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

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 the execution efficiency of DNNs on different devices. Therefore, we propose InstantNet to automatically generate and deploy instantly switchable-precision networks which can operate at variable bit-widths. Extensive experiments show that the proposed InstantNet consistently outperforms state-of-the-art designs. All codes will be released upon acceptance.

Index Terms—switchable-precision networks, NAS, dataflow

I. INTRODUCTION

Powerful deep neural networks (DNNs)’ prohibitive complexity calls for hardware efficient DNN solutions [1], [2]. 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 [3]. Recognizing this gap, recent works [3]–[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 [3]–[5], there are still major challenges in enabling their deployments 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 determinant 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 fast DNN development and deployment.
• 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 dataset and device.

II. RELATED WORKS

Static and switchable-precision DNNs. DNN quantization aims to compress DNNs in the most fine-grained bit-level [7]–[10]. To better accommodate constrained and time-varying resources at IoT devices, SP-Nets [4], [5] aim for instantaneous switchable accuracy-efficiency trade-offs from the bit-level. Pioneering SP-Nets [4], [5] push the accuracy-efficiency trade-off frontiers to offer more favorable DNNs for IoT devices. 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.

Neural Architecture Search for efficient DNNs. To release human efforts from laborious manual design, NAS [11], [12] has been introduced to enable automatically searching for efficient DNNs with both competitive accuracy and hardware efficiency given the datasets. [9], [10], [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 assignment, lacking the capability to instantly adapt their networks to other bit-widths without extra 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], [2], [14], [15] have been developed to maximize the reuse opportunities. Recently, MAGNet is proposed to automatically identify optimal dataflows and design parameters of a tiled architecture. However, their highly template-based design space, e.g., pre-defined set of nested loop-orders, can restrict the generality and result in sub-optimal performance. Despite their promising performance, the exploration to automatically identify optimal mapping methods for DNNs with different bit-widths have not yet been considered.

III. THE PROPOSED INSTANET 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, where the SP-NAS incorporates an innovative cascade distillation to search for SP-Nets, providing IoT devices’ desired instantaneous accuracy-efficiency trade-off capability, and 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 dedicatedly designed 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 are more sensitive to quantization. For example, SOTA SP-Nets [5] fails to work on lower bit-widths when being applied to MobileNetV2 [16]. The above challenge has motivated InstantNet’s CDT method, which takes advantages of the fact that 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:

\[
L_{total} = \frac{1}{N} \sum_{i=0}^{N-1} L_{train}^{cas}(Q_i(\omega))
\]

where \( L_{train}^{cas}(Q_i(\omega)) = L_{ce}(Q_i(\omega), label) \)

\[+\beta \sum_{j=1}^{N-1} L_{mse}(Q_j(\omega), SG(Q_j(\omega))) \]

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 \( \omega \) under the \( i \)-th bit-width, \( \beta \) 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 [17]) 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
how to automatically generate networks which naturally favor to work under various bit-widths? In addition, to resolve the performance bottleneck in SOTA SP-Nets (manually designed) \cite{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 enforced to inherently tackle SP-Nets’ bottleneck of high accuracy loss under the lowest bit-width:

$$\min_{\alpha} L_{\text{val}}(Q_{0}(\omega^*), \alpha) + \lambda L_{\text{eff}}(\alpha)$$

s.t. $\omega^* = \arg\min_{\omega} \frac{1}{N} \sum_{i=0}^{N-1} L_{\text{train}}^\text{cas}(Q_{i}(\omega), \alpha)$

(2)

where $\omega$ and $\alpha$ are the supernet’s weights \cite{12} and architecture parameters, respectively, $L_{\text{eff}}$ 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 algorithm to derive high accuracy loss under the lowest bit-width.

**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 \cite{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 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, there are as much more operations in DNNs than the number of operations (e.g., 19.6E + 9 \cite{19} vs. 900 MACs \cite{20} assuming a 16-bit data precision) an IoT device can execute in each clock cycle, numerous ways of dataflows exist for running DNNs on a target 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 description \cite{1,21}. For better illustration, here we describe the high-level principles. From a nested for-loop description, our dataflow design space extracts all possible choices characterized by the following design 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 \cite{22} 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 while better suiting the highly discrete space \cite{23,24}. 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 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 a 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 \cite{1,15,25}.

**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...
TABLE I: InstantNet’s CDT over SBM [17] (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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>70.55 (-0.60)</td>
<td>66.75 (-4.40)</td>
<td>68.07 (-3.08)</td>
<td>71.15</td>
</tr>
<tr>
<td>8</td>
<td>74.40 (-0.72)</td>
<td>71.69 (-3.43)</td>
<td>73.86 (-1.26)</td>
<td>75.12</td>
</tr>
<tr>
<td>12</td>
<td>74.87 (-0.16)</td>
<td>74.16 (-0.87)</td>
<td>73.65 (-1.38)</td>
<td>75.03</td>
</tr>
<tr>
<td>16</td>
<td>75.03 (-0.19)</td>
<td>74.23 (-0.99)</td>
<td>73.87 (-1.35)</td>
<td>75.22</td>
</tr>
<tr>
<td>32</td>
<td>75.23 (-0.25)</td>
<td>74.11 (-0.87)</td>
<td>74.51 (-0.47)</td>
<td>74.98</td>
</tr>
<tr>
<td>4</td>
<td>70.55 (-5.33)</td>
<td>67.63 (-3.45)</td>
<td>68.37 (-2.71)</td>
<td>71.08</td>
</tr>
<tr>
<td>5</td>
<td>74.13 (-3.32)</td>
<td>72.95 (-1.50)</td>
<td>73.52 (-0.93)</td>
<td>74.45</td>
</tr>
<tr>
<td>6</td>
<td>74.69 (-3.33)</td>
<td>74.15 (-0.97)</td>
<td>74.60 (-0.42)</td>
<td>75.02</td>
</tr>
<tr>
<td>8</td>
<td>74.40 (-0.64)</td>
<td>74.99 (-0.05)</td>
<td>75.02 (-0.02)</td>
<td>75.04</td>
</tr>
</tbody>
</table>

(AdaBits [4] and SP [5]) with the DoReFa [26] quantizer and (2) a SOTA quantized DNN method SBM [17] to train a SOTA compact DNN MobileNetV2 [10] under individual bits. Evaluate the derived networks: for training the derived networks from scratch, we follow [12].

Search and training on CIFAR-10/100 and ImageNet. Search space: we adopt the same search space as [18] 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 [18]), 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 [18]. 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 [18].

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 tool-flow 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 one 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 [25] for FPGA, and Eyeriss [1] and MAGNet [9] for ASIC. For DNNBuilder [15], MAGNet [9] and CHaiDNN [25], we use their reported results; For Eyeriss [1], we use their own published and verified simulator [27] to obtain their results.

B. Ablation study of InstantNet: CDT

Experiment settings. For evaluating InstantNet’s CDT, we benchmark it over a 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 [10] (a 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 [17] and switchable batch normalization as in SP [5].

Results and analysis. From Tab. [1] 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 distill from all higher bit-
TABLE II: CDT over independently trained SBM [17] on ResNet-38, where the values in the bracket represent CDT’s accuracy gain over SBM (the higher, the better).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit-widths</td>
<td>SBM</td>
<td>CDT (Proposed)</td>
</tr>
<tr>
<td>4</td>
<td>90.91</td>
<td>91.45 (+0.54)</td>
</tr>
<tr>
<td>8</td>
<td>92.78</td>
<td>93.03 (+0.25)</td>
</tr>
<tr>
<td>12</td>
<td>92.75</td>
<td>93.06 (+0.31)</td>
</tr>
<tr>
<td>16</td>
<td>92.90</td>
<td>93.09 (+0.19)</td>
</tr>
<tr>
<td>32</td>
<td>92.75</td>
<td>93.08 (+0.58)</td>
</tr>
</tbody>
</table>

TABLE III: CDT over independently trained SBM [17] on ResNet-74, where the values in the bracket represent CDT’s accuracy gain over SBM (the higher, the better).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit-widths</td>
<td>SBM</td>
<td>CDT (Proposed)</td>
</tr>
<tr>
<td>4</td>
<td>91.82</td>
<td>92.34 (+0.52)</td>
</tr>
<tr>
<td>8</td>
<td>93.22</td>
<td>93.56 (+0.34)</td>
</tr>
<tr>
<td>12</td>
<td>93.26</td>
<td>93.53 (+0.27)</td>
</tr>
<tr>
<td>16</td>
<td>93.40</td>
<td>93.51 (+0.11)</td>
</tr>
<tr>
<td>32</td>
<td>93.38</td>
<td>93.49 (+0.11)</td>
</tr>
</tbody>
</table>

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) [4,8,12,16,32].

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

widths can help more smoothly approach accuracy of the full-precision; (2) CDT is particularly capable of boosting 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-Nets 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 [29] with CIFAR-10/CIFAR-100 over independently trained SBM [17]. As shown in Tab. II and Tab. III for ResNet-38 and ResNet-74, respectively, we can observe that CDT consistently achieves a better or comparable accuracy (0.02%~1.04%) compared with 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 [29] and TinyImageNet [30] over the SP [17] baseline. The results are shown in Tab. IV. It can be observed that the CDT 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 (i.e., handle larger bit-width sets) 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 85.74% EDP reduction on AlexNet [22] and VGG16 [19] 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 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]; (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 deployments.

Results and analysis on ImageNet. As shown in Fig. 7 InstantNet generated system achieves a 1.86x 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. Through real-device measurements, extensive experiments and ablation studies show that InstantNet has promised an effective automated framework for expediting development and deployment of efficient DNNs for numerous IoT applications with diverse specifications.

REFERENCES