NAWQ-SR: A Hybrid-Precision NPU Engine for Efficient On-Device Super-Resolution

Stylianos I. Venieris, Mario Almeida, Royson Lee, and Nicholas D. Lane

Abstract—In recent years, image and video delivery systems have begun integrating deep learning super-resolution (SR) approaches, leveraging their unprecedented visual enhancement capabilities while reducing reliance on networking conditions. Nevertheless, deploying these solutions on mobile devices still remains an active challenge as SR models are excessively demanding with respect to workload and memory footprint. Despite recent progress on on-device SR frameworks, existing systems either penalize visual quality, lead to excessive energy consumption or make inefficient use of the available resources. This work presents NAWQ-SR, a novel framework for the efficient on-device execution of SR models. Through a novel hybrid-precision quantization technique and a runtime neural image codec, NAWQ-SR exploits the multi-precision capabilities of modern mobile NPUs in order to minimize latency, while meeting user-specified quality constraints. Moreover, NAWQ-SR selectively adapts the arithmetic precision at run time to equip the SR DNN's layers with wider representational power, improving visual quality beyond what was previously possible on NPUs. Altogether, NAWQ-SR achieves an average speedup of 7.9×, 3× and 1.91× over the state-of-the-art on-device SR systems that use heterogeneous processors (MobiSR), CPU (SplitSR) and NPU (XLSR), respectively. Furthermore, NAWQ-SR delivers an average of 3.2× speedup and 0.39 dB higher PSNR over status-quo INT8 NPU designs, but most importantly mitigates the negative effects of quantization on visual quality, setting a new state-of-the-art in the attainable quality of NPU-based SR.

Index Terms—Deep neural networks, mobile computing, super-resolution

1 Introduction

With the rapid rise of Internet content delivery services and devices that support higher resolution content, images and videos are predicted to account for 82% of the global Web traffic [1]. Mobile applications, in particular, constitute a great proportion of this growth, as services such as live streaming, video-conferencing, and video-on-demand have been on the rise. For instance, popular video app TikTok has over 50 million daily users with increases of 55% in unique users and 93.7% in the average time spent per user in just six months [2]. To meet such demands, mobile systems are required to maximize both the user satisfaction and their quality of experience (QoE).

A primary challenge of this class of mobile systems is their sensitivity to networking conditions. In real-world cellular networks, the network speed fluctuates substantially, and poor connectivity leads to excessive response times, dropped frames or video stalling, which rapidly degrade the QoE [3], [4], [5], [6]. This phenomenon is further aggravated by the increasing number of users which compete for the same pool of network resources and create contention [7].

A recent key method to handle the aforementioned drawbacks is *neural enhancement* via super-resolution (SR) deep neural networks (DNNs) [8]. SR DNNs operate by processing a low-resolution, degraded image to automatically generate a high-quality, high-resolution output. This allows compact, low-quality content to be transmitted across the network, at the expense of additional computation at the receiver's end. As such, neural enhancement removes the system's sole reliance on the network and opens up a new dimension in the design space by introducing a trade-off between the use of bandwidth and computational resources [9], [10].

Despite the increasing processing capabilities of mobile devices, *on-device* execution of SR models still remains an active challenge due to their demanding workload. In particular, the number of multiply-add operations and memory capacity required even by mobile-tailored SR DNNs is *orders of magnitude larger* than the more common classification DNNs [11]. To counteract the excessive computational needs, existing systems *1)* rely on powerful platforms, such as assuming the availability of a desktop GPU client [12], [13], *2)* require the parallel use of all available processors (CPU, GPU, NPU) [11], *3)* leverage frame dependencies in order to cache previously enhanced results [14] or *4)* resort to cloud offloading [15]. As such, existing solutions are either restricted to high-end deployment settings [12], [13], thus not accommodating mobile devices, or incur additional issues as a by-product, such as thermal throttling [11], [16], [17] and a drastic drop in visual quality [14], [15].

To counteract these limitations and enable the use of SR DNNs on mobile devices, there has been an increased focus towards low-precision DNN execution on faster and more efficient processing units like NPUs [11], [18]. These units provide higher energy efficiency than CPUs and GPUs by omitting general-purpose hardware logic, increasing at the same time the availability of computational resources for other tasks by taking over the compute-intensive DNN execution. Despite the NPUs' demonstrated benefits for *classification* DNNs, executing SR models at lower precision often comes at the cost of degraded visual quality; as shown

- *Stylianos I. Venieris is with the Samsung AI Center, CB1 2JH Cambridge, U.K.*
- *Mario Almeida is with Rain Instant Pay.*
- *Royson Lee and Nicholas D. Lane are with the University of Cambridge, CB2 1TN Cambridge, U.K. and also with the Samsung AI Center, CB1 2JH Cambridge, U.K.*

Preprint: Under review.

in Fig. 1, upscaling with INT8 - *which is conventionally thought to be the best data type for inference* - would result in unnatural visual artifacts on both the texture and color in some images, especially for deeper DNNs. Notably, these anomalies could also happen despite marginal quantitative loss in standard metrics (0.1 dB drop in PSNR) as minor differences at the pixel level can still result in considerable high-level deformation. As a result, existing on-device SR frameworks such as MobiSR [11] and NEMO [14] underutilize or entirely avoid execution on the NPU to meet an acceptable visual quality. Thus, there is an emerging need for novel solutions that allow leveraging mobile NPUs for SR without the quality impact of low-precision data types.

In this work, we present NAWQ-SR, a framework that overcomes the limitations of existing on-device SR systems and delivers fast, efficient and high-quality SR on mobile. NAWQ-SR introduces an NPU-centric approach, comprising a novel *hybrid-precision execution* paradigm and a runtime *neural image codec* that exploit the multi-precision processing capabilities of modern mobile NPUs to minimize latency while meeting the user-specified quality targets. Moreover, to push visual quality beyond the state-of-the-art NPU-based designs, we propose a mechanism that selectively re-customizes the arithmetic precision of the DNN layers on-the-fly. This paper makes the following key contributions:

- A novel hybrid-precision execution scheme together with a methodology for optimizing the deployment of SR DNNs to the latency and quality requirements of the target application. By considering the multiple precisions supported by a given NPU, our framework adapts each layer's wordlength through a single-shot optimization algorithm that co-optimizes the per-layer quantization of the DNN and the scheduling of its layers on the NPU.
- A novel technique that selectively applies adaptive arithmetic precision on quantization-sensitive layers, enhancing them with wider representational power at run time. We dynamically adapt the quantization parameters of the selected layers in a per-sample input-dependent manner, leading to lower quantization error and higher visual quality than previously attainable on mobile NPUs.
- A new neural image codec comprising a hybrid-precision dispatcher and a runtime quantization unit. Through our low-overhead codec, we provide a *fully NPU-based* execution of SR DNNs that avoids barriers of current NPU support for upsampling layers, acknowledged by previous works, that conventionally required CPU or GPU fallback.
- To the best of our knowledge, this work is the first SR approach to exploit the multi-precision capabilities of the heterogeneous processing units that reside in NPUs. Hence, it can be orthogonally combined with existing on-device SR systems such as MobiSR [11] to counteract their performance limitations on the NPU. As a standalone framework, it delivers a speedup of 1.6×-9.8× over state-of-the-art on-device SR systems and 91% over XLSR, the winner of the Mobile AI 2021 challenge on real-time quantized SR.

2 Background & Related Work

In this section, we discuss the emerging use of super-resolution for efficient visual enhancement on mobile devices, model- and system-level optimizations for the on-device execution of SR models and the main characteristics of the latest mobile NPUs.

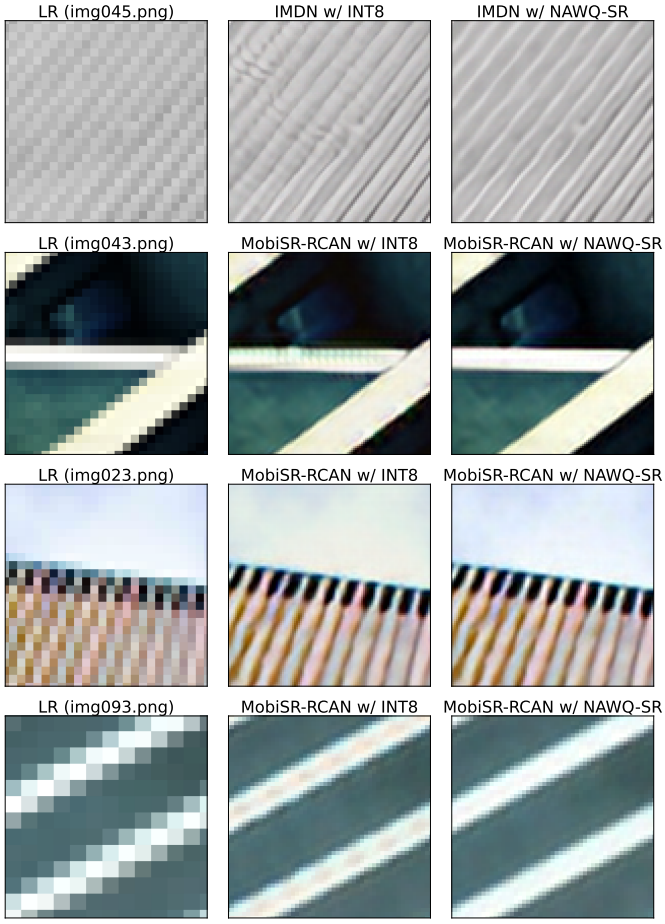


Fig. 1: Qualitative comparison between INT8 and NAWQ-SR $\times 4$ upscaling on the Urban100 [19] dataset. INT8 results in visual artifacts on both textures and colors when compared to NAWQ-SR's hybrid-precision and DRE-based quality enhancement.

2.1 Super-resolution for Mobile Devices

The unprecedented performance of SR DNNs in restoring realistic textures, together with their orthogonal integration with image/video compression and adaptive bitrate schemes, has made them a key component behind a broad range of products, from high-resolution TVs [20] to gaming GPUs [21]. As such, several works have focused on improving the quality of mapping low-resolution (LR) images to high resolution (HR) [22], [23], [24]. Despite the significant progress [25], [26], SR DNNs still have prohibitively high computational and memory demands for most real-world mobile deployments.

Efficient Super-resolution. Recent works have proposed efficiency-optimized model architectures. Prominent techniques span from avoiding the computation of large feature maps [8] and mitigating the cost of upsampling through the use of pixel-shuffle layers [27], [28], to employing more efficient blocks, such as group convolutions [29] and channel splitting [23], [30]. Neural architecture search for efficient SR is also gaining traction [31], [32], [33]. Nonetheless, the on-

device execution of these models is still impractical, resulting in numerous system-based solutions [10].

On-device Super-resolution. To deploy SR models on mobile, the state-of-the-art on-device SR frameworks have adopted various approaches. One line of work [11], [15] has focused on utilizing the heterogeneous processors (CPU, GPU, NPU) residing in many recent devices. To effectively load-balance across processors, these systems exploit the observation that patches of an image have varying upsampling difficulty. For instance, MobiSR [11] adopts a criterion to quantify the difficulty of each patch and dispatch it to the appropriate processor. Besides scheduling, the video-focused NEMO [14] leverages the inter-frame dependencies in order to cache and reuse previously super-resolved patches. Finally, SplitSR [34] combined efficient model design with compiler optimizations to improve CPU-based SR and XLSR [18] presented a hand-crafted lightweight model.

Even though these frameworks enable fast on-device upsampling, they come at the high cost of *quality degradation*. Notably, mapping these models on compute engines that run on lower bitwidths, such as NPUs, causes a considerable drop in visual quality as observed in recent mobile SR systems [11], [14], [18]. As a result, existing systems either reduce the number of patches dispatched to NPUs [11] or entirely avoid using them [14], [34], leading to reduced efficiency compared to NPU-only execution. As little work has been done to mitigate the effects of quantization on SR models, our work aims to breach this gap to allow existing techniques to leverage the full capabilities of modern NPUs that can be found across smartphones [35], [36], [37], [38].

Quantization. Precision quantization constitutes a prominent method for minimizing the computational and memory demands of DNNs. State-of-the-art approaches typically adopt block floating-point schemes (also known as dynamic fixed-point), using a *uniform* wordlength[1] across layers. The majority of existing works apply either *1)* quantization to already trained full-precision models, followed by a retraining step to fine-tune the weights [39], [40], or *2)* quantization-aware training to directly obtain low-precision models [41], [42]. As such, a commonality of both approaches is that they require an expensive training step.

A third approach that allows for *nonuniform* per-layer wordlength are mixed-precision schemes, such as HAQ [43] and HAWQ [44]. However, both HAQ and HAWQ impose an excessive computational overhead by relying on reinforcement learning and a multi-stage retraining process, respectively. More importantly, both are tailored for classification DNNs.

Although the aforementioned quantization approaches have been successfully applied on *classification* DNNs with minimal accuracy loss, they do not generalize to SR models, as they often lead to a catastrophic drop in visual quality [11], [14], [45], as shown in Fig. 1. This is primarily due to the removal of Batch Normalization (BN) layers from recent SR models [22], [30], [31] as they were shown to severely restrict their representational power [46]. In turn, the absence of BN leads to significant variability in the dynamic range of activations, making the direct utilization of existing quantization methods futile [41] or requiring expensive architectural modifications and retraining [45], [47], [48].

With the integration of low-precision NPUs in smartphones, there is an emerging need for novel quantization methods that are particularly crafted for on-device SR in order to combine high quality with efficiency. In this context, our NAWQ-SR framework introduces novel *post-training* techniques that closely approach the quality of full-precision models, leaving *little room for improvement* through expensive retraining. In addition, NAWQ-SR can be applied complementarily on models trained in a quantization-aware manner.

2.2 Challenges and Opportunities of NPUs

Designed explicitly for DNN workloads, mobile NPUs typically rely on low-precision processing units, employing 16- or 8-bit fixed-point arithmetic [35], [37]. Despite the potential processing benefits and although such narrow precision has been used effectively for classification DNNs [41], quantized SR models suffer excessive quality drops compared to their full-precision versions (Fig. 1), making NPU execution prohibitive.

Nonetheless, recent hardware advances have led to NPUs that support *multiple arithmetic precisions*. Such examples are Hexagon 698 on Qualcomm Snapdragon 865 (SDM865) [38], Arm Ethos [49] and MediaTek AI processing unit (APU) [50], all supporting two precision modes: 8-bit activations and weights (INT8) or 16-bit activations and 8-bit weights (A16W8). In spite of the new opportunities of these hardware architectures, existing deployment methods fail to exploit them, leading to *1)* fast but low-quality execution in INT8 - due to the quantization-induced error, *2)* higher quality but slow execution in A16W8 - close to 2× slower than INT8, as shown in §5.3), or *3)* slow *and* low-quality execution in A16W8 for models where even 16 bits do not suffice - which is often the case for SR models. Our work pushes the boundaries of what is possible in terms of mapping SR models to NPUs, yielding fast and high-quality designs that fully utilize their multi-precision capabilities.

3 NAWQ-SR Overview

Towards addressing the shortcomings of existing mobile SR systems, we propose NAWQ-SR, an NPU-centric framework that maximizes the efficiency of on-device SR. NAWQ-SR leverages the fact that different parts of SR neural architectures have nonuniform precision needs, in order to partition the execution across the NPU's heterogeneous units. With SR models deployed across a broad range of use-cases, NAWQ-SR is in a unique position to enhance the performance of a wide range of visual-content mobile applications.

Offline Flow. Fig. 2 shows NAWQ-SR's offline flow. The framework is supplied with a trained SR DNN and a quality drop tolerance using an image distortion metric. As a first step, the *Weights Quantizer* analyses the dynamic ranges of the model's weights in each layer and accordingly reduces their precision to 8 bits, using suitable scale factors. Next, the *Multi-Wordlength Quantizer* (§4.1) considers the NPU-supported bitwidths and determines the wordlength for the activations of each layer, allowing for different bitwidths

1. We use the terms *wordlength* and *bitwidth* interchangeably.

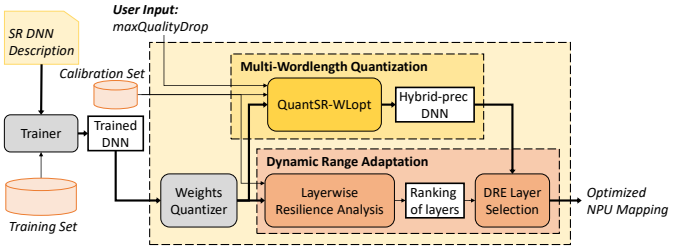


Fig. 2: Overview of NAWQ-SR's offline flow.

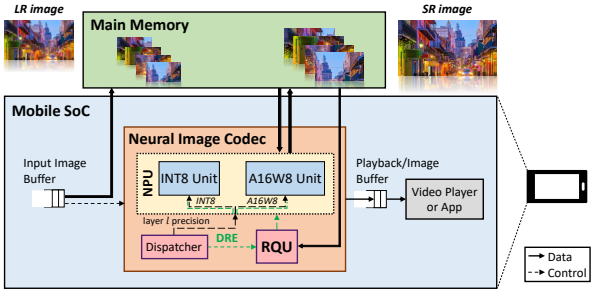


Fig. 3: NAWQ-SR's runtime architecture.

across layers. The output of this stage is a quantized *hybrid-precision* DNN. At this stage, the user-supplied calibration set is used to find the least computationally costly hybrid-precision DNN that meets the user's quality constraint.

As a next step, the weights-quantized DNN is passed to the *Dynamic Range Adaptation* module (§4.2). This module is responsible for deciding which layers will *not* use the quantization *scale factors* that the *Multi-Wordlength Quantizer* selected. Instead, these layers derive their scale factors *at run time* by examining the dynamic range of the input activations tensor and quantizing them on-the-fly. We refer to this technique as run-time *dynamic range estimation* (DRE) and determine the DRE layers using the *DRE Layer Selection* module based on a *Layerwise Resilience Analysis*, which assesses the resilience of each layer to low precision. Overall, given the user-defined quality drop tolerance, NAWQ-SR generates a *DRE-augmented hybrid-precision* model together with an *execution schedule*, tailored for the NPU of the target mobile device and content.

Runtime Architecture. Fig. 3 depicts the architecture of NAWQ-SR upon deployment. The process is triggered when LR images arrive at the *Input Image Buffer*. These are passed in a per-image manner to the *Neural Image Codec* (§4.3), which is responsible for their upscaling. The *Dispatcher*, already hosting the NAWQ-SR's hybrid-precision model and its associated execution schedule, schedules the processing of the input images on the NPU. As such, each layer is executed either on the INT8 or the A16W8 unit. If DRE is selected, the layer's input activations tensor is redirected to the *Runtime Quantization Unit (RQU)*, which in turn quantizes it based on its actual dynamic range and then feeds it to the appropriate unit. Finally, the processed images are passed to the *Playback/Image Buffer*.

4 Design of NAWQ-SR

In this section, we detail how NAWQ-SR leverages the heterogeneous processing units of mobile NPUs through hybrid-precision execution and formally define the optimization problem that jointly decides the quantization and mapping of DNN layers to the NPU resources. Moreover, we describe the runtime components of NAWQ-SR and the associated optimizations that ensure efficient and high-performance integration into commodity mobile devices.

4.1 Multiple Wordlengths for Mobile SR

Traditional mobile implementations of DNNs commonly employ a single uniform wordlength across all computations, with either floating-point arithmetic on CPUs and GPUs or fixed-point on DSPs and NPUs. This is a result of targeting pre-designed processing units, such as a CPU's FP32 or a DSP's INT8 units. Nevertheless, the latest NPUs can help us overcome this restriction for two reasons. At the hardware level, modern NPUs either host heterogeneous processing units that support different arithmetic precision, *e.g.* the 8-bit HVX and A16W8 HTA units on the Hexagon 698 NPU, or provide precision-configurable units, *e.g.* Samsung S21's NPU [51]. This property allows the optimization of the DNN execution so that different operations are performed using different precision. At the algorithmic level, we can design methodologies that allow the customization of each operation's precision, shaping the per-operation wordlength to the requirements of the DNN algorithm.

Together, these optimization opportunities point to an alternative design paradigm, which we name *hybrid-precision*. This implementation style introduces a multiple-wordlength approach and inherits the speed and energy advantages of fixed-point arithmetic. However, by allowing each operation in the DNN to be encoded with a different wordlength, the design degrees of freedom are significantly increased.

To comply with the widely adopted practice of applying 8-bit quantization on the weights of a model and with the NPU trend of supporting only 8-bit weights [38], [49], we quantize the weights using 8 bits across all layers (line 1 in Alg. 1 and Weights Quantizer in Fig. 2), and tailor our hybrid-precision method to the activations. We first define the *granularity* at which different wordlengths can be applied. In NAWQ-SR, we opt for a *layerwise* parametrization. This approach ensures the efficient utilization of the underlying hardware: the quantization step prior to execution has to be amortized across several computations, which is achieved by the compute-intensive convolution or matrix operations of a DNN layer. Finer granularity, such as allowing for different wordlength per channel, would incur significant overhead due to the low computation-to-quantization ratio.

Hybrid-Precision Quantization Strategy. To implement multi-wordlength DNNs, a hybrid-precision quantization strategy needs to be defined. The proposed strategy utilizes different wordlength b_l, scale factor s_l and zero point z_l for each layer l, such that a value x is quantized to a b-bit integer x_{quant} as $x_{\text{quant}} = \lfloor x \cdot s_l - z_l \rceil$. To introduce different wordlengths among layers, quantization is performed such that all values within each activations tensor at the input of each layer have a single wordlength, scale factor and zero point. As such, the quantization configuration, q_l, for the l-th layer is given by $q_l = \langle b_l, s_l, z_l \rangle \ \forall l \in \mathcal{L}$, where $\mathcal{L}$ is the set of layers in the given DNN. Furthermore, the scale factor s_l and zero point z_l are derived based on the estimated dynamic range of the activations tensor $\mathbf{x}$ as

$$s_l = \frac{(2^{b_l} - 1)}{\hat{\mathbf{x}}_{\max} - \hat{\mathbf{x}}_{\min}} \quad , \qquad z_l = \lfloor s_l \cdot \hat{\mathbf{x}}_{\min} \rceil \tag{1}$$

where $\widehat{\mathbf{x}}_{\{\max,\min\}}$ are estimates of the max/min values in $\mathbf{x}$, derived by processing a dataset that is representative of the target task. We refer to this set as the calibration set.

Hybrid-Precision Wordlength Optimization. Given a DNN m with $|\mathcal{L}|$ layers, we define a wordlength b_l for each layer l, referred to collectively as the vector $\mathbf{b}$. We further denote by $m(\mathbf{b})$ a model quantized with hybrid precision across its layers as dictated by $\mathbf{b}$. Let ϵ be the user-specified maximum allowable drop on average quality, which can be quantified using the peak signal-to-noise ratio (PSNR) image reconstruction metric, denoted by $\mathbb{E}(Q(m(\mathbf{b})))$. Given a cost estimator $T(m(\mathbf{b}))$ (*e.g.* latency estimate or FLOPs), we pose the following constrained optimization problem

$$\min_{\mathbf{b}} T(m(\mathbf{b})) \quad \text{subject to} \tag{2}$$
$$\forall l \in \mathcal{L} : b_l \in \mathcal{W} \quad \text{and} \quad \mathbb{E}\Big(Q\big(m(\mathbf{b})\big)\Big) - \mathbb{E}\Big(Q\big(m(\mathbf{u})\big)\Big) \leq \epsilon$$

where $\mathcal{W}$ is the candidate wordlength set and $\mathbf{u}$ is the uniform wordlength vector that assigns 32 bits to all layers. The scale factor s_l and zero point z_l are analytically derived as per Eq. (1) and hence are implicitly co-optimized with the selection of b_l. Thus, we omit them from Eq. (2).

The optimization considers the supported bitwidths of the underlying NPU (*e.g.* $\mathcal{W} = \{8, 16\}$ for SDM865) and aims to find the wordlengths and scale factors of all layers that minimize the execution cost of an SR DNN on the NPU, subject to the given quality constraints. To capture the execution cost on the specialized hardware of NPUs, we adopt a variation of the number of bit operations (BOPs) metric as our cost estimator T [42], [52]. Our metric weights each operation with a cost based on the number of bytes used. Specifically, operations performed in 32, 16, and 8 bits are assigned a cost of 4, 2 and 1, respectively, reflecting the runtime and memory differences among the different bitwidths. Hence, given a model m and a wordlength vector $\mathbf{b}$, GetBOPs($m(\mathbf{b})$) returns the total cost of executing m by considering each layer's number of operations and assigned wordlength (b_l).

The per-layer wordlength selection can be cast as a search problem aiming to achieve peak processing speed by selecting suitable bitwidths. For an SR DNN with $|\mathcal{L}|$ layers and $|\mathcal{W}|$ candidate bitwidths, the total number of candidate hybrid-precision configurations is $|\mathcal{W}|^{|\mathcal{L}|}$. With an increase in either the depth of a DNN or the number of available bitwidths, an exhaustive enumeration rapidly becomes intractable. In real-world deployments, although NPUs currently support up to two bitwidths, *e.g.* 8 or 16 bits, state-of-the-art SR DNNs reach significant depths, ranging from 33 layers for the lightweight TPSR model [31] and hence 8 billion design points, up to more than 1500 layers for RCAN [53] and 2^{1500} design points. As a result, the combinatorial scaling of the design space size and the large depth of SR DNNs prohibit optimization by means of enumeration.

QuantSR-WLopt. In this context, we propose QuantSR-WLopt, a heuristic method to obtain a solution in the non-convex design space. The key principle behind QuantSR-WLopt is a cost-prioritizing strategy that applies more aggressive quantization to the most FLOPs-heavy layers first, through an efficient single-shot wordlength adaptation, *i.e.* by modifying the wordlength of each layer only once.

With reference to Algorithm 1 and with a running example of $\mathcal{W} = \{8, 16\}$, QuantSR-WLopt first quantizes all layers with the same uniform high precision (*e.g.* 16 bits) (lines 1-3) and sorts them with respect to the amount of BOPs (lines 4-5). Next, the algorithm iterates *once* along the depth of the DNN and sets the wordlength of the l-th layer to 8 bits (line 8). By passing through the calibration set, the achieved quality q is calculated (line 10), together with the new cost (line 11). If the current quality satisfies the constraint, layer l is kept to 8 bits; else it is reverted back to 16 bits to recover the lost quality (lines 12-14).

Algorithm 1: Wordlength Optimization (**QuantSR-WLopt**)

Input: DNN m with layers $\mathcal{L}$, Wordlengths set $\mathcal{W} = \{8, 16\}$
Calibration set $\mathcal{D}_{\text{calib}}$
Reference quality q_{ref} (PSNR in dB or SSIM in $[-1, 1]$)
Quality drop tolerance ϵ
Output: Optimized wordlength vector $\mathbf{b}^{\text{sel}} \in \mathcal{W}^{|\mathcal{L}|}$

1 $m \leftarrow$ WeightsQuantizer(m, 8) ▷ Quantize weights to 8 bits
2 $\mathbf{u} \leftarrow$ uniform wordlength (in our case 16 bits)
3 $\mathbf{b}^{\text{sel}} \leftarrow \mathbf{u}$
4 InitScales&ZeroPoints($m(\mathbf{b})$, $\mathcal{D}_{\text{calib}}$)
5 $c^{\text{bops}}_{\text{total}}, \mathbf{c}^{\text{bops}}_{\text{layers}} \leftarrow$ GetBOPs($m(\mathbf{b})$) ▷ Initial cost
6 $\mathbf{c}^{\text{sorted}}_{\text{layers}}, \mathcal{L}^{\text{sorted}}_{\text{bops}} \leftarrow$ SortDescending($\mathbf{c}^{\text{bops}}_{\text{layers}}$)
7 **foreach** l **in** $\mathcal{L}^{\text{sorted}}_{\text{bops}}$ **do** ▷ Single-shot pass through the layers
8 $\quad \mathbf{b} \leftarrow \mathbf{b}^{\text{sel}}$
9 $\quad b_l \leftarrow 8$
10 $\quad$ UpdateScale&ZeroPoint(b_l) ▷ Using Eq. (1)
11 $\quad q \leftarrow$ GetQuality($m(\mathbf{b})$, $\mathcal{D}_{\text{calib}}$)
12 $\quad c^{\text{bops}} \leftarrow$ GetBOPs($m(\mathbf{b})$)
13 $\quad$ **if** $q_{\text{ref}} - q \leq \epsilon$ **then** ▷ Quality constraint
14 $\quad\quad \mathbf{b}^{\text{sel}}_l \leftarrow 8, \quad q_{\text{best}} \leftarrow q, \quad c^{\text{bops}}_{\text{min}} \leftarrow c^{\text{bops}}$
15 $\quad$ **end**
16 **end**

QuantSR-WLopt exhibits a number of crucial properties. With respect to complexity, it scales linearly with the number of layers $|\mathcal{L}|$ as each layer is examined only once. With respect to execution cost, QuantSR-WLopt's cost-aware criterion ensures that a less costly layer is never quantized to lower precision at the expense of a heavier layer. Hence, it prioritizes the quantization of layers that will have a larger impact on minimizing the runtime. With respect to quality, the algorithm guarantees by design the return of a configuration that meets the quality constraint, if and only if such a design exists in the design space. As such, the upper bound in quality is given by $m(\mathbf{b}^{\max})$ where $b^{\max}_l = \max(\mathcal{W})$ for all $l \in \mathcal{L}$. Thus, to address cases where the upper bound in quality is not satisfactory, we introduce a new design dimension in the quantization scheme by deciding whether to fix or dynamically determine the scale factor and zero point of each layer. We discuss this in the following section.

4.2 Dynamic Range Adaptation

As described in Section 4.1, the A16W8 mode constitutes our scheme's upper bound in attainable visual quality. However, there are cases where A16W8 fails to satisfy the constraint of Eq. (2). As such, current NPU mappings often fail to reach acceptable quality, especially when targeting efficient SR models. This has led to existing works either partially using the NPU [11] or avoiding it altogether [14], [34].

To push the quality of NPU-based SR beyond what was previously attainable, while sustaining the processing benefits of hybrid-precision execution, NAWQ-SR introduces a new design dimension to the quantization strategy, which we name *dynamic range estimation* (DRE). DRE adapts the scale factor and zero point of an activations tensor *at run time*, based on the *actual* range of its values for the *particular* input sample. This technique overcomes the limitations of existing

Algorithm 2: Layerwise Resilience Analysis (**LRA**)

Input: DNN m with layers $\mathcal{L}$, Wordlengths set $\mathcal{W} = \{8, 16\}$
Quality drop $q^{\text{drop}}_{\text{w-quant}}$ of 8-bit weights-quantized DNN
Reference quality q_{ref} (PSNR in dB or SSIM in $[-1, 1]$)
Output: Sorted layers with respect to quality drop $\mathcal{L}^{\text{sorted}}_{\text{drop}}$
Sorted layerwise quality drops $\mathbf{q}^{\text{drop}}_{\text{sorted}}$

1 $q_{\text{ref}} \leftarrow q_{\text{ref}} - q^{\text{drop}}_{\text{w-quant}}$ ▷ Remove quality drop due to 8-bit weights
2 $u \leftarrow$ uniform wordlength (in our case 16 bits)
3 **foreach** l **in** $\mathcal{L}$ **do** ▷ for each layer
4 $\quad \mathbf{b} \leftarrow \mathbf{u}$
5 $\quad b_l \leftarrow 8$ ▷ Set bitwidth for the l-th layer's activations to 8 bits
6 $\quad q \leftarrow \text{GetQuality}(m(\mathbf{b}))$
7 $\quad \mathbf{q}^{\text{drop}}_l \leftarrow q_{\text{ref}} - q$
8 **end**
9 $\mathbf{q}^{\text{drop}}_{\text{sorted}}, \mathcal{L}^{\text{sorted}}_{\text{drop}} \leftarrow \text{SortDescending}(\mathbf{q}^{\text{drop}})$

Algorithm 3: DRE Layer Selection

Input: Hybrid-precision DNN m_q with layers $\mathcal{L}$
Sorted layers with respect to quality drop $\mathcal{L}^{\text{sorted}}_{\text{drop}}$
Sorted layerwise quality drops $\mathbf{q}^{\text{drop}}_{\text{sorted}}$
Energy concentration threshold $K \in [0, 1]$
Output: DRE-augmented quantized model m^{DRE}_q

1 **for** $l \leftarrow 0$ **to** $|\mathcal{L}| - 1$ **do** ▷ loop through sorted layers
2 $\quad \mathbf{E}_l \leftarrow \sum_{i=0}^{l} |\mathbf{q}^{\text{drop}}_{\text{sorted},l}|^2$ ▷ Energy concentration up to layer l
3 **end**
4 **for** $l \leftarrow 0$ **to** $|\mathcal{L}| - 1$ **do** ▷ loop through sorted layers
5 $\quad$ **if** $\mathbf{E}_l$ / $\mathbf{E}_{|\mathcal{L}|-1} \leq K$ **then** ▷ Energy constraint
6 $\quad\quad \mathcal{L}_{\text{DRE}} \leftarrow \text{Append}(\mathcal{L}^{\text{sorted}}_{\text{drop}}(l))$
7 $\quad$ **end**
8 **end**
9 $m^{\text{DRE}}_q \leftarrow \text{AddDRE}(m, \mathcal{L}_{\text{DRE}})$ ▷ Use DRE on the selected layers

works, where the values of s_l and z_l are statically derived prior to deployment and remain fixed at run time. The primary limitation that leads to degraded output quality is manifested in cases where the estimated dynamic range does not capture the actual encountered range of an input. In these cases, the statically determined precision underutilizes the representation range of the selected wordlength, leading to excessive numerical error and, in turn, quality drop. Instead, DRE adapts the scale factor and zero point in an input-dependent manner, occupying the full range of values for the activations of the current input.

With this scheme, we formulate the new quantization method for each layer as $q_l = \langle b_l, s_l, z_l, d_l \rangle \ \forall l \in \mathcal{L}$, where $d_l \in \{0, 1\}$ indicates whether DRE is applied on layer l. When d_l is 1 and DRE is enabled, the actual dynamic range of the input activations tensor $\mathbf{x}$ is first calculated and the scale factor s_l and zero point z_l are derived on-the-fly as per Eq. (1), by substituting the statically determined estimates at the denominator with the actual values, $\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$.

The advantages of DRE come at a cost: the computational overhead of finding the actual range (*i.e.* min/max values) of the activations tensor and computing the new scale factor and zero point has to be taken into account. In other words, applying DRE across all layers in a brute-force manner can lead to excessive latency and thus negate its benefits. Hence, to effectively utilize DRE, we have to devise a method of: *i)* quantifying the resilience of each layer to low precision, and *ii)* an algorithm that leverages this information to *selectively* apply DRE to a subset of the DNN's layers.

Layerwise Resilience Analysis. Algorithm 2 presents our technique for estimating each layer's resilience to reduced precision. The core idea behind LRA is to isolate each layer's contribution to the quality drop of a quantized model. As the weights are already 8 bits (§4.1), we first subtract the PSNR drop caused solely by the weights quantization (line 1). In this manner, any subsequently observed PSNR degradation is due to the activations quantization. The algorithm starts by using a uniform higher-precision representation for the activations of all layers (line 2). Next, we iterate through the layers, quantizing each one *individually* to 8 bits and obtaining the associated drop with respect to that of the weight-quantized model (line 7). Finally, the layers are sorted in a decreasing order of quality drop (line 8).

DRE Layer Selection. After selecting the highest performing bitwidths via QuantSR-WLopt and estimating the layerwise resilience to quantization through LRA, NAWQ-SR picks a subset of layers, to have their scale factors and zero points computed at run time based on their actual dynamic range. Algorithm 3 describes this layer selection process. The objective of the algorithm is to recover the visual quality for the layers which exhibit large quality degradation when quantized. Our key insight is to interpret the layerwise PSNR drop as a discrete signal and adopt the respective signal energy [54] (line 2) as a criterion to tune the amount of layers that will utilize DRE. Given the DNN layers ordered by quality drop, the DRE layer selection algorithm first calculates the energy concentration up to each of these layers (lines 1-3). For instance, the energy concentration of a layer l includes the energy concentration of the previous ordered layers (0 to l-1). Next, the algorithm selects for DRE all the layers until the first one that meets the requested energy concentration threshold K (lines 4-7). Threshold K is represented as a fraction of the total energy concentration ($K \in [0, 1]$) and allows for enhancing quality at the expense of the extra DRE-induced latency (quantified in §5.2). A key property of our method is that the number and selection of layers that use DRE *do not require tuning*; instead, they are adapted automatically based on K and can be nonuniform across different SR DNNs for the same value of K.

4.3 Neural Image Codec

The Neural Image Codec is responsible for dividing the LR images into fixed-size patches and upscaling them using the target SR DNN through an optimized NPU mapping.

Dispatcher. To guide the on-device execution, the Neural Image Codec introduces a dispatcher that, given the per-layer quantization configuration q_l, schedules execution to the appropriate hardware processor of the NPU, using the specified bitwidth, scale factor and zero point. To ensure efficient execution, this process is performed in a number of steps. First, the dispatcher adopts a partitioning strategy to reduce the communication between the codec components and the target processors. Specifically, the dispatcher partitions the DNN into groups of consecutive layers based on their target bitwidth (*e.g.* INT8 or A16W8) and range estimation technique (d_l), scheduling execution on a per-partition basis. As such, the scheduling of consecutive layers that need to interact with the same components is coalesced, amortizing the cost of communication between components.

Second, the dispatcher considers the requested range estimation technique (d_l). Partitions without DRE can be executed without additional supervision using the supplied scale factors and zero points. The remaining partitions are monitored by the RQU to adjust the per-layer scaling factors and zero points at run time, as detailed in the next section.

Finally, the dispatcher coordinates with the NPU executor to perform inference on a target processor (*e.g.* either HVX or HTA in SDM865's NPU) that supports the requested partition's bitwidth. We note that while the DNN partitions are represented with distinct bitwidths, their weights are always in 8 bits and, hence, only activations are quantized on-the-fly. As such, NAWQ-SR shares the weights between the activation wordlength representations by storing a single 8-bit copy in memory and thus incurs no extra memory cost for supporting both INT8 and A16W8.

Many commercial NPUs already provide either dedicated processors or extra cores for orchestrating execution where NAWQ-SR's dispatcher can be integrated. Such instances are the Q6 processor in QC's AI processor [38], or the NPU controller (NPUC) in the latest Samsung Exynos chipsets [51], [55]. By executing on a separate processor, NAWQ-SR's dispatcher and the partitioned inference can be performed in parallel in a pipelined fashion, thus sustaining high utilization of the NPU resources, while requiring no access to the resources of the main CPU and improving the overall efficiency.

Runtime Quantization Unit. For the partitions that require DRE, the RQU is responsible for estimating the per-layer dynamic range and adapting the respective scale factors and zero points at run time. To derive the new scale and zero point values (as detailed in §4.2), the RQU captures each layer's input tensors and extracts their range of values (*i.e.* $\mathbf{x}_{\text{min}}$ and $\mathbf{x}_{\text{max}}$). Then, the unit proceeds with the computation of the new scale factor and zero point as dictated by Eq. (1). The layer's inputs are then quantized using the new computed parameters and fed to the appropriate processing unit for the actual layer execution.

To be deployable without starving the resources of the target mobile device, the RQU has to exhibit low resource usage when invoked. To this end, we first vectorize the max/min operations by dividing the input activations tensor across parallel max/min search tasks and then apply a parallel-reduce operation to obtain the final range. Moreover, the RQU execution is placed on the same processing unit as the layers' partition at hand, to avoid unnecessary data transfers. Overall, the use of DRE results in improved quality with minimal overhead as shown in Section 5.2.

Memory-Aware Mapping of Upsampling. Modern state-of-the-art SR DNNs employ pixel-shuffle [27] for upsampling to the desired resolution. However, due to the limited cache of NPUs [56], [57], [58], [59] and pixel-shuffle's excessive memory demands, these layers cannot be directly mapped to NPU, leading to runtime errors [59], substitution with less performant blocks [14] or expensive fallback to CPU-based execution. This may be primarily attributed to the 6-dimensional intermediate data of the pixel-shuffle operation, which, if not manipulated efficiently, significantly affect the memory footprint. It is often the case that the NPU executor attempts to partition the tensor by storing each dimension on a separate memory bank, to provide the processing units with parallel access to all dimensions [59]. Hence, in cases where the tensor dimensions exceed the number of NPU memory banks or the depth of the banks is severely underutilized, the NPU can run out of memory.

To address this problem, we introduce a data layout transformation that caps and minimizes the footprint of pixel-shuffle-based upsampling. Our approach restructures the input and activation tensors so that a maximum of four dimensions are used throughout the pixel-shuffling process.

The original pixel-shuffle operation with an upscale factor of s on a tensor $\mathbf{x} \in \mathbb{R}^{1\times c_{\text{in}}\times h\times w}$ with c_{in} channels, height h and width w involves the following steps:

1) `Reshape` 4D tensor $\mathbf{x}$ into a 6D tensor of shape: $1\times c_{\text{out}}\times s\times s\times h\times w$
2) `Permute` dimensions as: $1\times c_{\text{out}}\times h\times s\times w\times s$
3) `Reshape` 6D tensor into final 4D tensor of shape: $1\times c_{\text{out}}\times s\cdot h\times s\cdot w$

This implementation leads to underutilization of the NPU memory. Instead, we perform the following steps:

1) `Reshape` 4D tensor into a 2D tensor of shape: $c_{\text{out}}\times s\cdot s\cdot h\cdot w$
2) Extract each of the c_{out} channels in parallel, producing c_{out} 1D tensors of size: $s\cdot s\cdot h\cdot w$
3) `Reshape` each of the c_{out} 1D tensors to a 4D tensor of shape: $s\times s\times h\times w$
4) `Permute` each of the c_{out} 4D tensors as $h\times s\times w\times s$
5) `Reshape` each of the c_{out} 4D tensors to 2D tensor of shape: $s\cdot h\times s\cdot w$
6) `Stack` c_{out} 2D tensors to form a single 3D tensor of shape: $c_{\text{out}}\times s\cdot h\times s\cdot w$

In this manner, we never exceed 4D tensors and the memory of the NPU is more fully utilized, enabling the mapping of upsampling layers on the NPU. This technique was crucial in order to enable the full NPU-based execution of SR DNNs and avoid the costly CPU fallback of current deployments.

5 Evaluation

Experimental Setup. We target the Qualcomm Snapdragon 865 SoC (SDM865) of a Samsung Galaxy S20. SDM865 comprises an octa-core Kryo 585 CPU, an Adreno 650 GPU and the Hexagon 698 NPU. The NPU integrates a vector processor (HVX) supporting INT8 and a tensor accelerator (HTA) supporting both INT8 and A16W8. We consider $\mathcal{W} = \{8, 16\}$ as our activations wordlengths and map INT8 to HVX and A16W8 to HTA. We implemented the NAWQ-SR's offline components using PyTorch (v1.6) and the runtime components by leveraging the Snapdragon Neural Processing Engine (SNPE v1.47) SDK. To showcase the generality of our system, we further target the Snapdragon 888 SoC (SDM888) and present comparisons against highly optimized baselines in terms of processing speed (§ 5.3) and energy efficiency (§ 5.5). SDM888 is hosted on a Snapdragon 888 Mobile Hardware Development Kit (HDK) and consists of an octa-core Kryo 685 CPU, an Adreno 660 GPU and the Hexagon 780 NPU. The NPU comprises scalar, vector and tensor processing units, which are composable and support both INT8 and A16W8. Unless mentioned otherwise, SDM865 is used for measurements.

SR Models. We target three state-of-the-art models of varying depth, architecture and workload: the lightweight TPSR [31], the mid-range IMDN [30], and MobiSR-RCAN [11], an efficient RCAN [22] variant.

Training Details For TPSR and IMDN, we utilize the pre-trained models as provided by the respective authors. For MobiSR-RCAN, we follow the training scheme by Lee *et al.* [11] and reproduce the reported results. Following the common practice of both the SR [18], [22], [23], [30], [46], [60] and mobile [11], [15], [34], [61] communities, all models were trained on DIV2K [62], consisting of 800 diverse-content images of 2K resolution. Unless otherwise mentioned, we use an upscaling factor of $\times 4$ to compare with previous works.

Performance Metrics. We report both visual quality and latency as evaluation metrics. In order to compare with other works, we use the standard SR reconstruction quality

TABLE 1: QuantSR-WLopt vs. Heuristic Optimizers

Model	Layers	Dataset	Target PSNR Drop	Search Time	BOPs Reduction WLopt	SA	GA
TPSR	33	B100	0.1 dB	2.1 min	1.96×	1.68×	1.59×
TPSR	33	Urban100	0.1 dB	9.6 min	1.83×	1.37×	1.69×
IMDN	85	B100	0.1 dB	9.4 min	1.93×	1.66×	1.44×
IMDN	85	Urban100	0.1 dB	22 min	1.93×	1.67×	1.57×
MobiSR-RCAN	255	B100	0.1 dB	72 min	2.00×	1.72×	1.56×
MobiSR-RCAN	255	Urban100	0.1 dB	177 min	2.00×	1.49×	1.50×

*SA setup: init. temperature t_0=1, cooling schedule at iter i: t_i=$t_0 e^{-0.05i}$
GA setup: population size=$\lceil 0.25 \cdot$ #Layers$\rceil$

metrics: PSNR and structural similarity (SSIM) [63]. We also note that a gain in these quality metrics do not necessarily translate to more visually pleasing images [64] and would like to emphasize that seemingly minimal gains of 0.1 dB can help counteract undesirable artifacts which occur due to quantization. As such, we present qualitative results in Fig. 1. For processing speed, we report the average latency across 50 runs, with the latency measurements obtained through SNPE's timing utilities. Unless mentioned otherwise, we assume a target high-resolution of 720p.

Datasets. The evaluation was conducted on the standard SR benchmarks used across a large body of recent mobile SR works [11], [15], [18], [34], namely Set5 [65], Set14 [66], B100 [67], and Urban100 [19]. Set5 and Set14 are smaller datasets with 5 and 14 images, respectively, each with different SR challenges. B100 and Urban100, with 100 images each, represent a wider range of natural and urban scenes which might be more representative of SR tasks in the wild. For each benchmark dataset, we use 10% as our calibration set, sampled randomly with uniform probability. Note that while our calibration set selection performs quite well, further exploration of the optimal calibration set size for each model and dataset can be performed [40].

NAWQ-SR Parameters NAWQ-SR exposes two parameters used for the exploration of the per-layer wordlengths and for the DRE layer selection – the quality drop tolerance (ϵ) and the energy concentration threshold (K), respectively. Unless mentioned otherwise, we use a tolerance ϵ of 0.1. For the model-dataset pairs where weights quantization (FP32W8 in Table 2) leads to $\geq$ 0.1 dB PSNR drop with respect to the original model (FP32), the tolerance ϵ is considered with respect to FP32W8 (bold values in the table).[2] For the energy concentration threshold, we tune the value of K via grid search for each model-dataset pair. As such, K was set to 0.125, 0.5 and 1.0, for IMDN, TPSR and MobiSR-RCAN, respectively.

5.1 Evaluation of Wordlength Optimizer

We compare QuantSR-WLopt with three heuristic optimizers: *1)* simulated annealing (SA) [68], *2)* genetic algorithm (GA) [69] and *3)* random search (RS). We compare the achieved BOPs reduction with respect to A16W8 given a PSNR drop constraint of 0.1 dB *under the same search time budget*, across the evaluated SR DNNs and datasets B100 and Urban100. We utilize the runtime of QuantSR-WLopt as the search time budget and run each of the baselines 10 times on an Nvidia GTX1080Ti GPU, reporting the average best result in Table 1. First, as the attainable BOPs reduction over A16W8 is bounded to a maximum of 2×, corresponding to INT8, we observe that our achieved reductions are very close to the peak performance, leaving little room for further improvement. Furthermore, QuantSR-WLopt consistently outperforms all baselines, yielding a BOPs gain between 16%-33% (21.8% geo. mean) over SA and 8%-34% (24.7% geo. mean) over GA. Finally, RS yielded designs that violated the PSNR constraint in the vast majority of runs and hence we omit it from Table 1.

All three baseline optimizers are iterative and can quickly determine the next candidate design point to evaluate. As such, these strategies would be suitable in cases where the objective function (BOPs and PSNR in our setting) is cheap to evaluate. Nevertheless, as PSNR is costly to evaluate and the design space is combinatorially large, the more structured search approach of our QuantSR-WLopt is more effective in yielding a hybrid-precision design that lies close to the theoretical maximum of 2× BOPs reduction.

2. FP32W8's drop can be reduced further via more sophisticated weight-quantization methods and thus is orthogonal to this work.

TABLE 2: Quality Comparison with Baselines (×4 Upscaling)

Model Variant	Average PSNR/SSIM Set5	Set14	B100	Urban100
TPSR - (Depth=33, Params=61K)				
FP32	31.10/0.8779	27.95/0.7663	**27.15**/0.7214	**24.97**/0.7456
FP32W8	**30.92**/0.8737	**27.85**/0.7634	27.08/0.7190	24.90/0.7423
FP16	31.10/0.8779	27.95/0.7663	27.15/0.7214	24.97/0.7456
INT8	30.75/0.8669	27.74/0.7573	26.99/0.7136	24.82/0.7362
A16W8	30.91/0.8736	27.83/0.7630	27.07/0.7189	24.88/0.7417
NAWQ-SR	30.91/0.8730	27.83/0.7620	27.05/0.7170	24.88/0.7411
IMDN - (Depth=85, Params=698K)				
FP32	32.21/0.8948	28.58/0.7811	**27.55**/0.7351	26.04/0.7837
FP32W8	**32.04**/0.8921	**28.46**/0.7795	27.47/0.7338	**25.92**/0.7814
FP16	32.21/0.8948	28.56/0.7809	27.52/0.7333	26.04/0.7837
INT8	31.86/0.8865	28.31/0.7749	27.35/0.7295	25.80/0.7753
A16W8	31.96/0.8913	28.38/0.7788	27.41/0.7336	25.85/0.7795
NAWQ-SR	32.01/0.8911	28.47/0.7781	27.45/0.7325	25.89/0.7787
MobiSR-RCAN - (Depth=255, Params=148K)				
FP32	**31.73**/0.8873	28.23/0.7729	**27.33**/0.7283	**25.34**/0.7615
FP32W8	31.71/0.8865	**27.82**/0.7726	27.31/0.7282	25.33/0.7611
FP16	31.73/0.8873	28.23/0.7729	27.32/0.7283	25.34/0.7615
INT8	31.03/0.8793	27.76/0.7651	27.02/0.7225	24.97/0.7499
A16W8	31.10/0.8813	27.80/0.7668	27.06/0.7244	24.99/0.7517
NAWQ-SR	31.69/0.8851	28.14/0.7696	27.27/0.7255	25.24/0.7557

*Bold indicates the designs whose quality defines NAWQ-SR's PSNR drop constraint.

5.2 Evaluation of Neural Image Codec

Runtime Overheads. To evaluate the overhead of estimating new scale factors and zero points for each of the selected DRE layers, we measured the inference time, across 50 inferences, for each of the models with and without DRE enabled for these layers. Overall, across all DNNs, the average time overhead of DRE was 4.26% (up to 6.40%) and 1.53% (up to 4.58%) for B100 and Urban100, respectively.

Another potential overhead introduced by NAWQ-SR is the cost of switching between partitions with distinct bitwidths (*i.e.* INT8 vs. A16W8). To evaluate this, we measured the switching times across 50 inferences for each of the DNNs, using the partitions selected by NAWQ-SR. The average partition-switching overhead over the inference time across DNNs was 0.34% (up to 0.84%) and 1.04% (up to 2.41%), for B100 and Urban100, respectively, with an average latency overhead of 39.25μs (up to 53μs) per partition.

Ablation Study of LRA and DRE. We conduct an ablation study on *i)* our LRA-based layer selection and *ii)* using DRE altogether, in order to disentangle their impact on the achieved quality. For each model in Table 3, we show the achieved PSNR/SSIM for the following configurations:

TABLE 3: Ablation Study on LRA-based layer selection and using DRE altogether.

Model Variant	LRA	DRE	Average PSNR/SSIM Set5	Set14	B100	Urban100
TPSR - (Depth=33, Params=61K)						
w/o DRE	✗	✗	30.89/0.8725	27.81/0.7614	27.04/0.7166	24.87/0.7407
RandDRE	✗	✓	30.89/0.8726	27.81/0.7615	27.05/0.7169	24.87/0.7408
NAWQ-SR	✓	✓	30.91/0.8730	27.83/0.7620	27.05/0.7170	24.88/0.7411
IMDN - (Depth=85, Params=698K)						
w/o DRE	✗	✗	31.94/0.8900	28.36/0.7775	27.38/0.7317	25.83/0.7776
RandDRE	✗	✓	31.97/0.8903	28.38/0.7773	27.40/0.7318	25.83/0.7774
NAWQ-SR	✓	✓	32.01/0.8911	28.47/0.7781	27.45/0.7325	25.89/0.7787
MobiSR-RCAN - (Depth=255, Params=148K)						
w/o DRE	✗	✗	31.07/0.8803	27.76/0.7652	27.03/0.7227	24.97/0.7499
RandDRE	✗	✓	31.18/0.8811	27.86/0.7663	27.09/0.7234	25.12/0.7537
NAWQ-SR	✓	✓	31.69/0.8851	28.14/0.7696	27.27/0.7255	25.24/0.7557

i) w/o DRE, where we use NAWQ-SR's selected bitwidths for each layer, but the scale factors and zero points of each activations tensor are derived *a priori* based on the maximum range encountered in the calibration set and remain fixed during deployment; *ii)* RandDRE, where we use *1)* NAWQ-SR's selected bitwidths for each layer, and uniform probability to *2)* randomly select the number of DRE layers and then *3)* randomly select the layers. For RandDRE, we report the average quality across 10 runs; and *iii)* NAWQ-SR, our method that selectively applies DRE using our LRA-based layer selection scheme.

Across all models and datasets, we observe that although RandDRE already provides quality gains over w/o DRE, the informed layer selection of the complete NAWQ-SR contributes significant additional gains. Specifically, DRE with LRA yield similar or higher quality, with gains of up to 0.02 dB (0.015 dB average) for TPSR, 0.11 dB (0.08 dB average) for IMDN and 0.62 dB (0.38 dB average) for MobiSR-RCAN. Notably, the gains of DRE are higher for deeper models as these models are more affected by the accumulation of quantization errors across layers, resulting in a larger drop in visual quality. As DRE significantly reduces the degree of error accumulation, it results in significant qualitative improvements in IMDN and MobiSR-RCAN: specifically, the mitigation of undesirable quantization artifacts on both texture and color as shown in Fig. 1.

From a computational perspective, RandRE often picks a suboptimal set of layers, resulting in 19.5× average higher overhead compared to NAWQ-SR's DRE layer selection. Instead, our LRA-based approach offers the advantage of determining in a single step both the number and the DRE layers that have the highest impact on quality. As a result, although a naive application of DRE can still yield a performance improvement, our more selective layer selection method achieves a better trade-off that combines both higher quality and lower latency overhead, and is, thus, an essential component of the proposed system.

Overall, as shown in Fig. 4 and Table 2, the Neural Image Codec presents a very reasonable overhead considering its latency and visual quality when compared to the full- (FP32) and lowest-precision (INT8) baselines.

5.3 Comparison with Highly Optimized Baselines

This section presents a comparison of NAWQ-SR with the following: FP32-CPU, FP16-GPU, INT8-NPU and A16W8-NPU designs, obtained through SNPE. These represent status-quo implementations that have been highly optimized using the SNPE compiler targeting each of the available processors. In the case of INT8-NPU, we allow the layers to be executed on both HVX and HTA to obtain the fastest execution. Table 2 presents the achieved quality and Fig. 4 and Table 4 depict the achieved speedup measured on SDM865 and SDM888 across models and datasets. We also report the quality after quantizing only the weights (FP32W8).

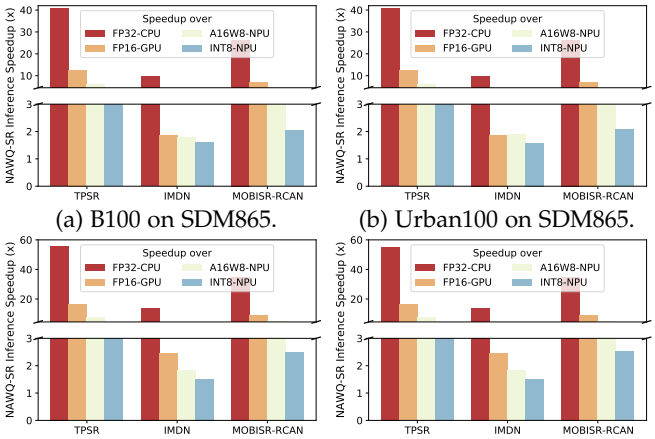
TABLE 4: Speedup over Highly Optimized Baselines

Model	Baseline	Speedup on SDM865 B100/Urban100	Speedup on SDM888 B100/Urban100
TPSR	FP32-CPU	40.89×/40.80×	55.55×/55.44×
	FP16-GPU	12.54×/12.51×	16.70×/16.66×
	A16W8-NPU	6.08×/ 6.07×	8.08×/ 8.06×
	INT8-NPU	3.65×/ 3.64×	4.61×/ 4.60×
IMDN	FP32-CPU	9.97×/ 9.91×	13.69×/13.62×
	FP16-GPU	1.88×/ 1.87×	2.46×/ 2.44×
	A16W8-NPU	1.89×/ 1.88×	1.83×/ 1.82×
	INT8-NPU	1.59×/ 1.58×	1.52×/ 1.51×
MOBISR-RCAN	FP32-CPU	26.11×/26.47×	34.59×/35.02×
	FP16-GPU	7.04×/ 7.14×	9.33×/9.45×
	A16W8-NPU	3.87×/ 3.92×	4.80×/ 4.86×
	INT8-NPU	2.04×/ 2.07×	2.49×/ 2.52×
Average (geo. mean)	FP32-CPU	25.69× (22.02×)	34.65× (29.77×)
	FP16-GPU	7.16× (5.51×)	9.51× (7.27×)
	A16W8-NPU	3.95× (3.55×)	4.91× (4.14×)
	INT8-NPU	2.43× (2.28×)	2.87× (2.59×)

Comparison to CPU/GPU Designs. With respect to the floating-point designs (FP32/FP16), NAWQ-SR delivers quality within 0.1 dB of the original model's for the vast majority of cases. In cases where weights quantization has a significant impact on quality, *e.g.* FP32W8 leads to $\geq$0.1 dB drop over FP32 for Set5, Set14 and Urban100 in IMDN, our framework was optimized with a 0.1 dB tolerance with respect to FP32W8. This is achieved across all cases. With respect to latency, NAWQ-SR outperforms both CPU and GPU designs by up to 40.8× (22× geo. mean across models and datasets) and 12.5× (5.5× geo. mean), respectively, on SDM865 and by up to 55.5× (29.7× geo. mean) and 13.6× (7.2× geo. mean), respectively, on SDM888.

Comparison to NPU Designs. With respect to A16W8-NPU, NAWQ-SR outperforms its PSNR for IMDN and MobiSR-RCAN with an average gain of 0.05 dB for IMDN and 0.35 dB for MobiSR-RCAN across datasets. For TPSR, NAWQ-SR generates mappings that either have slightly lower PSNR but still lie within the PSNR constraint with respect to FP32 (see B100), or meet the PSNR of A16W8-NPU. On the latency front, NAWQ-SR provides up to 6× and 8× faster execution than A16W8-NPU on SDM865 and SDM888, respectively, with a geometric mean of 3.55× and 4.14× on the respective device across models and datasets. Compared to INT8-NPU, NAWQ-SR yields higher PSNR with an average of 0.09 dB for TPSR, 0.12 dB for IMDN and 0.39 dB for MobiSR-RCAN across the datasets. With respect to latency, our system achieves up to 3.65× and 4.61× faster processing than INT8-NPU on SDM854 and SDM888, respectively, with a geometric mean of 2.28× and 2.59× on each device across models and datasets. NAWQ-SR's speedup is attributed to our highly optimized memory-aware mapping of the pixel-shuffle upsampling layers (§ 4.3) which enables the uninterrupted execution of the SR DNNs on the NPU, without falling back to CPU or GPU.

Overall, the results demonstrate how the hybrid-precision approach and the better utilization of the NPU's capabilities provided by our system allow us to closely track the quality of floating-point execution, outperform current INT8 designs, while pushing beyond A16W8's quality in several cases.



(a) B100 on SDM865. (b) Urban100 on SDM865.

(c) B100 on SDM888. (d) Urban100 on SDM888.

Fig. 4: NAWQ-SR's inference speedup over highly optimized baselines across SR DNNs, targeting SDM865 and SDM888.

5.4 Comparison with the state-of-the-art On-Device SR Systems

Here, we show the performance gains of NAWQ-SR as a standalone framework over the state-of-the-art on-device SR systems, MobiSR [11] and SplitSR [34], and the winning model of the 2021 MAI challenge [70] on quantized SR on mobile NPUs, XLSR [18] (Table 5). MobiSR, SplitSR and XLSR constitute the state-of-the-art image SR systems using heterogeneous processors, CPU, and NPU, respectively. For fair comparisons, we reimplemented and ran MobiSR on the same device (SDM865). Both systems base their design on lightweight variants of RCAN [22].

Comparison with MobiSR. MobiSR employs two models that are parallelized across the heterogeneous processors of the target device. The computationally heavier model is run on the CPU and GPU and the lightweight one on the NPU. MobiSR's scheduler divides the input image into patches and feeds them to each model-processor pair based on their difficulty; more difficult-to-upscale patches are sent for rapid processing to the NPU and easier patches are directed to the CPU and GPU in a load-balancing manner. Lee *et al.* [11] present three system configurations, each optimized for a different objective:

- **MobiSR-accuracy**: The accuracy-optimized model pair, denoted by $(m_{\text{ref}} + m_{\text{clc}})$ in [11]. m_{ref} denotes the original MobiSR-RCAN architecture. m_{clc} employs group convolutions and channel-shuffle layers [71], [72] to reduce the computational complexity of the original MobiSR-RCAN.
- **MobiSR-balanced**: The accuracy-latency balanced model pair, denoted by $(m_{\text{ref}} + m_{\text{s2}})$ in [11]. The compact model m_{s2} goes beyond the channel shuffling of m_{clc} and introduces channel splitting [73] and depthwise-separable convolutions [74] to further improve latency.
- **MobiSR-latency**: The latency-optimized model pair, denoted by $(m_{\text{clc}} + m_{\text{s2}})$ in [11]. This model pair combines the complexity-reduction techniques of the high-accuracy and balanced model pairs, delivering fast processing at the expense of degraded visual quality.

Furthermore, MobiSR introduces a parameter named total-variation (TV) threshold that tunes the accuracy-latency trade-off of each pair of models. To perform a fair comparison, we tune the TV threshold of each MobiSR variant, so that it meets 0.1 dB PSNR drop with respect to the original MobiSR-RCAN. As such, we set TV to $\langle 8, 8, 6, 6 \rangle$ for Set5, Set14, B100 and Urban100 for MobiSR-accuracy, $\langle 8, 8, 6, 8 \rangle$ for MobiSR-balanced and to 10 for all datasets for MobiSR-latency. Accordingly, we apply NAWQ-SR over MobiSR-RCAN with the same PSNR drop tolerance.

TABLE 5: Comparison with Existing On-Device SR Systems

System	Model	Memory (KB)	Average PSNR/SSIM Set5	Set14	B100	Urban100
Original	MobiSR-RCAN	594	31.73/0.8873	28.23/0.7729	27.33/0.7283	25.34/0.7615
MobiSR	(accuracy)	623	31.37/0.8787	28.10/0.7707	27.28/0.7258	25.28/0.7591
MobiSR	(balanced)	610	30.89/0.8590	27.98/0.7650	27.23/0.7207	25.31/0.7598
MobiSR	(latency)	134	31.05/0.8762	27.87/0.7640	27.11/0.7208	24.85/0.7415
NAWQ-SR	MobiSR-RCAN	148	31.69/0.8851	28.14/0.7696	27.28/0.7261	25.25/0.7558
SplitSR	(accuracy)	679	31.76/0.8982	28.29/0.7916	27.39/0.7491	25.46/0.7795
NAWQ-SR	IMDN	698	32.01/0.8911	28.47/0.7781	27.45/0.7325	25.89/0.7787
SplitSR	(latency)	367	31.53/0.8950	28.18/0.7887	27.28/0.7458	25.20/0.7704
NAWQ-SR	MobiSR-RCAN	148	31.69/0.8851	28.14/0.7696	27.28/0.7261	25.25/0.7558

System	Model	Upscale Factor	Precision	Memory (KB)	Average PSNR/SSIM B100	Urban100	Speedup over CPU	over XLSR
XLSR	-	×3	FP32	268	28.55/-	26.71/-	1.00×	1.00×
XLSR	-	×3	INT8	67	28.05-28.35/-	26.21-26.51/-	28.50×	1.00×
NAWQ-SR	TPSR	×3	hybrid	61	28.56/0.7861	26.78/0.8126	54.41×	1.91×

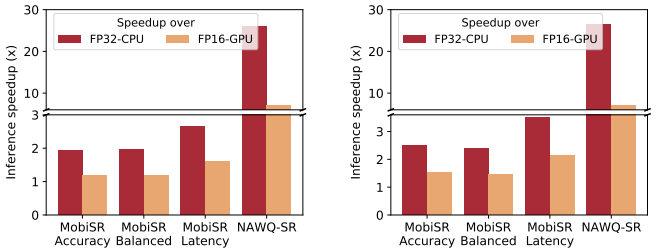
Fig. 5 depicts the actual speedup achieved by MobiSR and NAWQ-SR over highly optimized CPU and GPU implementations on Urban100. On B100, NAWQ-SR outperforms MobiSR yielding up to 13.4× and 5.9× higher speedup over the CPU and GPU mapping, respectively. Similarly, on Urban100, NAWQ-SR achieves up to 11.1× and 4.9× higher speedup over MobiSR compared to the CPU and GPU implementations, respectively. Due to its approach of quantizing the compact DNN that runs on the NPU, MobiSR has to compensate for the PSNR drop by scheduling a significant portion of patches to the expensive CPU- and GPU-pinned model. Instead, through the combination of hybrid-precision execution and DRE, NAWQ-SR alleviates the destructive effect of quantization on quality and enables the fast processing of all patches on the NPU. Overall, NAWQ-SR achieves an average speedup improvement of 7.93× (7.17× geo. mean) across models and datasets.

Comparison with SplitSR. SplitSR introduces a compact residual block (SplitSRBlock) and modifies RCAN to allow for a configurable accuracy-computational cost trade-off, using a single model. Two system configurations were presented in [34], optimized for different targets:

- **SplitSR-accuracy**: The accuracy-optimized model, composed of 7 residual groups, each with 7 residual blocks.
- **SplitSR-latency**: The latency-optimized model, composed of 5 residual groups, each with 6 residual blocks.

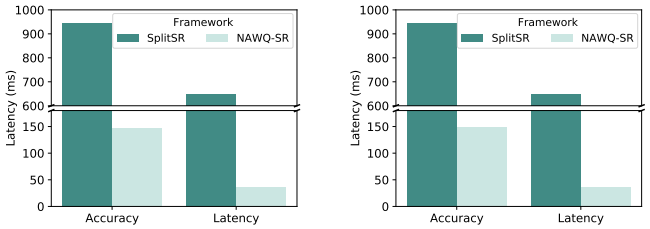
Moreover, SplitSR is optimized for mobile CPU execution through the TVM compiler [75]. To compare against SplitSR, we impose a PSNR constraint within 0.05 dB of the PSNR achieved by each SplitSR variant and select the NAWQ-SR model that satisfies it for each dataset. As such, we select IMDN and MobiSR-RCAN to compare with SplitSR-accuracy and -latency, respectively (Table 5).

Fig. 6 shows the measured latency of SplitSR and NAWQ-SR on Urban100 and B100. On the accuracy-driven designs, NAWQ-SR improves latency by 1.59× and 1.60× on Urban100 and B100, respectively. On latency-driven designs, NAWQ-SR demonstrates a performance gain of 4.40× and 4.37× over SplitSR on Urban100 and B100, respectively. As a result, although SplitSR effectively combines a lightweight model design together with compiler optimizations to achieve significant speedup, it still relies on CPU execution,



(a) Comparison on B100. (b) Comparison on Urban100.

Fig. 5: Speedup comparison against MobiSR.



(a) Comparison on B100. (b) Comparison on Urban100.

Fig. 6: Latency comparison against SplitSR.

remaining bounded by the performance of floating-point processors. On the other hand, NAWQ-SR's hybrid precision and optimized utilization of the NPU's processing units avoids the inefficiencies of floating-point execution and reaches higher raw performance over the highly optimized CPU-based SplitSR.

Comparison with XLSR. For fairness, we target the similarly sized TPSR with the same upscaling factor ($\times 3$) as XLSR (Table 5-bottom). NAWQ-SR outperforms the INT8 XLSR with 91% higher speedup. This can be attributed to the fact that XLSR changes the number of channels of the convolutional layers quite frequently along the DNN model as a way of balancing computational cost and model capacity. Despite the theoretical reduction in FLOP count, this has been shown to lead to increased cache-miss rates [73] and in turn to increased latency on existing NPUs. Instead, NAWQ-SR allows existing models to run without architectural modifications by providing latency gains through its hybrid-precision execution. As such, it does not require from DNNs to frequently change the number of channels across convolutional layers, leading to more efficient NPU execution.

With respect to quality, XLSR reports a drop between 0.2-0.5 dB when quantizing to INT8 [18]. NAWQ-SR achieves significant PSNR gains of 0.21-0.51 dB and 0.27-0.57 dB over the INT8 XLSR on B100 and Urban100, respectively, while yielding same or higher PSNR levels over the FP32 XLSR. This can be attributed to the fact that XLSR replaces pixel-shuffle blocks with transpose convolutions in order to avoid the lack of support for pixel-shuffling on NPUs. In turn, this leads to checkerboard artifacts and hence deteriorates the achieved visual quality [76]. On the other hand, NAWQ-SR's memory-aware mapping for efficiently executing pixel-shuffle blocks on the NPU (§ 4.3) leads to both lower latency and higher visual quality, setting a new state-of-the-art in latency-quality for NPU-based SR.

5.5 Energy Consumption

To evaluate NAWQ-SR's energy efficiency, we processed 50 images using TPSR and MobiSR-RCAN. The images are pre-hosted, representing the scenario where a user would have a downloaded content, which is then enhanced with on-device SR. Energy was measured with the Monsoon power monitor [77] at a sampling period of 200 μs.

Fig. 7c shows the average energy consumption for the two models when upscaling to 720p images on SDM865 and SDM888. In this case, we subtract the average idle energy when the screen is on. We observe that NAWQ-SR results in significant energy savings compared to the FP32 CPU execution, with an average $6.1\times$ and $10.3\times$ reduction per model on SDM865 and $8.5\times$ and $14.2\times$ on SDM888. This result motivates the adoption of NPU-optimized frameworks in comparison to state-of-the-art CPU-centric on-device SR approaches, such as SplitSR. Moreover, we see a significant $3.5\times$-$4.3\times$ and $2.1\times$-$2.4\times$ energy reduction, even when compared to the more efficient FP16 GPU and A16W8 NPU, respectively, with similar gains observed for SDM888.

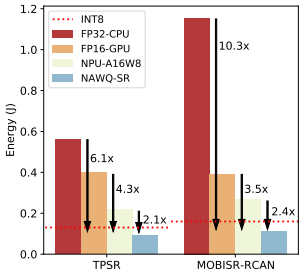
Fig. 7d estimates the battery life when a user continuously watches SR-enhanced video at 1080p on a device with 4000mAh, a common battery capacity for recent mobile devices (*e.g.* Samsung S20). In this case, we measure the total energy, including the screen consumption. NAWQ-SR greatly prolongs the battery life, with up to $3.8\times$, $2.3\times$ and $1.9\times$ battery life extension when compared to CPU, GPU and A16W8 NPU execution, respectively. When targeting SDM888, we observe similar gains, with a slight improvement due to the larger hardware improvement of SDM888's NPU perfomance over the CPU. This result highlights the potential for existing state-of-the-art end-to-end on-device SR systems, such as NEMO, which are bounded to GPU-based execution due to visual quality constraints, to integrate NAWQ-SR as a means of improving not only latency and visual quality, but also extending battery life.

6 Discussion

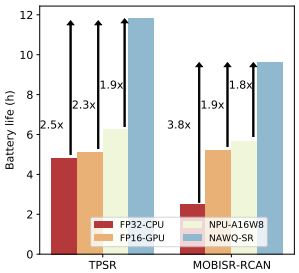
NAWQ-SR and existing mixed-precision schemes. Recently, the ML community has studied a range of mixed-precision quantization schemes that, similarly to NAWQ-SR, assign a different bitwidth to each layer. Focusing on the strategy of selecting the layerwise bitwidth and following the taxonomy of Huang *et al.* [78], we discuss *i)* search-based, *ii)* metric-based, and *iii)* optimization-based methods.

Search-based methods typically rely on neural architecture search (NAS) or reinforcement learning (RL) algorithms in order to yield the layerwise bitwidths. As noted in § 2 with the example of HAQ [43], this family of techniques introduces a significant computational overhead and requires re-training, making it unsuitable for post-training deployment of pre-trained SR models.

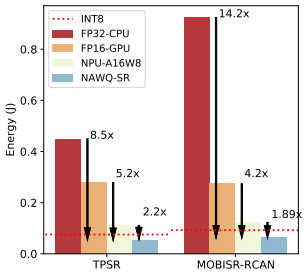
Metric-based methods assign bitwidths by estimating the layerwise resilience to low precision with metrics that are relatively cheap to calculate, such as the Hessian-based metric adopted by HAWQ [44], [79]. Despite the reduced computational burden, existing metric-based methods still require either quantization-aware training or a re-training stage and hence cannot be applied post-training to existing models. Finally, the optimization-based methods aim to



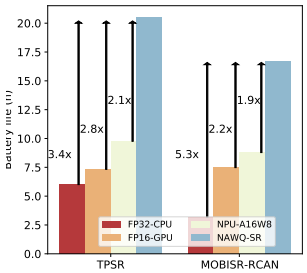
(a) Energy usage on SDM865.



(b) Battery life on SDM865.



(c) Energy usage on SDM888.



(d) Battery life on SDM888.

Fig. 7: Energy consumption and battery life comparison for 720p and 1080p content, respectively.

turn the wordlength selection to a differentiable optimization problem [78], [80]. This line of work can lead to a more stable quantization-aware training process when using mixed-precision. However, these methods have been tailored only for cases where quantization-aware training can be performed.

Despite the various merits of these methods, their effectiveness relies on *training-time* techniques or requires a *re-training step*. For both of these to be feasible, availability of the training set is required. With privacy concerns increasing by both users and service providers, this assumption is often not valid, as in the case of strict privacy regulations for sharing user data [81], privacy-centric initiatives by service providers [82] or confidentiality clauses over proprietary datasets collected by industrial companies.

In this context, NAWQ-SR introduces a wordlength selection method that requires a minimal calibration set and enables the use of hybrid precision in cases where the training set is *not* available. As such, our work offers the computational efficiency of metric-based techniques, but can also be applied directly on pre-trained models *post-training*.

Applicability to other mobile NPUs. In this work, we primarily targeted the NPU of Qualcomm's SDM865 SoC as a representative mobile NPU with available software tools. SDM865 has a vendor-specific hardware architecture, comprising two distinct units, HVX and HTA (see § 5), that support INT8 and both INT8 and A16W8 execution, respectively. As such, if NAWQ-SR was relying on the existence of two distinct units to obtain its performance, it would have narrow applicability to dual-unit NPUs.

On the contrary, NAWQ-SR does not require the existence of two distinct units. NAWQ-SR's processing flow and neural image codec is designed for NPU hardware architectures with either one and two processing units. In our evaluation, we demonstrate this generality of our framework by targeting also the NPU of SDM888, which comprises a single composable processing unit. In a similar fashion, a broad range of existing mobile NPUs, such as the Samsung Exynos NPU [51], MediaTek APU [50] and Arm Ethos [58], consist of a *single* processing unit that can be configured with either INT8 or A16W8 at run time. Hence, by not introducing any optimizations that are coupled to two processing units executing with different wordlength, NAWQ-SR constitutes an on-device SR framework that is generalizable across mobile NPUs from different vendors.

Despite the general underlying principles of our framework, mobile NPUs are often characterized by heterogeneity in terms of both hardware and software [37], [83]. As such, it is difficult to deploy our method out of the box without any further engineering step; to obtain the gains demonstrated by NAWQ-SR, its runtime components may have to be adapted and optimized based on the available API of the target NPU. Nonetheless, with interoperability across diverse mobile SoCs being an active area of research [84], it constitutes an important, yet orthogonal, consideration when attempting to deploy our framework on new NPU-equipped SoC architectures.

7 Conclusion

NAWQ-SR introduces both algorithmic and system optimization techniques on mobile NPUs in order to mitigate the quality drawbacks of executing SR DNNs on low-precision units. Our experiments show that our proposed hybrid-precision method can scale to SR models of varying computational complexity and the run-time precision adaptation method of NAWQ-SR's neural image codec can be efficiently deployed in existing commercial NPUs.

As a stand-alone framework, NAWQ-SR surpasses the performance of existing on-device SR systems, overcoming their limitations and significantly mitigating the quality drawbacks of executing SR DNNs on low-precision units. Additionally, NAWQ-SR can be orthogonally combined with existing frameworks to obtain further gains, by either enabling them to target NPUs, *e.g.* for the CPU-based SplitSR and GPU-based NEMO, or with better utilization of the NPU resources, *e.g.* for MobiSR's NPU-mapped compact model.

References

[1] Cisco, "Cisco Visual Networking Index (VNI) Complete Forecast Update, 2017 - 2022," Cisco Systems, Inc, Tech. Rep., 2020, [Retrieved: March 5, 2023]. [Online]. Available: https://www.cisco.com/c/dam/m/en_us/network-intelligence/service-provider/digital-transformation/knowledge-network-webinars/pdfs/1213-business-services-ckn.pdf

[2] "US Consumers Are Flocking to TikTok," https://www.emarketer.com/content/us-consumers-are-flocking-to-tiktok, 2020, accessed: March 5, 2023.

[3] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang, "Measuring the Quality of Experience of HTTP Video Streaming," in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, 2011, pp. 485–492.

[4] K. Piamrat, C. Viho, J. Bonnin, and A. Ksentini, "Quality of Experience Measurements for Video Streaming over Wireless Networks," in *2009 Sixth International Conference on Information Technology: New Generations (ITNG)*, 2009, pp. 1184–1189.

[5] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring Video QoE from Encrypted Traffic," in *Proceedings of the 2016 Internet Measurement Conference (IMC)*, 2016.

[6] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *SIGCOMM*, 2015.

[7] Y. Zhang, Z. M. Mao, and M. Zhang, "Detecting Traffic Differentiation in Backbone ISPs with NetPolice," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2009.

[8] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016.

[9] R. Lee, S. I. Venieris, and N. Lane, "Neural Enhancement in Content Delivery Systems: The State-of-the-Art and Future Directions," *Proceedings of the 1st Workshop on Distributed Machine Learning (DistributedML)*, 2020.

[10] R. Lee, S. I. Venieris, and N. D. Lane, "Deep Neural Network-based Enhancement for Image and Video Streaming Systems: A Survey and Future Directions," *ACM Comput. Surv. (CSUR)*, 2021.

[11] R. Lee, S. I. Venieris, L. Dudziak, S. Bhattacharya, and N. Lane, "MobiSR: Efficient On-Device Super-Resolution through Heterogeneous Mobile Processors," in *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2019.

[12] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural Adaptive Content-aware Internet Video Delivery," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018.

[13] P. Hu, R. Misra, and S. Katti, "Dejavu: Enhancing Videoconferencing with Prior Knowledge," in *HotMobile*, 2019.

[14] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han, "NEMO: Enabling Neural-enhanced Video Streaming on Commodity Mobile Devices," in *The 26th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2020.

[15] J. Yi, S. Kim, J. Kim, and S. Choi, "Supremo: Cloud-Assisted Low-Latency Super-Resolution in Mobile Devices," *IEEE Transactions on Mobile Computing (TMC)*, 2020.

[16] S. Wang, G. Ananthanarayanan, and T. Mitra, "OPTiC: Optimizing Collaborative CPU–GPU Computing on Mobile Devices With Thermal Constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 38, no. 3, pp. 393–406, 2019.

[17] S. Kang, H. Choi, S. Park, C. Park, J. Lee, U. Lee, and S.-J. Lee, "Fire in your Hands: Understanding Thermal Behavior of Smartphones," in *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2019.

[18] M. Ayazoglu, "Extremely Lightweight Quantization Robust Real-Time Single-Image Super Resolution for Mobile Devices," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 2472–2479.

[19] J. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[20] "QLED 8K: Where AI Upscaling meets Deep Learning," https://news.samsung.com/global/the-future-of-viewing-1-qled-8k-where-ai-upscaling-meets-deep-learning, 2021, accessed: March 5, 2023.

[21] Nvidia, "Dynamic Super-Resolution Improves Your Games with 4K-Quality Graphics on HD Monitors," https://www.nvidia.com/en-us/geforce/news/dynamic-super-resolution-instantly-improves-your-games-with-4k-quality-graphics/, 2021, accessed: March 5, 2023.

[22] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image Super-Resolution Using Very Deep Residual Channel Attention Networks," in *European Conference on Computer Vision (ECCV)*, 2018.

[23] Z. Hui, X. Wang, and X. Gao, "Fast and Accurate Single Image Super-Resolution via Information Distillation Network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[24] J. Yoo, N. Ahn, and K.-A. Sohn, "Rethinking Data Augmentation for Image Super-resolution: A Comprehensive Analysis and a New Strategy," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[25] K. Zhang, L. Gool, and R. Timofte, "Deep Unfolding Network for Image Super-Resolution," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] Z. Luo, Y. Huang, S. Li, L. Wang, and T. Tan, "Unfolding the Alternating Optimization for Blind Super Resolution," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[27] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1874–1883, 2016.

[28] T. Vu, C. Van Nguyen, T. X. Pham, T. M. Luu, and C. D. Yoo, "Fast and Efficient Image Quality Enhancement via Desubpixel Convolutional Neural Networks," in *The European Conference on Computer Vision (ECCV) Workshops*, 2018.

[29] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, Accurate, and Lightweight Super-Resolution with Cascading Residual Network," in *European Conference on Computer Vision (ECCV)*, 2018.

[30] Z. Hui, X. Gao, Y. Yang, and X. Wang, "Lightweight Image Super-Resolution with Information Multi-distillation Network," *Proceedings of the 27th ACM International Conference on Multimedia (MM)*, 2019.

[31] R. Lee, L. Dudziak, M. Abdelfattah, S. I. Venieris, H. Kim, H. Wen, and N. Lane, "Journey Towards Tiny Perceptual Super-Resolution," in *European Conference on Computer Vision (ECCV)*, 2020.

[32] X. Chu, B. Zhang, H. Ma, R. Xu, J. Li, and Q. Li, "Fast, Accurate and Lightweight Super-Resolution with Neural Architecture Search," in *International Conference on Pattern Recognition (ICPR)*, 2021.

[33] D. Song, C. Xu, X. Jia, Y. Chen, C. Xu, and Y. Wang, "Efficient Residual Dense Block Search for Image Super-Resolution," in *Thirty-Fourth AAAI Conference on Artifical Intelligence (AAAI)*, 2020.

[34] X. Liu, Y. Li, J. Fromm, Y. Wang, Z. Jiang, A. Mariakakis, and S. Patel, "SplitSR: An End-to-End Approach to Super-Resolution on Mobile Devices," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. (IMWUT)*, 2021.

[35] M. Almeida, S. Laskaridis, I. Leontiadis, S. I. Venieris, and N. D. Lane, "EmBench: Quantifying Performance Variations of Deep Neural Networks Across Modern Commodity Devices," in *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications (EMDL)*, 2019.

[36] M. Almeida, S. Laskaridis, A. Mehrotra, L. Dudziak, I. Leontiadis, and N. D. Lane, "Smart at what cost? Characterising Mobile Deep Neural Networks in the wild," in *ACM Internet Measurement Conference (IMC)*, 2021.

[37] A. Ignatov *et al.*, "AI Benchmark: All About Deep Learning on Smartphones in 2019," in *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[38] Qualcomm, "Snapdragon Neural Processing Engine," https://developer.qualcomm.com/docs/snpe/snapdragon_npe_runtime.html, 2021, accessed: March 5, 2023.

[39] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 37, no. 1, pp. 35–47, 2018.

[40] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Accurate Post Training Quantization with Small Calibration Sets," in *International Conference on Machine Learning (ICML)*, 2021.

[41] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713.

[42] C. Louizos, M. Reisser, T. Blankevoort, E. Gavves, and M. Welling, "Relaxed Quantization for Discretized Neural Networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[43] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-Aware Automated Quantization with Mixed Precision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8612–8620.

[44] Z. Dong, Z. Yao, Y. Cai, D. Arfeen, A. Gholami, M. W. Mahoney, and K. Keutzer, "HAWQ-v2: Hessian Aware Trace-Weighted Quantization of Neural Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[45] H. Li, C. Yan, S. Lin, X. Zheng, B. Zhang, F. Yang, and R. Ji, "PAMS: Quantized Super-Resolution via Parameterized Max Scale," in *ECCV*, 2020.

[46] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.

[47] Y. Ma, H. Xiong, Z. Hu, and L. Ma, "Efficient Super Resolution Using Binarized Neural Network," *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[48] J. Xin, N. Wang, X. Jiang, J. Li, H. Huang, and X. Gao, "Binarized Neural Network for Single Image Super Resolution," in *European Conference on Computer Vision (ECCV)*, 2020.

[49] Arm, "Ethos NPU," https://developer.arm.com/ip-products/processors/machine-learning/arm-ethos-n, 2021, accessed: March 5, 2023.
[50] C.-H. Lin, C.-C. Cheng, Y.-M. Tsai, S.-J. Hung, Y.-T. Kuo, P. H. Wang, P.-K. Tsung, J.-Y. Hsu, W.-C. Lai, C.-H. Liu, S.-Y. Wang, C.-H. Kuo, C.-Y. Chang, M.-H. Lee, T.-Y. Lin, and C.-C. Chen, "7.1 A 3.4-to-13.3TOPS/W 3.6TOPS Dual-Core Deep-Learning Accelerator for Versatile AI Applications in 7nm 5G Smartphone SoC," in *IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 134–136.
[51] J.-W. Jang, S. Lee, D. Kim, H. Park, A. S. Ardestani, Y. Choi, C. Kim, Y. Kim, H. Yu, and H. Abdel-Aziz, "Sparsity-Aware and Re-configurable NPU Architecture for Samsung Flagship Mobile SoC," in *International Symposium on Computer Architecture (ISCA)*, 2021.
[52] C. Baskin, N. Liss, E. Schwartz, E. Zheltonozhskii, R. Giryes, A. M. Bronstein, and A. Mendelson, "UNIQ: Uniform Noise Injection for Non-Uniform Quantization of Neural Networks," *ACM Trans. Comput. Syst. (TOCS)*, vol. 37, no. 1–4, 2021.
[53] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image Super-Resolution Using Very Deep Residual Channel Attention Networks," in *European Conference on Computer Vision (ECCV)*, 2018.
[54] B. Boashash, *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*. Academic Press, 2015.
[55] J. Song, Y. Cho, J. Park, J. Jang, S. Lee, J. Song, J. Lee, and I. Kang, "7.1 An 11.5TOPS/W 1024-MAC Butterfly Structure Dual-Core Sparsity-Aware Neural Processing Unit in 8nm Flagship Mobile SoC," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2019.
[56] S. Wang, A. Pathania, and T. Mitra, "Neural Network Inference on Mobile SoCs," *IEEE Design Test*, vol. 37, no. 5, pp. 50–57, 2020.
[57] T. Tan and G. Cao, "FastVA: Deep Learning Video Analytics Through Edge Processing and NPU in Mobile," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020.
[58] Arm, "Powering the Edge: Driving Optimal Performance with the Ethos-N77 NPU," https://www.arm.com/-/media/files/pdf/ethos/Arm_Ethos_N77_white_paper_final_v4, 2021, accessed: March 5, 2023.
[59] Qualcomm, "Snapdragon Neural Processing Engine Limitations," https://developer.qualcomm.com/docs/snpe/limitations.html, 2021, accessed: March 5, 2023.
[60] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang, "ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks," in *European Conference on Computer Vision Workshops (ECCVW)*, 2018.
[61] J.-H. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han, "Neural-Enhanced Live Streaming: Improving Live Video Ingest via Online Learning," in *SIGCOMM*, 2020.
[62] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, and L. Zhang, "NTIRE 2017 Challenge on Single Image Super-Resolution: Methods and Results," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.
[63] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing (TIP)*, 2004.
[64] Y. Blau and T. Michaeli, "The Perception-Distortion Tradeoff," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
[65] M. Bevilacqua, A. Roumy, C. Guillemot, and M. line Alberi Morel, "Low-Complexity Single-Image Super-Resolution based on Non-negative Neighbor Embedding," in *British Machine Vision Conference (BMVC)*, 2012.
[66] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image Super-resolution via Sparse Representation," *Trans. Img. Proc. (TIP)*, vol. 19, no. 11, pp. 2861–2873, 2010.
[67] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *IEEE International Conference on Computer Vision (ICCV)*, 2001.
[68] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[69] C. R. Reeves, Ed., *Modern Heuristic Techniques for Combinatorial Problems*. USA: John Wiley & Sons, Inc., 1993.
[70] A. Ignatov, R. Timofte, M. Denna, and A. Younes, "Real-Time Quantized Image Super-Resolution on Mobile NPUs, Mobile AI 2021 Challenge: Report," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 2525–2534.
[71] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
[72] D.-Q. Zhang, "clcNet: Improving the Efficiency of Convolutional Neural Network Using Channel Local Convolutions," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
[73] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," in *The European Conference on Computer Vision (ECCV)*, 2018.
[74] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv*, 2017.
[75] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Q. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "TVM: An Automated End-to-End Optimizing Compiler for Deep Learning," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018.
[76] "Deconvolution and Checkerboard Artifacts," https://distill.pub/2016/deconv-checkerboard/, 2022, accessed: March 5, 2023.
[77] "Monsoon Official Website," https://www.msoon.com/, 2021, accessed: March 5, 2023.
[78] X. Huang, Z. Shen, S. Li, Z. Liu, X. Hu, J. Wicaksana, E. Xing, and K.-T. Cheng, "SDQ: Stochastic Differentiable Quantization with Mixed Precision," in *International Conference on Machine Learning (ICML)*, 2022.
[79] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "HAWQ: Hessian Aware Quantization of Neural Networks with Mixed-Precision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 293–302.
[80] L. Yang and Q. Jin, "FracBits: Mixed Precision Quantization via Fractional Bit-widths," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, no. 12, 2021, pp. 10 612–10 620.
[81] European Commission, "GDPR: 2018 Reform of EU Data Protection Rules." [Online]. Available: https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf
[82] Apple, "Learning with Privacy at Scale," in *Differential Privacy Team Technical Report*, 2017.
[83] C. Wu, D. Brooks, K. Chen, D. Chen, S. Choudhury, M. Dukhan, K. Hazelwood, E. Isaac, Y. Jia, B. Jia, T. Leyvand, H. Lu, Y. Lu, L. Qiao, B. Reagen, J. Spisak, F. Sun, A. Tulloch, P. Vajda, X. Wang, Y. Wang, B. Wasti, Y. Wu, R. Xian, S. Yoo, and P. Zhang, "Machine Learning at Facebook: Understanding Inference at the Edge," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 331–344.
[84] S. I. Venieris, I. Panopoulos, and I. S. Venieris, "OODIn: An Optimised On-Device Inference Framework for Heterogeneous Mobile Devices," in *IEEE International Conference on Smart Computing (SMARTCOMP)*, 2021.

PLACE PHOTO HERE

Stylianos I. Venieris (S'16-M'22) is currently a Senior Research Scientist at Samsung AI, Cambridge, U.K., where he leads the Distributed AI group. He received the Ph.D degree in Reconfigurable Hardware and Deep Learning in 2018 and the M. Eng. degree (Hons.) in Electrical and Electronic Engineering in 2014 from Imperial College London, London, U.K. He has published over 20 research papers in peer-refereed journals and international conferences. His current research interest include methodologies for the principled mapping of deep learning algorithms on mobile and embedded platforms, as well as the design of custom hardware accelerators for the high-performance, energy-efficient deployment of deep neural networks.

PLACE PHOTO HERE

Mario Almeida is currently VP of Engineering at Rain Instant Pay. He received the Ph.D degree in Mobile and Network Systems in 2017 from the Technical University of Catalonia, Barcelona, Spain, and the Master degree in Distributed Computing in 2014 from the KTH Royal Institute of Technology, Stockholm, Sweden. He has published multiple research papers in peer-refereed journals and international conferences. His research interests lie in the intersection of machine learning and mobile systems and networks.

PLACE PHOTO HERE

Royson Lee is currently a Ph.D student in Computer Science at the University of Cambridge, U.K and a part-time Research Engineer at Samsung AI, Cambridge, U.K. He received the MPhil degree in Computer Science in 2018 from the University of Cambridge and the B.Eng degree in Computing from Imperial College London, London, U.K. His research interests include super-resolution, federated learning, and meta-learning.

PLACE PHOTO HERE

Nicholas D. Lane is a Professor in the Department of Computer Science and Technology at the University Cambridge, U.K., where he leads the Machine Learning Systems lab (CaMLSys). Alongside his academic role, he is also a Program Director at the Samsung AI Center in Cambridge where his teams study on-device and distributed forms of machine learning. To find out more about his research, please visit http://niclane.org and https://mlsys.cst.cam.ac.uk.