.42	**69.65 (+0.23)**
8	93.22	**93.72 (+0.50)**	69.85	**70.02 (+0.17)**

accuracy in low bit-widths which has been shown to be the bottleneck in exiting SP-Nets [4], e.g., a 2.71%~4.4% higher accuracy on the lowest 4-bit over the two SP-Net baselines; and (3) CDT always achieves a higher or comparable accuracy over the SOTA quantized DNN training method SBM that independently trains and optimizes each individual bit-width: for bit-widths ranging from 4-bit to 8-bit, CDT achieves 0.32%~0.72% improvement in accuracy over SBM, indicating the effectiveness of our CDT in boosting DNNs' accuracies under lower bit-widths.

We also benchmark CDT on ResNet-38/74 [30] with CIFAR-10/CIFAR-100 over independently trained SBM [18]. As shown in Tab. II and Tab. III for ResNet-38 and ResNet-74, respectively, CDT consistently achieves a better/comparable accuracy (0.02%~1.04%) over the independently trained ones under all the models/datasets/bit-widths, and notably boosts the accuracy of the lowest bit-width (4-bit) by 0.30%~1.04%.

To evaluate CDT's performance when involving extremely low bit-width (2-bit), we further benchmark CDT on ResNet-18 [31] and TinyImageNet [32] over the SP [18] baseline. The results are shown in Tab. IV. It can be observed that the CDT

TABLE IV: CDT over SP [18] on ResNet-18 and TinyImageNet in terms of test accuracy, where the values in the bracket represent CDT's accuracy gain over SBM.

Bit-widths		Methods	
Weight	Activation	SP	**CDT (Proposed)**
2	2	47.8	**52.3 (+4.5)**
2	32	50.5	**51.3 (+0.8)**
32	2	51.8	**53.4 (+1.6)**

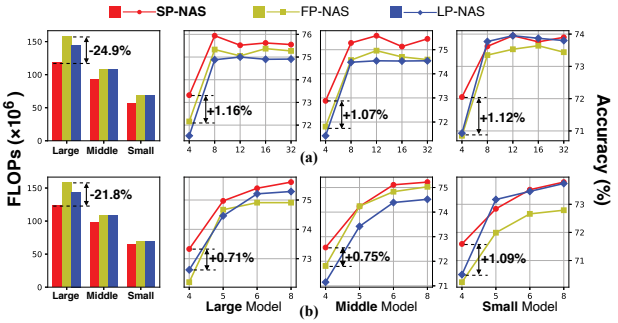

Fig. 4: InstantNet's SP-NAS over Full-Precision-NAS (FP-NAS) and Low-Precision-NAS (LP-NAS) on CIFAR-100 under large, middle, and small FLOPs constraints trained for two bit-width sets: (a) [4, 8, 12, 16, 32], and (b) [4, 5, 6, 8].

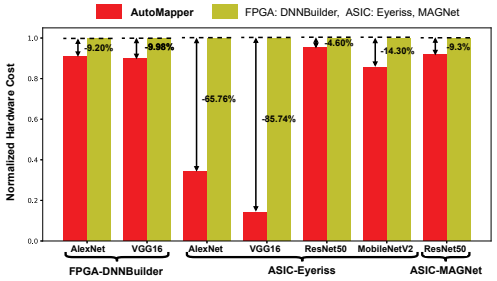

Fig. 5: AutoMapper over SOTA expert-crafted and tool generated dataflows on FPGA/ASIC.

is particularly effective in boosting the accuracy in lower bit-widths. Specifically, when the weights and activations both adopt 2-bit, the proposed CDT achieves a 4.5% higher accuracy than that of the baseline SP method.

C. Ablation study of InstantNet: SP-NAS

From Fig. 4, we can see that: (1) SP-NAS consistently outperforms the baselines at the lowest bit-width, which is the bottleneck in SOTA SP-Nets [4], while offering a higher/comparable accuracy at higher bit-widths. Specifically, SP-NAS achieves a 0.71%~1.16% higher accuracy over the strongest baseline at the lowest bit-width on both bit-width sets under the three FLOPs constraints; and (2) SP-NAS shows a notable superiority on the bit-width set with a larger dynamic range which is more favorable for IoT applications as larger bit-width dynamic ranges provide more flexible instantaneous accuracy-efficiency trade-offs. Specifically, compared with the strongest baseline, SP-NAS achieves a 1.16% higher accuracy at the lowest bit-width and a 0.25%~0.61% higher accuracy at other bit-widths, while offering a 24.9% reduction in FLOPs on the bit-width set [4, 8, 12, 16, 32]. This experiment validates that SP-NAS can indeed effectively tackle SP-Nets' bottleneck and improve its scalability over previous search methods which fail to guarantee accuracy at lower bit-widths.

D. Ablation study of InstantNet: AutoMapper

As shown in Fig. 5, we can see that (1) the dataflows suggested by AutoMapper (taking less than 10 minutes of search time) even outperforms SOTA expert-crafted designs: the mapping generated by AutoMapper achieves 65.76% and

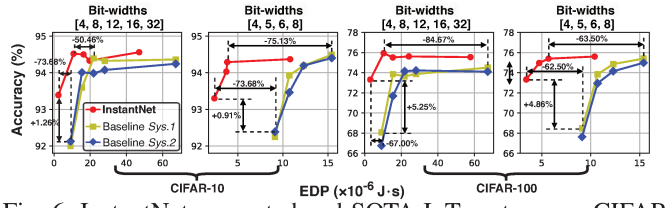

Fig. 6: InstantNet generated and SOTA IoT systems on CIFAR-10/100 under two bit-width sets.

85.74% EDP reduction on AlexNet [24] and VGG16 [21] compared with Eyeriss [1], respectively; (2) AutoMapper achieves a higher cost savings on ASIC than that of FPGA. This is because ASIC designs are more flexible than FPGA in their dataflows and thus achieve superior performance when exploring using effective automated search tools; and (3) when comparing with MAGNet, we have roughly 9.3% reduction in terms of the energy cost. MAGNet only used a pre-defined set of loop-orders to cover different dataflow scenarios, which may not generically fit network's diverse layer structures, thus resulting in inferior performance.

E. InstantNet over SOTA systems

Results and analysis on CIFAR-10/100. As shown in Fig. 6, we can see that (1) InstantNet generated systems consistently outperforms the SOTA baselines in terms of the trade-off between accuracy and EDP (a commonly-used hardware metric for ASIC) by achieving a higher or comparable accuracy and better EDP under lower bit-widths over the baselines. In particular, InstantNet can achieve up to 84.67% reduction in EDP with a 1.44% higher accuracy on CIFAR-100 and the bit-width set of $[4, 8, 12, 16, 32]$; and (2) InstantNet always surpasses the SOTA baselines under the bottleneck bit-width, i.e., the lowest one, with a 62.5%~73.68% reduction in EDP and a 0.91%~5.25% higher accuracy, which is notably more practical for real-world IoT deployment.

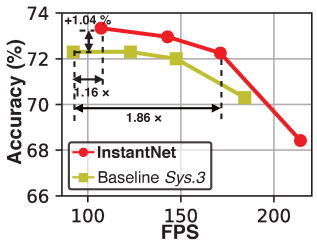

Fig. 7: InstantNet and SOTA IoT systems on ImageNet with bit-widths of $[4, 5, 6, 8]$.

Results and analysis on ImageNet. As shown in Fig. 7, InstantNet generated system achieves a 1.86× improvement in Frame-Per-Second (FPS) while having a comparable accuracy (-0.05%) over the SOTA FPGA based IoT system.

V. Conclusion

We propose an *automated* framework termed **InstantNet** to automatically search for SP-Nets (i.e., capable of operating at variable bit-widths) that can achieve the same or even better accuracy than DNNs optimized for individual bit-widths, and to generate optimal dataflows to maximize efficiency when DNNs are executed under various bit-widths on different devices. Extensive experiments show that InstantNet has promised an effective automated framework for expediting development and deployment of efficient DNNs for numerous IoT applications with diverse specifications.

Acknowledgement

The work is supported by the National Science Foundation (NSF) through the Energy, Power, Control, and Networks (EPCN) program (Award number: 1934755, 1934767).

References

[1] Y. Chen *et al.*, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *ISCA'16*, 2016, pp. 367–379.

[2] S. Liu *et al.*, "On-demand deep model compression for mobile devices: A usage-driven model selection framework," in *MobiSys'18*.

[3] Y. Lin *et al.*, "Predictivenet: An energy-efficient convolutional neural network via zero prediction," in *ISCAS'2017*, pp. 1–4.

[4] Q. Jin *et al.*, "Adabits: Neural network quantization with adaptive bit-widths," *arXiv preprint arXiv:1912.09666*, 2019.

[5] L. Guerra *et al.*, "Switchable precision neural networks," *arXiv preprint arXiv:2002.02815*, 2020.

[6] R. Venkatesan *et al.*, "Magnet: A modular accelerator generator for neural networks," in *ICCAD*, 2019, pp. 1–8.

[7] Y. Fu *et al.*, "FracTrain: Fractionally squeezing bit savings both temporally and spatially for efficient dnn training," in *NeurIPS'20*, vol. 33.

[8] Y. Fu *et al.*, "CPT: Efficient deep neural network training via cyclic precision," in *ICLR'20*.

[9] B. Barret Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016.

[10] Y. Fu *et al.*, "Autogan-Distiller: Searching to compress generative adversarial networks," in *ICML'20*.

[11] K. Wang *et al.*, "Haq: Hardware-aware automated quantization with mixed precision," in *CVPR*, 2019, pp. 8612–8620.

[12] Y. Chen *et al.*, "Joint neural architecture search and quantization," *arXiv preprint arXiv:1811.09426*, 2018.

[13] B. Wu *et al.*, "Mixed precision quantization of convnets via differentiable neural architecture search," *arXiv preprint arXiv:1812.00090*, 2018.

[14] C. Zhang *et al.*, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *FPGA*, 2015, p. 161–170.

[15] X. Zhang *et al.*, "Dnnbuilder: an automated tool for building high-performance dnn hardware accelerators for fpgas," in *ICCAD*, 2018.

[16] Y. Zhao *et al.*, "SmartExchange: Trading higher-cost memory storage/access for lower-cost computation," in *ISCA'20*, p. 954–967.

[17] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[18] R. Banner *et al.*, "Scalable methods for 8-bit training of neural networks," in *NeurIPS*, 2018, pp. 5145–5153.

[19] H. Liu *et al.*, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[20] B. Wu *et al.*, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *CVPR*, 2019.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[22] Xilinx Inc., "Xilinx zynq-7000 soc zc706 evaluation kit," https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html.

[23] Y. Zhao *et al.*, "DNN-Chip Predictor: An analytical performance predictor for dnn accelerators with various dataflows and hardware architectures," in *ICASSP'2020*.

[24] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012, pp. 1097–1105.

[25] E. Real *et al.*, "Regularized evolution for image classifier architecture search," *arXiv preprint arXiv:1802.01548*, 2018.

[26] A. Samajdar *et al.*, "Genesys: Enabling continuous learning through neural network evolution in hardware," in *MICRO*, 2018, pp. 855–866.

[27] Xilinx Inc., "Xilinx/chaidnn: Hls based deep neural network accelerator library for xilinx ultrascale+ mpsocs," https://github.com/Xilinx/CHaiDNN.

[28] S. Zhou *et al.*, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *preprint arXiv:1606.06160*, 2016.

[29] M. Gao *et al.*, "Tetris: Scalable and efficient neural network acceleration with 3d memory," in *ASPLOS*, New York, NY, USA, 2017, p. 751–764.

[30] X. Wang *et al.*, "Skipnet: Learning dynamic routing in convolutional networks," in *ECCV*, 2018, pp. 409–424.

[31] K. He *et al.*, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[32] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, 2015.