TileSR: Accelerate On-Device Super-Resolution with Parallel Offloading in Tile Granularity

Ning Chen[1], Sheng Zhang[1*], Yu Liang[2], Jie Wu[3], Yu Chen[1], Yuting Yan[1], Zhuzhong Qian[1] and Sanglu Lu[1]
[1]State Key Lab. for Novel Software Technology, Nanjing University, P.R. China
[2]School of Computer and Electronic Information and the School of AI, Nanjing Normal University, P.R. China
[3]Center for Networked Computing, Temple University, USA
Email: {ningc, DZ1933005, yuting.yan}@smail.nju.edu.cn, {sheng, qzz, sanglu}@nju.edu.cn,
jiewu@temple.edu, liangyu@njnu.edu.cn

***Abstract*—Recent years have witnessed the unprecedented performance of convolutional networks in image super-resolution (SR). SR involves upscaling a single low-resolution image to meet application-specific image quality demands, making it vital for mobile devices. However, the excessive computational and memory requirements of SR tasks pose a challenge in mapping SR networks on a single resource-constrained mobile device, especially for an ultra-high target resolution. This work presents TileSR, a novel framework for efficient image SR through tile-granular parallel offloading upon multiple collaborative mobile devices. In particular, for an incoming image, TileSR first uniformly divides it into multiple tiles and selects the top-K tiles with the highest upscaling difficulty (quantified by mPV). Then, we propose a tile scheduling algorithm based on multi-agent multi-armed bandit, which attains the accurate offload reward through the exploration phase, derives the tile packing decision based on the reward estimates, and exploits this decision to schedule the selected tiles. We have implemented TileSR fully based on COTS hardware, and the experimental results demonstrate that TileSR reduces the response latency by 17.77-82.2% while improving the image quality by 2.38-10.57% compared to other alternatives.**

***Index Terms*—image super-resolution, tile granularity, parallel offloading, multi-agent multi-armed bandit**

I. Introduction

Popular social media networks like Facebook [1], Instagram [2], and Reddit [3] rely heavily on images as users scroll through their feeds or post messages. Given the growing popularity of data-saving alternatives [1], coupled with the increasing need for responsiveness in congested network conditions, enabling mobile devices to download low-resolution images and upscale them locally, is not only feasible but also highly desirable. Despite the superior performance of convolutional neural networks (CNNs) [4]–[6] for image upscaling, deploying and running them on local mobile devices poses significant challenges as super-resolution (SR) models require an excessive number of operations and run-time memory.

Currently, service providers typically use cloud computing solutions [27] where users offload images to a powerful server for upscaling, resulting in high response latency and security risks. Consequently, there is an emerging need to develop systems that support on-device SR. However, as running SR model is resource-intensive, it can hardly meet users' demands for time efficiency by using only local constrained resources, especially for an ultra-high target resolution (i.e., 4k). Instead, we seek a collaborative method that utilizes multiple mobile devices with parallel offloading. As Fig. 1 shows, the local device divides the image into multiple tiles and offloads them to each participating device for tile SR, and each device periodically retrieves the latest SR model from the cloud side.

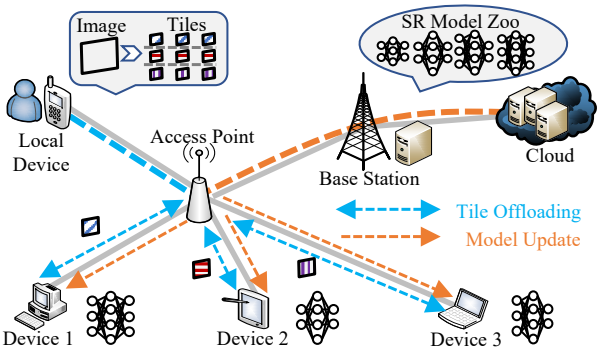


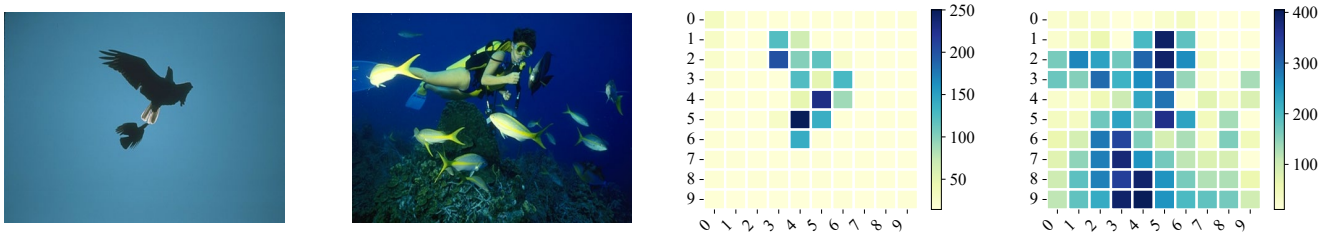
Fig. 1. System architecture of collaborative image SR with parallel offloading.

It is, however, non-trivial to effectively run such a collaborative framework upon multiple mobile devices, as it necessitates seamless management of various aspects, including model deployment due to device heterogeneity, image dividing method, and offloading scheme. In particular, this image SR system with parallel offloading encounters three critical challenges:

First and foremost, since local and collaborative devices are resource-constrained, offloading all the tiles is likely to cause an expensive overhead in terms of power consumption, resource occupancy and response latency. More tiles leads to more data transfers and neural inference. This challenge becomes even more pronounced when upscaling images with larger original resolutions and upscaling factors, or when dealing with unreliable inter-device network connections and intense resource contention among multiple applications. Consequently, there is a vital need to develop methods that effectively alleviate this resource overuse by filtering out tiles that contribute minimally to the overall system improvement.

Second, without prior and accurate knowledge of the offloading revenue mapping between selected tiles and participating devices, making an informed offloading decision becomes challenging. In addition to the complexity of the SR

* The corresponding author is Sheng Zhang (sheng@nju.edu.cn). This work was supported in part by NSFC (62202233, 61832008), Double Innovation Plan of Jiangsu Province (JSSCBS20220409), and Collaborative Innovation Center of Novel Software Technology and Industrialization.



(a) Structured image (b) Unstructured image (c) mPV heatmap for image (a) (d) mPV heatmap for image (b)

Fig. 2. Two images from the DIV2K dataset. (a) and (b) show their visual differences; (c) and (d) present the mPVs for each tile.

model, other factors like image content and run-time resource contention also significantly affect the SR performance. While users are aware of the concrete model deployment for each device, capturing the impact of run-time resource contention and image content is difficult, making it hard to attain the actual offloading revenue. In such a dilemma, finding a way to approximate the actual revenue is of utmost importance.

Third, given the accurate reward estimates, making optimal tile scheduling to maximize the overall reward gains remains challenging. With the tile size and referenced SR models, it is straightforward to calculate the computational demands (i.e., weights) for each device, coupled with the above reward mappings (profits) and device capacities, the proposed tile offloading problem constitutes a multi-knapsack problem [19], which has been proven NP-hard and cannot be solved optimally in polynomial time. Thus it is vital to find an alternative to approximate the optimal solution as accurately as possible.

Existing research falls short of addressing these challenges. Some studies [31], [41] use multiple processors within a local device to run parallel SR. As the image-centric rather than tile-centric input, coupled with the limited capacity of single processor, they fail to meet user demands for time efficiency. Others focus on on-device optimizations [42], [44], [45], such as model compression, model early exit, selective frame for SR, and result reuse, which inevitably lead to a SR quality degradation. A large body of research [27]–[30] has explored edge-server or cloud-based frameworks to alleviate resource constraints on edge devices. For instance, ELF [30] also adopts the parallel offloading method to handle video analytics tasks, but it turns to the edge server rather than the mobile device, thus causing extra latency and privacy risks. To the best of our knowledge, we are the first to propose a multi-device collaborative image SR framework, which effectively addresses these challenges without compromising user experience or privacy.

This work presents TileSR, a new collaborative framework with tile-granular parallel offloading upon multiple devices for accelerating image SR. In particular, TileSR consists of three key designs, e.g., mPV-aware tile selection, exploration-based reward estimation and MKP-like tile packing, which correspond to each of the above three challenges, respectively.

First of all, to reduce the useless tile offloading, we propose a selective mechanism based on the fact derived from our experimental preliminaries: tiles vary in upscaling difficulty quantified by mPV (mean Pixel Variation), and high-mPV tiles yield higher quality (i.e., PSNR) when using CNN models compared to interpolation-based methods, while low-mPV tiles attain high quality regardless of the upscaling method used. Using this property, TileSR selects only the top K tiles with the highest mPV for offloading and tile SR in collaborative devices, while upscaling the remaining tiles using interpolation methods with local resources.

Then, we model and formulate the problem of offloading top-K tiles upon multiple participating devices, with the goal of maximizing the overall reward in terms of SR quality, response latency and energy consumption in the long run. Furthermore, the sum of demanding computations of the offloaded tiles in each device should not exceed its capacity, and the frame-level response time should meet the threshold. Considering that the offloading reward is indeed stochastic and time-varying, but bounded up and down within a range, we design an *exploration-based* method in a multi-agent bandit framework to attain the expected value as the reward estimates.

Finally, based on the resource demand, the concrete reward mapping between selected tiles and collaborative devices, and the device capacities, we consider the tile offloading problem as a multi-knapsack problem (MKP), and propose a 2-approximation algorithm, which decouples the MKP problem into a series of knapsack sub-problems and recursively solves each sub-problem to update the final offloading decision.

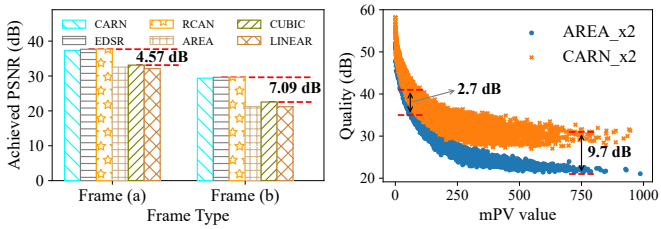
We have implemented TileSR fully based on COTS hardware including four collaborative devices, and each device is deployed a SR model from models CARN [5], EDSR [7], MSRN [8] and RCAN [9] that best matches its capacity. The experimental results measured in Set5 [25], Set14 [26] and DIV2K [24] demonstrate that TileSR reduces the response latency by 17.77-82.2% while improving the image quality by 2.38-10.57% compared to other alternatives.

II. Motivating Studies for Tile Selection

This section first introduces a metric called *upscaling difficulty* that plays a key role in image SR. Then, based on this, we present a tile selection mechanism.

A. Difficulty Analysis for Neural Super-Resolution

To maintain high quality, more powerful CNN models are used for super-resolution, ignoring the fact that not all images have the same upscaling difficulty. That is, using the same model to upscale different images could yield different qualities. To verify this, we collect two images from DIV2K dataset [24], including: (1) Fig. 2(a), which is highly structured and smooth, and (2) Fig. 2(b) that contains unstructured



(a) Quality under diverse methods (b) Quality distribution in DIV2K

Fig. 3. Quality comparison of CNN-based and interpolation-based methods.

fine details and texture. Then, six methods, including (1) CNN-based CARN [5], EDSR [7], and RCAN [9] and (2) interpolation-based AREA, CUBIC, and LINEAR provided in OpenCV, are applied to upscale Fig. 2(a) and Fig. 2(b).

Fig. 3(a) illustrates the achieved SR quality measured by PSNR. It is clear that both CNN-based and interpolation-based methods work well to upscale Fig. 2(a), while achieving significantly lower quality for Fig. 2(b), indicating that Fig. 2(a) is much easier to upscale than Fig. 2(b). To quantify such upscaling difficulty similar to the prior work [31], we first define the Pixel Variant Matrix PV_f for image f,

$$PV_f[i,j]=\sum_{w=\max\{1,i-1\}}^{\min\{i+1,W\}}\sum_{h=\max\{1,j-1\}}^{\min\{j+1,H\}}|p_{i,j}-p_{w,h}|, \quad (1)$$

where $p_{i,j}$ is the pixel value, and W and H represent the frame width and height, respectively. To this end, we define the mean pixel variant (mPV) $mPV_f = \sum_{i=1}^{W}\sum_{j=1}^{H} PV_f[i,j]/(WH)$, which is used to indicate the upscaling difficulty for image f. For instance, Fig. 2(a) has a lower mPV (i.e., 125) than Fig. 2(b) (i.e., 275), meaning that Fig. 2(a) is easier to upscale.

B. mPV-aware Top-K Tiles Selection

Without loss of generality, the eq. (1) can also be used to define the mPV for image tiles. Within a frame, we observe that there are likely to be tiles that are both easy and difficult to upscale for super-resolution. As shown in Fig. 2(c) and Fig. 2(d), we divide Fig. 2(a) and Fig. 2(b) equally into 10×10 tiles and calculate their mPVs individually. It is clear that even though Fig. 2(a) is easily upscaled, it still contains several tiles with higher mPVs. On the other hand, Fig. 2(b) has a high mPV, but it consists of substantial low-mPV tiles.

Goal: Among these tiles $\{m\}$ with different mPVs, the goal of TileSR is to select partial tiles that benefit the overall performance (described in section III) the most in terms of SR quality, response latency, and energy consumption. Since TileSR splits the images uniformly, each tile has the same size and thus has the identical impact on response latency and energy consumption [10], [11]. Therefore, we only consider the contribution of each tile to the SR quality, which relies heavily on its upscaling difficulty mPV.

Heuristics: To explore the relationship between tile mPV and SR quality in-depth, we conduct a large-scale preliminary experiment. We first uniformly divided each image in the DIV2K training and validation sets [24] into multiple tiles with the same size 20x20, and then we calculated the mPV of each tile together with the achieved PSNR by running our reference model (i.e., CARN [5]) and interpolation-based method (i.e., AREA). Based on the results in Fig. 3(b), we conclude that: *(1) for low-mPV tiles, both CNN-based and interpolation-based methods yield high and similar PSNR values, and (2) for high-mPV tiles, both these two methods acquire lower PSNR values, but the CNN-based method gets significantly higher PSNRs than the interpolation-based method.* For example, in Fig. 3(b), CARN achieves 9.7 dB higher PSNR than AREA when the mPV is 750, which is significantly higher than that (2.7 dB) when the mPV is only 100.

Method: Above all, we propose a mPV-aware tile selection mechanism as follows: (1) calculate the upscaling difficulties $\{mPV\}$ for each tile $m_i \in \{m\}$, and sort $\{m\}$ in descending order by the mPV value; (2) select the top-K tiles with the highest mPV values as the offload targets for CNN-based super-resolution, and locally upscale the remaining tiles with the interpolation-based method. Next, we focus on how to schedule these selected tiles to the participating edge devices.

TABLE I
SUMMARY OF NOTATIONS USED FOR SYSTEM MODEL

Inputs	Description
$\mathcal{K}$	$\{1,\ldots,k,\ldots,K\}$, set of selected top-K *Tiles*
$\mathcal{D}$	$\{1,\ldots,d,\ldots,D\}$, set of involved *Devices*
$\mathcal{T}$	$\{1,\ldots,t,\ldots,T\}$, set of time slots with equal length
$\mathcal{K}_{d,t}$	Set of tiles that are offloaded to device d at slot t
C_d	Computing capacity of device d
$c_{k,d}$	Computation demand to run tile k in device d
$Q_{k,d}$	Achieved SR quality (PSNR) to run tile k in device d
$E_{k,d}$	Energy consumption to inference tile k in device d
$L_{k,d}$	Response latency to inference tile k in device d
Decisions	Descriptions
$x_{k,t}$	Device selection for tile $k \in \mathcal{K}$ in time slot $t \in \mathcal{T}$

III. System Model and Problem Formulation

A. System Settings and Models

We summarize the major notations used in Table I.

Inter-Device Networks: We consider a set of edge devices accessed by a local device, denoted as $\mathcal{D} = \{1,\ldots,D\}$, each of which supports lightweight neural inference through highly compressed models that best match its capacity. These devices are connected to users via wireless local area networks (WLANs), such as WiFi, ZigBee, and Bluetooth, to ensure fast response times and to mitigate privacy and security risks. Examples of such devices include smart home devices such as cell phones, iPads, electronic picture frames and televisions, which are ideal for this purpose. Another scenario involves multiple unmanned aerial vehicles (UAVs) within a WLAN that can collaborate to perform various computing tasks.

Tile-Granular Parallel Offloading: By mPV-aware selection, we get the top-K tile set, denoted by $\mathcal{K}=\{1,\ldots,K\}$. The entire time horizon is divided into several slots, represented by $\mathcal{T}=\{1,\ldots,T\}$. We use $x_t=(x_{1,t},\ldots,x_{K,t})$ to indicate the overall offload decision for $\mathcal{K}$ in time slot t, where $x_{k,t}$ represents the device selected for tile k. In each time slot $t \in \mathcal{T}$, based on the control decision, TileSR handles tiles $\mathcal{K}$ in three steps: (1) for each tile $k \in \mathcal{K}$, TileSR offloads it to device $d = x_{k,t} \in \mathcal{D}$; (2) then device d runs the SR model to upscale tile k to the target resolution and returns the output

to the user side; and (3) finally, TileSR merges these upscaled tiles into a complete high-resolution frame.

Super-Resolution Quality: For image quality assessment, the most commonly used metrics are the comparative scores such as VMAF, PSNR, and SSIM, which calculate the difference between the enhanced image and the baseline frame (i.e., ground truth). For a given tile k, due to the unavailability of the high-resolution tile (i.e., ground truth) k', we cannot directly obtain the difference of the tile pair (k, k'). Instead, we model the SR quality based on the following two observations derived from the preliminary experiment: (1) the SR quality is negatively related to the tile mPV value, which can be modeled as a concave function; (2) a larger upscale factor leads to lower quality; and (3) the mPV value and the upscale factor affect the SR quality independently. In particular, we compute $Q_{k,d}$

$$Q_{k,d} = \varphi_d\left(V_k\right)\psi_d\left(P_k\right), \tag{2}$$

where the functions $\varphi_d\left(V_k\right)$ and $\psi_d\left(P_k\right)$ represent the quality with respect to the mPV value V_k and the upscaling factor P_k of the tile k, respectively. Considering the heterogeneity of participating devices, we prepare multiple SR CNN models by controlling the compression and pruning level, and each device uses a model that best matches its maximum resource.

Energy Consumption: CNN-based super-resolution is expensive in power consumption, but edge devices have inherently limited power, so the energy consumption should be considered [10]. Since two main procedures, including tile transfer and CNN inference, account for most of the energy consumption, we model the energy consumption $E_{k,d}$ as

$$E_{k,d} = \left(\alpha + \beta_d\right)\gamma\left(p_{k,l} * p_{k,w}\right), \tag{3}$$

where $\gamma\left(p_{k,l} * p_{k,w}\right)$ is the data scale of tile k with size $p_{k,l} \times p_{k.w}$, and compression rate γ, and α and β_d denote the energy per bit for transmission and inference [11], respectively.

Constraint of Response Latency: For each tile $k \in \mathcal{K}$, its response latency $L_{k,d}$ consists of three components, i.e., the tile transfer time $l^{up}_{k,x_{k,t}}$ from the user side to the target device $x_{k,t}$, the tile super-resolution time $l^{sr}_{k,x_{k,t}}$, and the result feedback time $l^{down}_{k,x_{k,t}}$. Considering that all devices perform CNN super-resolution on their received tiles in parallel, the response latency to handle tiles $\mathcal{K}$ in slot t is defined as the completion time of the last tile. To ensure the time efficiency of multi-device parallel SR, the response time must hold,

$$\max\left\{l^{up}_{k,x_{k,t}} + l^{sr}_{k,x_{k,t}} + l^{down}_{k,x_{k,t}} | k \in \mathcal{K}\right\} \leq L_{max}. \tag{4}$$

Constraint of Devices Capacity: Each heterogeneous edge device $d \in \mathcal{D}$ is provisioned with a limited computational capacity C_d. If too many tiles are offloaded to device j that the computational demand exceeds C_d, some of the tiles will have to wait for idle resources, resulting in a longer response time. In particular, for each participating device, it holds that,

$$\sum\nolimits_{k \in \mathcal{K}_{d,t}} c_{k,d} \leq C_d, \tag{5}$$

where $\mathcal{K}_{d,t}$ is the set of tiles that are offloaded to device d at slot t. Similarly to the research in [12], the number of operations is proportional to the input tile scale $\gamma\left(p_{k,l} * p_{k,w}\right)$, so the computational demand $c_{k,d}$ is denoted as $\lambda_d\gamma\left(p_{k,l} * p_{k,w}\right)$ with the proportionality λ_d to indicate model complexity.

Reward: From the user's perspective, SR quality and response latency play a key role in the quality of the viewing experience; from the system's perspective, power consumption has a significant impact on overall performance. Thus, we define the reward $u_{k,d}$ as the weighted sum of them,

$$u_{k,d} = \delta_1 Q_{k,d} - \delta_2 E_{k,d} - \delta_3 L_{k,d}, \tag{6}$$

where δ_1, δ_2 and δ_3 are parameter weights to tradeoff SR quality, response latency and energy consumption, respectively. Given the unknown and uncertain system-side information, including 1) the runtime quality functions (i.e., φ_d and ψ_d in each device), which are largely affected by image content and resource contention, and 2) the energy consumption rate (i.e, β_d) for participating devices, each tile can only derive an independent identical distribution (i.i.d) random reward value $\tilde{u}_{k,d}(t)$ in time slot t, with $u_{k,d} = \mathbb{E}\left[\tilde{u}_{k,d}(t)\right], \tilde{u}_{k,d}(t) \in [\underline{u}, \bar{u}], \forall k \in \mathcal{K}, \forall d \in \mathcal{D}$, where $\underline{u}$ and $\bar{u}$ are the lower and upper bounds of the reward, respectively.

B. Problem Formulation

With the above system models, we formulate the following optimization problem to control the tile offloading decision x_t upon such a multi-device parallel offloading framework:

$$\begin{aligned}
\mathbb{P}: &\max_{\{x_t | t \in \mathcal{T}\}} \sum\nolimits_{t \in \mathcal{T}} \sum\nolimits_{k \in \mathcal{K}} \mathbb{E}\left[\tilde{u}_{n,x_{k,t}}(t)\right] \\
\text{s.t. } &\sum\nolimits_{k \in \mathcal{K}_{d,t}} c_{k,d} \leq C_d, \forall d \in \mathcal{D}, \forall t \in \mathcal{T}, \\
&\max\left\{L_{k,x_{k,t}} | k \in \mathcal{K}\right\} \leq L_{max}, \forall t \in \mathcal{T}.
\end{aligned}$$

Model Discussion: To aggressively maximize the utilitarian compound reward in $\mathbb{P}$, tiles that perceive high rewards are likely to be offloaded to devices with lower energy consumption and high super-resolution quality, thereby depriving other tiles of the opportunity to select "better" devices and causing unfair device selection for neural super-resolution. Despite the *constraint of response latency*, which mitigates such unfairness to some extent, multiple tiles are likely to be offloaded to the same devices, making the response latency close to the time threshold and making it run out of energy quickly.

Problem Transformation: Therefore, to avoid these undesired results, we explore the proportional fairness maximization [13], [14] to achieve a fair reward distribution among the frame tiles by modifying the reward function $r_{k,x_{k,t}}$ as $\ln\left(1 + \eta u_{k,x_{k,t}}\right), \eta > 0$. Above all, we take into account the proportional fairness and transform $\mathbb{P}$ into $\mathbb{P}_1$, i.e,

$$\begin{aligned}
\mathbb{P}_1: &\max_{\{x_t | t \in \mathcal{T}\}} \sum\nolimits_{t \in \mathcal{T}} \sum\nolimits_{k \in \mathcal{K}} \mathbb{E}\left[\tilde{r}_{k,x_{k,t}}(t)\right] \\
\text{s.t. } &\tilde{r}_{k,x_{k,t}}(t) = \ln\left(1 + \eta\tilde{u}_{k,x_{k,t}}(t)\right), \\
&\sum\nolimits_{k \in \mathcal{K}_{d,t}} c_{k,d} \leq C_d, \forall d \in \mathcal{D}, \forall t \in \mathcal{T}, \\
&\max\left\{L_{k,x_{k,t}} | k \in \mathcal{K}\right\} \leq L_{max}, \forall t \in \mathcal{T}.
\end{aligned}$$

Accordingly, the user perceives the i.i.d random reward function $\tilde{r}_{n,x_{k,t}}(t)$ instead of $\tilde{u}_{n,x_{k,t}}(t)$, with $r_{k,d} = \mathbb{E}\left[\tilde{r}_{k,d}(t)\right], \tilde{r}_{k,d}(t) \in [\underline{r}, \bar{r}], \forall k \in \mathcal{K}, \forall d \in \mathcal{D}$, where $\underline{r}$ and $\bar{r}$ are the lower and upper bounds of the reward, respectively.

Key Challenge: The goal is to maximize the long-run total reward. It is not yet trivial to derive the optimal solution for $\mathbb{P}_1$, since the observed reward $\tilde{r}_{n,x_{k,t}}(t)$ is uncertain and

Algorithm 1 Decentralized Online Tile Offloading

Input: $\mathcal{K}$, $\mathcal{D}$, $\mathcal{T}$, $T_{explore}$, $T_{exploit}$

1: $R^{(\pi)} \leftarrow \left\{\tilde{r}_{k,d}^{(\pi)}= 0, \forall k \in \mathcal{K}, \forall d \in \mathcal{D}\right\}$, $\forall \pi \in \Pi$;
2: $\Xi^{(\pi)} \leftarrow \left\{\chi_{k}^{(\pi)}= 0, \forall k \in \mathcal{K}\right\}$, $\forall \pi \in \Pi$;
3: **for** epoch $\pi = 1$ to π_T **do**
4: Invoke **Alg. 2**: $\tilde{R}^{(\pi)}$= **Exploring**$\left(R^{(\pi)}, T_{explore}\right)$;
5: Invoke **Alg. 3**: $\tilde{\Xi}^{(\pi)}$ = **Packing**$\left(\tilde{R}^{(\pi)}, \Xi^{(\pi)}\right)$;
6: **for** each of the remaining $T_{exploit}$ time slots **do**
7: Offloading tiles to device by **Exploiting** $\tilde{\Xi}^{(\pi)}$;

Algorithm 2 Reward **Exploring** with Multi-Agent Bandit

Input: $R^{(\pi)}$, $T_{explore}$, K, $\underline{K}$, D

1: **for** time slot $t = 1$ to $T_{explore}$ **do**
2: **for** group g to $\left\lceil \frac{K}{\underline{K}D} \right\rceil$ **do**
3: Each tile k in group g is offloaded to target
4: device $d = \lfloor ((k+t)\%\left(\underline{K}D\right))/\underline{K} \rfloor + 1$;
5: Update $\tilde{r}_{k,d}^{\pi} \leftarrow \tilde{r}_{k,d}^{\pi-1} + \left(\tilde{r}_{k,d} - \tilde{r}_{k,d}^{\pi-1}\right)/\pi$;

Output: $\tilde{R}^{(\pi)} = \left\{\tilde{r}_{k,d}^{(\pi)}, \forall k \in \mathcal{K}, \forall d \in \mathcal{D}\right\}$

time-varying. Fortunately, we observe that the changes in SR quality, latency, and energy consumption remain bounded within a specific range. For example, in Fig. 3(b), the achieved PSNR fluctuates within a certain range for a given upscaling difficulty. The other two metrics show a strong correlation with the tile size, resulting in their bounded behavior as well. As a result, their expected values can effectively represent a device's SR performance, inspiring us to develop an algorithm based on exploration and exploitation to solve problem $\mathbb{P}_1$.

IV. Decentralized Tile Offloading Algorithm

In this section, we detail the algorithm design for TileSR. In particular, the local side tile scheduling is modeled as a multi-agent multi-armed bandit problem by treating offloading to an edge device as playing an arm. The overall workflow is presented in Alg. 1, where Alg. 2 is responsible for exploring the potential reward variants, and Alg. 3 is invoked to derive the final solution by solving a series of knapsack subproblems.

A. Decentralized Tile Offloading

Given that the offloading reward is uncertain but bounded, we develop an exploration and exploitation-based algorithm. Prior research [15]–[18] has shown that multi-armed bandit methods effectively learn random rewards from multiple "bandits". However, directly applying the multi-armed bandit method to each tile may cause conflicts of interest due to the decentralized nature. Besides, since tiles vary in mPV, it is essential to devise an extended multi-armed bandit method. Thus, as Alg. 1 shows, we introduce a decentralized algorithm based on the multi-agent multi-armed bandit for tile offloading.

Regret: Since we apply bandit-based Alg. 1 to solve $\mathbb{P}_1$, we define the *regret* to quantify the offloading performance as

$$\mathcal{R}(T) = T \sum\nolimits_{k\in\mathcal{K}} r_{k,x_{k,t}^*} - \sum\nolimits_{t=1}^{T} \sum\nolimits_{k\in\mathcal{K}} \mathbb{E}\left[\tilde{r}_{k,x_{k,t}}(t)\right], \quad (7)$$

Algorithm 3 Tile **Packing** upon Multi-Knapsack Problem

Input: $\tilde{R}^{(\pi)}, \tilde{\Xi}^{(\pi)}$, K, D, C

1: $R_1 \leftarrow \tilde{R}^{(\pi)}$, $\widehat{S} \leftarrow \varnothing$, $d \leftarrow 1$;
2: **while** $d \le D$ **do**
3: Run algorithm $\Pi(R_d, C_d)$ and return S_d;
4: **for** tile $i \in S_d$ **do**
5: **if** $\exists d', 1 \le d' < d$ s.t. $i \in S_{d'}$ **then**
6: Update $S_{d'} \leftarrow S_{d'} \setminus \{i\}$;
Update decision $\widehat{S} \leftarrow \widehat{S} \bigcup \{S_d\}$;
7: **for** tile k to K **do**
8: **for** device j to D **do**
9: $R_d^1[k,j] = \begin{cases} R_d[k,j], & \text{if } k \in S_d \text{ or } j = d, \\ 0, & \text{otherwise,} \end{cases}$
10: Set $R_d^2 \leftarrow R_d - R_d^1$, $R_{d+1} \leftarrow R_d^2$, $d \leftarrow d+1$;

Output: $\widehat{S}$

where the offloading decision $\{x_{k,t}, k \in \mathcal{K}, t \in \mathcal{T}\}$ is derived from Alg. 1, and $\{x_{k,t}^*, k \in \mathcal{K}, t \in \mathcal{T}\}$ is the optimal solution. To minimize the regret $\mathcal{R}(T)$, striking exploration-exploitation balance is crucial. As the total time T is not predetermined, we divide the time horizon into distinct epochs $\{1, 2, \ldots, \pi_T\}$, with each epoch containing a variable number of time slots, and π_T indicating the last epoch. This epoch-based structure allows us to effectively address the exploration-exploitation tradeoff, with each epoch encompassing an exploration phase, a packing phase, and an exploitation phase.

B. Reward Exploration and Exploitation

Exploration: During the exploration phase, which spans $T_{explore}$ time slots in each epoch, there is a concern that the total computational demand for CNN super-resolution of tiles $\mathcal{K}$ may surpass the available computational capacity of the participating devices. To prevent potential issues like tile queuing or abandonment, we introduce a group-based offloading scheme in a round-robin fashion for reward exploration. As illustrated in lines 2-4 of Alg. 2, for each tile k belonging to a group in time slot t, it is sent to device $x_{k,t}$ computed as

$$x_{k,t} = \lfloor ((k+t)\%(\underline{K}D))/\underline{K} \rfloor + 1, \quad (8)$$

where $\underline{K}$ is calculated as $\min_{d\in\mathcal{D}, k\in\mathcal{K}}\{\frac{C_d}{c_{k,d}}, \}$, and $\underline{K}D$ is the group size so not to violate any device capacity. The perceived reward $\tilde{r}_{k,d}^{\pi}$ is initially set to 0 and updated (line 5) with

$$\tilde{r}_{k,d}^{\pi} \leftarrow \tilde{r}_{k,d}^{\pi-1} + \left(\tilde{r}_{k,d} - \tilde{r}_{k,d}^{\pi-1}\right)/\pi, \quad (9)$$

which includes the explored reward $\tilde{r}_{k,d}^{\pi-1}$ along with the reward gain or loss of the current epoch π compared to the reward of the previous epoch $\pi - 1$. As π rises, the reward variant achieves a diminishing marginal benefit.

Exploitation: Using the reward estimates $\{\tilde{r}_{k,d}^{(\pi)}\}$, TileSR employs Alg. 3 to generate the offloading scheme $\tilde{\Xi}^{(\pi)}$, which will be further elaborated in the following subsection. During the remaining $T_{exploit}$ time slots in epoch π, we offload the selected top-K tiles by exploiting the packing scheme $\tilde{\Xi}^{(\pi)}$. Notably, the number of time slots allocated to $T_{exploit}$ is

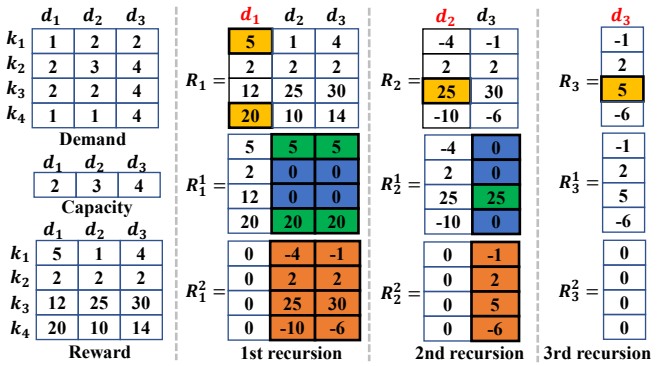


Fig. 4. Running example of Alg. 3 with 4 tiles and 3 devices.

significantly larger than $T_{explore}$, and it continuously increases throughout epoch π.

C. Tile Packing upon Multi-Knapsack Problem

After obtaining the reward estimation $\{\tilde{r}_{k,d}^{(\pi)}\}$ through the exploration phase, this phase aims to obtain the tile packing decision $\{\chi_k^{(\pi)}\}$. In particular, for each time slot t, tile packing can be modeled as a multi-knapsack problem (MKP) [19]:

$$\begin{aligned}\mathbb{P}_2: &\max_{\boldsymbol{x}_t} \sum\nolimits_{k\in\mathcal{K}} \tilde{r}_{k,x_{k,t}} \\ \text{s.t.} &\sum\nolimits_{k\in\mathcal{K}_{d,t}} c_{k,d} \le C_d, \forall d\in\mathcal{D}, \\ &\max\left\{L_{k,x_{k,t}} | k\in\mathcal{K}\right\} \le L_{max},\end{aligned}$$

which has been proven NP-hard [20] and cannot obtain the optimal solution in polynomial time. Instead, as Alg. 3 shows, we propose a multi-knapsack based approximation algorithm.

MKP Packing: For problem $\mathbb{P}_2$ with K tiles and D devices, let R be a $K \times D$ profit matrix derived from the reward estimations $\{\tilde{r}_{k,d}^{(\pi)}\}$, where $R[k,d]$ represents the profit of tile k when selected for device d. Let Π be an algorithm (e.g., branch-and-band or dynamic programming methods [21], [22]) that effectively solves the single-knapsack problem.

Considering that a recursive method is adopted to update the selected tiles in each device, we use R_d to denote the profit matrix at the d-th recursive call. Initially, we set R_1 to R, and each recursion follows two steps:

- Lines 3-6: Run algorithm Π on device d with profit function R_d, obtain the initial selected item set S_d, and update $\widehat{S}$;
- Lines 7-11: Based on S_d, decompose the profit function into two functions R_d^1 and R_d^2 for each $k \in \mathcal{K}$ and $d \in \mathcal{D}$. In particular for R_d^1, $\forall j \in \mathcal{D}$, $R_d^1[k,j]$ is set to $R_d[k,d]$ if tile $k \in S_d$ or $j = d$; otherwise, $R_d^1[k,j]$ is set to 0. Then, we set R_d^2 to $R_d - R_d^1$. In short, R_d^1 is identical to R_d in first column; besides, if item $k \in S_d$, then $R_d^1[k,j]$ for all devices are the same; other entries are set to zero. Finally, we set R_{d+1} to R_d^2 and abandon the column values for device d in R_{d+1}, and begin $(d+1)$-th recursion.

The final union set $\widehat{S}$ indicates the final packing result. Fig. 4 illustrates a running example of the proposed algorithm on 4 tiles $\{k_1, k_2, k_3, k_4\}$ and three devices $\{d_1, d_2, d_3\}$. At the first (e.g., d_1-th) recursion, based on the values in the first column in R_1, algorithm Π returns the current best item sets $S_1 = \{k_1, k_4\}$ for device d_1. Then, in R_1^1, it sets the entries in rows 1 and 4 to $R_1^1[1,1]$ and $R_1^1[4,1]$, respectively, but assigns 0 to other entries. The operations are similar for d_2-th and d_3-th recursions. Finally, it outputs the packing decision $\widehat{S} = \{\{k_1, k_4\}, \{\}, \{k_3\}\}$, which can be used for exploitation.

TABLE II
DEVICE DESCRIPTION IN TILESR IMPLEMENTATION.

Device	Description	SR Model
Dell Desktop	Intel Core i5-11500, 2.70GHz	AREA
Raspberry Pi 4B	500 MHz VideoCore VI	CARN [5]
Jetson NANO	128-core NVDIA Maxwell	RCAN [9]
Jetson TX2	256-core NVDIA Pascal	MSRN [8]
Jetson Xavier NX	385-core Volta +48 Tensor Cores	EDSR [7]

V. PERFORMANCE ANALYSIS

This section theoretically analyzes the performance of TileSR, including the *Error Probability* of exploration, the *Packing Accuracy* given the explored and expected reward, the overall *regret* and the *approximation* of tile packing.

Lemma 1 (Exploration Error Bound): *After the π-th reward exploration in Alg. 2, the Error Probability is bounded:*

$$\Pr\left(\left|\tilde{r}_{k,d}^{\tau} - r_{k,d}\right| > 2\mu_{\min}/9\overline{K}\right) \le 2KDe^{-\pi}, \quad (10)$$

where item $\overline{K}$=$\max\{\frac{C_d}{c_{k,d}}, d\in\mathcal{D}, k\in\mathcal{K}\}$, and item $\mu_{\min} = \{\mu_1, \mu_2, \mu_3\}$, in which $\mu_1 = \min_{k\neq k'\in\|, d\in\mathcal{D}}\{|r_{k,d} - r_{k',d}|\}$, $\mu_2 = \min_{k\neq k'\in\mathcal{K}, d\neq d'\in\mathcal{D}}\{|(r_{k,d} - r_{k,d'}) - r_{k',d}|\}$, and $\mu_3 = \min_{k\neq k'\in\mathcal{K}, d'\neq d\neq d''\in\mathcal{D}}\{|(r_{k,d} - r_{k,d'}) - (r_{k',d} - r_{k',d''})|\}$.

Proof. See Appendix A. □

Lemma 2 (Packing Accuracy Guarantee): *The packing scheme $\tilde{\Xi}^{(\pi)}$ derived from the explored reward estimates $\{\tilde{r}_{k,d}^{(\pi)}\}$ is the same as that from the expected rewards $\{r_{k,d}, \forall k\in\mathcal{K}, \forall d\in\mathcal{D}\}$ if the inequality $|\tilde{r}_{k,d}^{\tau} - r_{k,d}| \le \frac{2\mu_{min}}{9\overline{K}}$ holds.*

Proof. See Appendix B. □

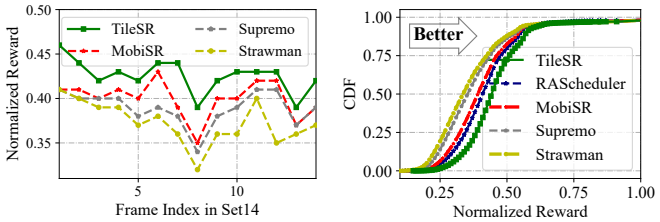
Theory 1 (Regret Bound): *The **regret** is bounded as $\mathcal{R}(T) \le O(log_2 T)$ when using exploration method to estimate the reward in Alg. 2 without certain prior information.*

Proof. See Appendix C. □

Lemma 3 (Approximation Ratio): *Alg. 3 outputs a 2-approximate solution for the multi-knapsack problem $\mathbb{P}_2$ by recursively tackling the tile packing for each device.*

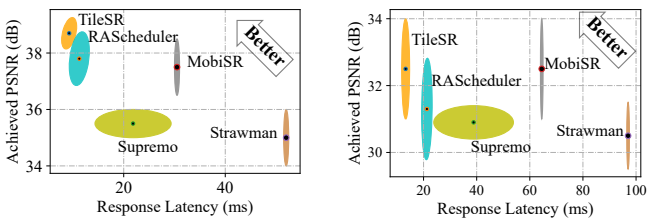
Proof. See Appendix D. □

Discussion: **Lemma 1** ensures that after several epochs in the exploration phase, the reward estimates $\{\tilde{r}_{k,d}^{(\pi)}\}$ are close to the expected reward $\{r_{k,d}^{(\pi)}\}$ with a high probability. To guarantee a lower error, i.e. $2\mu_{\min}/9\overline{K}$, a larger number of time slots for reward exploration is required. **Lemma 2** states that if $\{\tilde{r}_{k,d}^{(\pi)}\}$ are close enough to $\{r_{k,d}^{(\pi)}\}$, they will yield the same results when entered into Alg. 3 individually. **Theorem 1** shows that the logarithmic regret bound is tight, since a lower $log_2 T$ regret bound can be derived. Finally, by decoupling the multi-knapsack problem into a series of single-knapsack problems and using a recursive method to update the packing decision, Alg. 3 obtains a 2-approximation solution for $\mathbb{P}_2$.



(a) Performance in Set14 (b) CDF of reward in DIV2K

Fig. 5. Overall performance of TileSR and other designs.



(a) Under upscaling factor x2 (b) Under upscaling factor x4

Fig. 6. The response latency vs. SR quality of TileSR and other alternatives.

VI. Implementation and Evaluation

We have implemented TileSR fully based on commodity hardware. This section demonstrates its performance.

A. Experimental Setup

Hardware and Software Settings: Table II showcases the commodity devices utilized for image super-resolution. Specifically, the Dell Desktop acts as the local device responsible for collecting images and performing parallel-offloading. The other devices serve as collaborative devices, receiving tiles from the local device and executing specific neural super-resolution algorithms. All devices are connected within a WLAN, sharing the same WiFi network and communicating through Socket using the TCP protocol. To best match computational capacity, we assign suitable models to each device, resulting in the deployment of four models (CARN [5], RCAN [9], MSRN [8], and EDSR [7]) within the TensorFlow framework. Additionally, the AREA interpolation method is provided in OpenCV for comparison. Moreover, we monitor energy consumption using the AITEK AWE2101 [23].

Training and Testing Dataset: The DIV2K [24] dataset consists of 800 images for training, 100 images for validation, and 100 images for testing. Thus, we use the first part to pre-train the above models on two powerful Tesla V100 GPUs, and we use the last two parts in combination with the Set5 [25] and Set14 [26] datasets to test the performance of TileSR.

We compare our proposed TileSR with following schemes:

- **Supremo** [27]: A cloud-based image super-resolution system that offloads multiple image blocks with the highest edge intensity to a cloud server for neural SR, while using interpolation locally upscales other areas.
- **MobiSR** [31]: An on-device framework that routes incoming image patches to the appropriate model-engine pair based on the estimated upscaling difficulty of the patch to balance image quality and processing speed.
- **Strawman** upscales the whole image with the referred cheapest SR model (e.g., CARN) using local GPUs.
- **RASchduler** (Resource-Aware Scheduler) offloads tiles to each device purely based on its computational capacity, regardless of the upscaling difficulty.

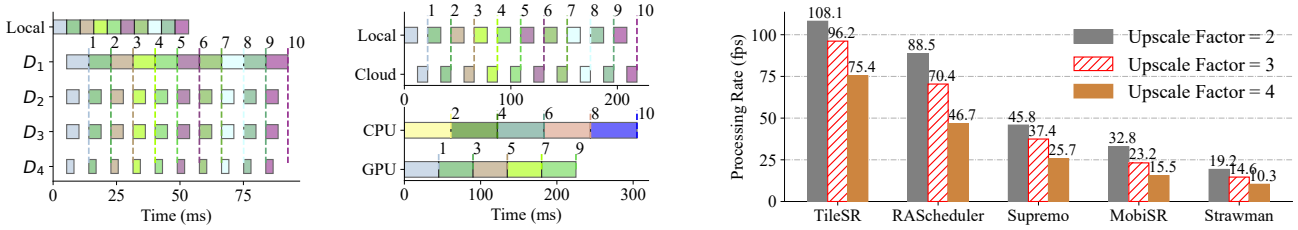
B. Performance Analysis

Fig. 5 illustrates the reward measurement of TileSR and other alternatives on the Set14 and DIV2K datasets. Fig. 5(a) shows the normalized reward of TileSR to process each image in Set14, with improvements of 6.43%, 9.36%, and 14.18% in average normalized reward compared to MobiSR, Supremo and Strawman. In particular, for frames 1, 7 and 8, TileSR yields more than 15% reward compared to other approaches. Then, in Fig. 5(b) we test these methods on the validation data that contains 100 images in DIV2K. TileSR allows more than 70% of the images to achieve at least 0.4 reward, which is higher than RAScheduler, MobiSR, Supremo, and Strawman (i.e., 58%, 45%, 31%, and 28%, respectively). In addition, more than 28% of images get at least 0.5 reward using TileSR.

Tradeoff between Latency and Quality: Next, we conduct an analysis of TileSR's performance with a focus on SR quality and processing latency. As Fig. 6(a) shows, given upscaling factors x2, TileSR impressively reduces response latency by 17.77%, 57.63%, 69.66%, and 82.2% on average compared to RAScheduler, Supremo, MobiSR, and Strawman, respectively. Besides, TileSR achieves a 2.38%, 9%, 3.2%, and 10.57% higher PSNR. Supremo offloads selected tiles to a remote cloud, causing unpredictable latency; MobiSR utilizes multiple local processors for parallel inference, resulting in long inference times to perform frame-level SR. Strawman runs the cheapest model on the entire frame, leading to long inference times and lower SR quality. Given the upscaling factor x2 in Fig. 6(b), despite a significant decrease in SR quality and latency, TileSR still outperforms other alternatives.

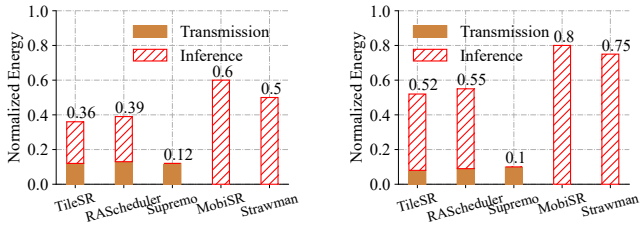
Processing Rate (fps): As demonstrated in Fig. 7(a), we further illustrate the time efficiency by comparing the concrete executions of TileSR, Supremo, and MobiSR under the upscaling factor x2. In Fig. 7(a), TileSR initiates the new super-resolution process only after the local device has received all upscaled tiles from each participating device, resulting in an average latency of 9.254 ms. For Supremo (top in Fig. 7(b)), its latency mainly stems from data transfer, totaling 21.84 ms to process each frame. On the other hand, MobiSR (bottom in Fig. 7(b)) utilizes both GPU and CPU to perform SR in parallel, leading to an average latency of 30.5 ms. We calculate the corresponding processing rate in frames per second (fps) as shown in Fig. 7(c). It is evident that TileSR achieves processing rates of 0.22x, 1.36x, 2.3x, and 4.62x compared to RAScheduler, Supremo, MobiSR, and Strawman, respectively.

Energy Efficiency: In Fig. 8, we present the energy consumption analysis, which primarily results from data transfer between the local device and other devices or a remote cloud, as well as the concrete neural inference. Fig. 8(a) demonstrates that TileSR achieves nearly 40% and 28% energy savings compared to MobiSR and Strawman, respectively. Both MobiSR and Strawman perform SR on the entire frame without any



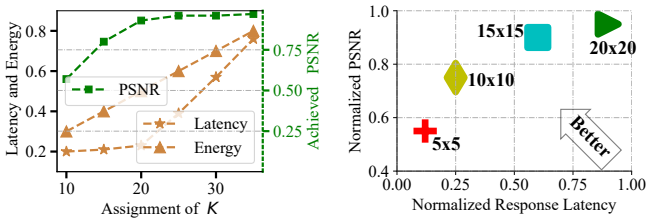
(a) TileSR execution (b) Supremo and MobiSR executions (c) Processing rate in fps of Tile and other alternatives

Fig. 7. Latency efficiency and corresponding processing rate in fps of TileSR compared to other alternatives.



(a) Under upscaling factor x2 (b) Under upscaling factor x4

Fig. 8. Impact of tile number and tile size on the response latency and quality.



(a) Impact of tile quantity K (b) Impact of tile size

Fig. 9. Impact of tile quantity and size on the response latency and quality.

inter-device data transfer. On the other hand, Supremo offloads the inference task to the cloud, but it overlooks the cloud-side power consumption. It is worth noting that a higher upscaling factor corresponds to a lower resolution of the original image, resulting in reduced data transmission consumption in Fig. 8(a). However, it also leads to a significant increase in neural inference consumption. Specifically, TileSR saves 35% and 30.67% energy compared to MobiSR and Strawman, respectively, highlighting its superior energy efficiency.

C. Sensitivity to Tile Settings

Impact of Tile Quantity K: We conducted experiments by setting K to $\{10, 15, 20, 25, 30, 35\}$, then run TileSR in the validation data in DIV2K, and calculate the average SR quality, response latency, and energy consumption. Fig. 9(a) shows the normalized values under the upscaling factor x2. As K increases, TileSR first shows significant quality gains but gradually achieves diminishing returns. This is attributed to the fact that increasing K results in more lower-mPV tiles being offloaded, whereas they yield approximate quality using local interpolation-based methods. As for energy consumption, it exhibits a proportional relationship with K. Additionally, the response latency is hardly affected when increasing K from 10 to 20 but presents a proportional relationship when K exceeds 20. This is mainly because the device has sufficient resources to execute all tiles in parallel when K is small, but it needs to wait when unloading an excessive number of tiles.

Impact of Tile Size: We varied the tile size from the set $\{$5x5, 10x10, 15x15, 20x20$\}$ and analyzed the corresponding response latency and SR quality, as depicted in Fig. 9(b). Clearly, enlarging the tile size leads to higher response latency, with latency ranging from 0.12 for a 5x5 size to 0.95 for a 20x20 size. As for SR quality, it initially improves as the size increases (e.g., from 5x5 to 10x10), but the benefit gradually becomes incremental (e.g., from 15x15 to 20x20). This behavior is largely due to the fact that excessively large tile sizes may contain substantial lower-mPV regions, which provide little quality gains from neural super-resolution.

VII. Related Work

Efficient Image Super-Resolution: Recent studies have proposed efficiency-optimized model architectures, incorporating various prominent techniques. These approaches range from avoiding the computation of large feature maps [32] and reducing upsampling costs by employing pixel-shuffle layers [33], [34], to utilizing more efficient blocks like group convolutions [8] and channel splitting [4], [35]. Furthermore, the field of neural architecture search for efficient SR is gaining traction [36]–[38]. Despite these advancements, the on-device execution of these models still remains impractical, leading to the need for numerous system-based solutions [39].

On-Device Deep Super-Resolution: Many works aim to deploy SR models on mobile devices. One approach [41], [43] utilizes the heterogeneous processors within a device. For example, MobiSR [41] quantifies the difficulty of each patch and dispatches it to the appropriate processor. Another approach [42] uses image feature to reduce computation. For instance, NEMO [42] utilizes inter-frame dependencies to cache and reuse previously super-resolved patches. SplitSR [44] combines efficient model design with compiler optimizations to enhance CPU-based SR, and XLSR [45] introduces a hand-crafted lightweight model. To mitigate the quality degradation, some works reduce utilization rate of NPUs [41], [42], [44], resulting in reduced efficiency compared to NPU-only execution. An attempt to bridge this gap and allow existing techniques to leverage the full capabilities of modern NPUs can be found in [40], focusing on smartphones [46]–[49]. However, achieving a perfect tradeoff between SR quality and inference efficiency remains challenging for these methods. In contrast, TileSR meets both latency and quality requirements effectively.

Hardware Acceleration: Several research works have explored the design of custom hardware architectures to ef-

ficiently execute CNN workloads in resource- and power-constrained environments [50]. In the context of SR, He et al. [51] proposed a highly optimized FPGA-based hardware accelerator specifically tailored to the FSRCNN [52] network. Additionally, Kim et al. [53] adopted a hardware-software co-design method to develop a CNN-based SR model and implemented it on an FPGA-based platform. However, TileSR gets real-time SR without relying on any hardware accelerator.

VIII. Conclusion

To overcome the resource barrier of a single mobile device, we introduce TileSR, a collaborative image SR framework that leverages multiple accessible mobile devices. Upon receiving an incoming image, TileSR uniformly divides it into multiple tiles and selects the top-K tiles with the highest upscaling difficulties. Then TileSR incorporates a tile scheduling algorithm based on the multi-agent multi-armed bandit, which effectively obtains accurate offload rewards through an exploration phase, determines the tile packing decision, and exploits this solution to schedule the selected tiles. Rigorous proof ensures a sub-linear regret for reward. The experimental results show the superiority of TileSR compared to other designs.

Appendix

A. Proof of Lemma 1

Proof. We first denote event set $\mathcal{E}$ as $\{\exists k \in \mathcal{K}, d \in \mathcal{D}, |\tilde{r}^{\pi}_{k,d} - r^{\pi}_{k,d}| > 2\underline{\mu}/(9\overline{K})\}$. In Alg. 2, the group-based scheme allows each tile $k \in \mathcal{K}$ to receive reward observations from the same device $d \in \mathcal{D}$ at least $\underline{N} \geq \frac{T_{explore}}{D}\pi \geq \frac{81\overline{K}^2(\overline{r}-\underline{r})^2}{8(\underline{\mu})^2}\pi$ times after π-th epoch exploration. Thus, the error probability $\Pr(\mathcal{E}|\underline{N})$

$$\begin{aligned}\Pr(\mathcal{E}|\underline{N}) &\leq \sum\nolimits_{k\in\mathcal{K}}\sum\nolimits_{d\in\mathcal{D}} \Pr(|\tilde{r}^{\pi}_{k,d} - r^{\pi}_{k,d}| > 2\underline{\mu}/9\overline{K}) \\ &\leq KD_{k\in\mathcal{K},d\in\mathcal{D}} \Pr\left(|\tilde{r}^{\pi}_{k,d} - r^{\pi}_{k,d}| > 2\underline{\mu}/9\overline{K}\right) \\ &\overset{(a)}{\leq} 2KDe^{-\frac{8(\underline{\mu})^2\underline{N}}{81\overline{K}^2(\overline{r}-\underline{r})^2}} \leq 2KDe^{-\pi},\end{aligned}$$

where (a) is derived from Hoeffding's inequality. □

B. Proof of Lemma 2

Proof. Based on Lemma 1, for the packing phase in Alg. 3, the probability of $|\tilde{r}^{\pi}_{k,d} - r^{\pi}_{k,d}| \leq \frac{2\underline{\mu}}{9\overline{K}}$ is at least $1 - 2KDe^{-\pi}$. Then, we denote $\rho_{k,d} = \tilde{r}^{\pi}_{k,d} - r^{\pi}_{k,d}$. In addition, similarly to the packing output $\widetilde{\Xi}^{(\pi)}$ given $\tilde{r}^{\pi}_{k,d}$, we denote the the packing output under the expected rewards $\{r_{k,d}, k \in \mathcal{K}, d \in \mathcal{D}\}$ as $\widehat{\Xi}^{(\pi)} = \{\hat{\chi}^{(\pi)}_k, k \in \mathcal{K}\}$ and the corresponding decision derived from $\widehat{\Xi}^{(\pi)}$ is $\hat{x} = \{\hat{x}_t, t \in \mathcal{T}\}$. Now, for each device $d \in \mathcal{D}$, we analyze the reward improvements $\Delta r_{k,d}$ and $\Delta\tilde{r}_{k,d}$ under the packing schemes $\widetilde{\Xi}^{(\pi)}$ and $\widehat{\Xi}^{(\pi)}$, respectively.

Under the packing scheme $\widetilde{\Xi}^{(\pi)}$, we introduce two metrics,

$$\mathrm{OPT}_d(x) = \sum\nolimits_{k\in\mathcal{K}} \Delta r_{k,d} = \sum\nolimits_{k\in\mathcal{K}} r^{\pi}_{k,d} - r^{\pi}_{k,\chi^{(\pi)}_k\in\widetilde{\Xi}^{(\pi)}}.$$

$$\widehat{\mathrm{OPT}}_d(x) = \sum\nolimits_{k\in\mathcal{K}} \Delta\tilde{r}_{k,d} = \sum\nolimits_{k\in\mathcal{K}} \tilde{r}^{\pi}_{k,d} - \tilde{r}^{\pi}_{k,\chi^{(\pi)}_k\in\widetilde{\Xi}^{(\pi)}},$$

Then, based on the above probability $1 - 2KDe^{-\pi}$, we get

$$\begin{aligned}\mathrm{OPT}_d(x) - \widehat{\mathrm{OPT}}_d(x) &\leq \underline{K}\max\{|\rho_{k,d}| + |\rho_{k,\chi^{(\pi)}_k\in\widetilde{\Xi}^{(\pi)}}|\} \\ &\leq (\overline{K}4\underline{\mu})/(9\overline{K}) \leq (4\underline{\mu})/9.\end{aligned}$$

Under the packing scheme $\widehat{\Xi}^{(\pi)} = \{\hat{\chi}^{(\pi)}_k, k \in \mathcal{K}\}$, we have,

$$\mathrm{OPT}_d(\hat{x}) = \sum\nolimits_{k\in\mathcal{K}} \Delta r_{k,d} = \sum\nolimits_{k\in\mathcal{K}} r^{\pi}_{k,d} - r^{\pi}_{k,\hat{\chi}^{(\pi)}_k\in\widehat{\Xi}^{(\pi)}},$$

$$\widehat{\mathrm{OPT}}_d(\hat{x}) = \sum\nolimits_{k\in\mathcal{K}} \Delta\tilde{r}_{k,d} = \sum\nolimits_{k\in\mathcal{K}} \tilde{r}^{\pi}_{k,d} - \tilde{r}^{\pi}_{k,\hat{\chi}^{(\pi)}_k\in\widehat{\Xi}^{(\pi)}}.$$

Then, similarly we get $\mathrm{OPT}_d(\hat{x}) - \widehat{\mathrm{OPT}}_d(\hat{x}) \leq (4\underline{\mu})/9$, thus

$$\begin{aligned}&\mathrm{OPT}_d(\hat{x}) - \mathrm{OPT}_d(x) \\ &= \mathrm{OPT}_d(\hat{x}) - \widehat{\mathrm{OPT}}_d(\hat{x}) + \widehat{\mathrm{OPT}}_d(\hat{x}) - \mathrm{OPT}_d(x) \\ &\overset{(b)}{\leq} \mathrm{OPT}_d(\hat{x}) - \widehat{\mathrm{OPT}}_d(\hat{x}) + \widehat{\mathrm{OPT}}_d(x) - \mathrm{OPT}_d(x) \overset{(c)}{\leq} 8\underline{\mu}/9,\end{aligned}$$

where (b) is derived since $\widehat{\mathrm{OPT}}_d(\hat{x}) \leq \mathrm{OPT}_d(x)$ always holds given the input $\{\Delta\tilde{r}_{k,d}, k \in \mathcal{K}\}$ for device d. Besides, based on the definition of $\underline{\mu}$, we get $\mathrm{OPT}_d(\hat{x}) - \mathrm{OPT}_d(x) \geq \underline{\mu}, \forall d \in \mathcal{D}$, which contradicts (c), hence it must hold that $\widehat{\mathrm{OPT}}_d(\hat{x}) = \mathrm{OPT}_d(x), \forall d \in \mathcal{D}$. Thus, the output $\widetilde{\Xi}^{(\pi)}$ under the explored estimates improvement $\{\Delta\tilde{r}_{k,d}\}$ is the same as $\widehat{\Xi}^{(\pi)}$ derived from $\{\Delta r_{k,d}\}$ if $|\tilde{r}^{\tau}_{k,d} - r_{k,d}| \leq 2\underline{\mu}/9\overline{K}$ holds. □

C. Proof of Theorem 1

Proof. Recall that π_T is the last epoch and $T_{explore}$ is set to be 2^{π} for epoch π. We have $T \geq \sum_{\pi=1}^{\pi_T} 2^{\pi} = 2^{\pi_T+1} - 2$, hence $\pi_T \leq \log(T+2)$. The regret of $\mathbb{P}_2$ consists of the regrets in the exploration, packing and exploitation phase, thus,

$$\begin{aligned}\mathcal{R}(T) &\leq \sum\nolimits_{\pi=1}^{\pi_T}(T_{explore}K\overline{r} + KD\overline{r} + 2KDe^{-\pi} + T_{exploit}K\overline{r}) \\ &\leq (T_{explore}K\overline{r} + KD\overline{r})\pi_T + (K^2D\underline{r})\sum\nolimits_{\pi=1}^{\pi_T}(2^{\pi}e^{-\pi}) \\ &\leq (T_{explore}K\overline{r} + KD\overline{r})\log_2(T+2) + 4K^2D\underline{r} \\ &= O(log_2 T).\end{aligned}$$

Above all, we bound the **regret** as $\mathcal{R}(T) \leq O(log_2 T)$. □

D. Proof of Lemma 3

Proof. This lemma is proved by induction on the number of devices. Let $R(\widehat{S})$ represent the profit under decision $\widehat{S}$.

For the inductive step, we denote the solution after the d-th recursion as $\widehat{S_{d+1}}$ and assume that $\widehat{S_{d+1}}$ has a 2-approximation for R_{d+1}. We shall prove that $\widehat{S_d}$ also has a 2-approximation with respect to R_d. Considering the R^2_d is identical to R_{d+1}, except that it contains a column whose values are 0, $\widehat{S_{d+1}}$ also has an 2-approximation to R^2_d. Since $\widehat{S_{d+1}}$ contains the tiles in $\widehat{S_d}$, $\widehat{S_d}$ is also a 2-approximation with respect to R^2_d. Next we shall prove $\widehat{S_d}$ is 2-approximate to R^1_d.

Matrix R^1_d includes (a) a column, i.e, the first column in R_d, (b) the rows corresponding the tiles in S_d, and (c) the remaining entries. Only components (a) and (b) contribute profit to a decision. (a) is used as input for Π, therefore the best possible solution S^*_a gains at most $R^1_d(S_d)$ with respect to (a). As the profits of each row in (b) are identical, the best possible solution S^*_b also gains at most $R^1_d(S_d)$ with respect to (b). This implies that S_d has a 2-approximation with respect to R^1_d. Since S_d are a subset of tiles of $\widehat{S_d}$, we have $R^1_d(\widehat{S_d}) \geq R^1_d(S_d)$, therefore $\widehat{S_d}$ is 2-approximate to R^1_d.

As defined, $R_d = R^1_d + R^2_d$, $R_d(\widehat{S_d}) = (R^1_d + R^2_d)(\widehat{S_d}) \geq 2R^1_d(S^*_a) + 2R^2_d(S^*_b) \geq 2(R^1_d(S^*) + R^2_d(S^*)) = 2R_d(S^*)$, where S^* indicates the optimal solution. □

REFERENCES

[1] Facebook official website. https://www.facebook.com/.

[2] Instagram official website. https://www.instagram.com/.

[3] Reddit official website. https://www.reddit.com/.

[4] Z. Hui, X. Wang, and X. Gao. 2018. "Fast and Accurate Single Image Super-Resolution via Information Distillation Network," in *IEEE CVPR*, 2018, pp. 723–731.

[5] Y. Zhang, K.g Li, K. Li, L. Wang, B. Zhong, Y. Fu, "Image Super-Resolution Using Very Deep Residual Channel Attention Networks," in *ECCV*, 2018, pp. 286-301

[6] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, Y. Fu. 2018. "Residual Dense Network for Image Super-Resolution," in *IEEE CVPR*, 2018, pp. 2472-2481.

[7] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *IEEE CVPR Workshops*, 2017, pp. 1132–1140.

[8] N. Ahn, B. Kang, and K. A. Sohn, "Fast, accurate, and lightweight super resolution with cascading residual network," in *ECCV*, 2018, pp. 252–268.

[9] J. C. Li, F. M. Fang, and K. F. Mei, "Multiscale residual network for image super-resolution," in *ECCV*, 2018, pp. 517–532.

[10] Y. Chen, S. Zhang, Y. Jin, Z. Qian, M. Xiao, N. Chen, and Z. Ma, "Learning for crowdsourcing: Online dispatch for video analytics with guarantee," in *IEEE INFOCOM*, 2022, pp. 1908–1917.

[11] Z. Lu, S. Rallapalli, K. Chan, and T. La Porta, "Modeling the resource requirements of convolutional neural networks on mobile devices," in *ACM MM*, 2017, pp. 1663–1671.

[12] T. X. Tran, K. Chan, and D. Pompili, "Costa: Cost-aware service caching and task offloading assignment in mobile-edge computing," in *IEEE SECON*, 2019, pp. 1–9.

[13] L. Li, M. Pal, and Y. R. Yang, "Proportional fairness in multi-rate wireless lans," in *IEEE INFOCOM*, 2008, pp. 1004–1012.

[14] R. Srikant and L. Ying, "Communication networks: an optimization, control, and stochastic networks perspective," in *Cambridge University Press*, 2013.

[15] G. Xiong, S. Wang, G. Yan, and J. Li, "Reinforcement learning for dynamic dimensioning of cloud caches: A restless bandit approach," in *IEEE INFOCOM*, 2022, pp. 2108–2117.

[16] G. Gao, J. Wu, M. Xiao, and G. Chen, "Combinatorial multi-armed bandit based unknown worker recruitment in heterogeneous crowdsensing," in *IEEE INFOCOM*, 2020, pp. 179–188.

[17] Y. Song and H. Jin, "Minimizing entropy for crowdsourcing with combinatorial multi-armed bandit," in *IEEE INFOCOM*, 2021, pp. 1–10.

[18] J. Yang and S. Ren, "Bandit learning with predicted context: Regret analysis and selective context query," in *IEEE INFOCOM*, 2021, pp. 1–10.

[19] A. R. Kan, L. Stougie, and C. Vercellis, "A class of generalized greedy algorithms for the multi-knapsack problem," *Discrete Applied Mathematics*, vol. 42, no. 2-3, 1993, pp. 279–290.

[20] K. Pak and R. Dekker, "Cargo revenue management: Bid-prices for a 0-1 multi knapsack problem," Available at *SSRN* 594991, 2004.

[21] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Information Processing Letters*, vol. 100, no. 4, 2006, pp. 162–166.

[22] R. Neapolitan and K. Naimipour, "Foundations of algorithms using java pseudo code, jones and bartleet publishers," ISBN-978-443-5000, Tech. Rep., 2004.

[23] Aitek Technology. https://www.aitek.com.tw/EN/shouye.html.

[24] DIV2K dataset. https://data.vision.ee.ethz.ch/cvl/DIV2K/.

[25] Set5 dataset. http://people.rennes.inria.fr/Aline.Roumy/results /SR_BMVC12.html.

[26] Set14 dataset. https://github.com/jbhuang0604/SelfExSR.

[27] J. Yi, S. Kim, J. Kim, and S. Choi, "Supremo: Cloud-assisted low-latency super-resolution in mobile devices," *IEEE TMC*, vol. 21, no. 5, 2022, pp. 1847–1860.

[28] S. Huang, J. Xie and M. M. A. Muslam, "A Cloud Computing Based Deep Compression Framework for UHD Video Delivery," *IEEE TCC*, vol. 11, no. 2, 2023, pp. 1562-1574.

[29] Y. Wang, W. Wang, D. Liu, X. Jin, J. Jiang and K. Chen, "Enabling Edge-Cloud Video Analytics for Robotics Applications," *IEEE TCC*, vol. 11, no. 2, 2023, pp. 1500-1513.

[30] W. Zhang, Z. He, L. Liu, Z. Jia, Y. L, M. Gruteser, D. Raychaudhuri and Y. Zhang. "Elf: Accelerate High-resolution Mobile Deep Vision with Content-aware Parallel Offloading," in *ACM MobiCom*, 2021, pp. 201-214.

[31] R. Lee, S. I. Venieris, L. Dudziak, S. Bhattacharya, and N. D. Lane, "MobiSR: Efficient on-device super-resolution through heterogeneous mobile processors," in *ACM MobiCom*, 2019, pp. 1–16.

[32] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE TPAMI*, vol. 38, no. 2, 2016, pp. 295-307.

[33] W. Shi, J. Caballero, F. Huszar, 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," in *IEEE CVPR*, 2016, pp. 1874–1883.

[34] 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 *ECCV Workshops*, 2018.

[35] Z. Hui, X. Gao, Y. Yang, and X. Wang, "Lightweight Image Super-Resolution with Information Multi-distillation Network," in *ACM MM*, 2019, pp. 2024-2032.

[36] R. Lee, L. Dudziak, M. Abdelfattah, S. I. Venieris, H. Kim, H. Wen, and N. Lane, "Journey Towards Tiny Perceptual Super-Resolution," in *ECCV*, 2020, pp. 85-102.

[37] X. Chu, B. Zhang, H. Ma, R. Xu and Q. Li, "Fast, Accurate and Lightweight Super-Resolution with Neural Architecture Search," in *IEEE ICPR*, 2021, pp. 59-64.

[38] D. Song, C. Xu, X. Jia, Y. Chen, C. Xu, and Y. Wang, "Efficient Residual Dense Block Search for Image Super-Resolution," in *AAAI*, 2020, pp. 12007-12014.

[39] 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 CSUR*, vol. 54, no. 8, 2021, pp. 1-30.

[40] S. I. Venieris, M. Almeida, R. Lee and N. D. Lane, "NAWQ-SR: A Hybrid-Precision NPU Engine for Efficient On-Device Super-Resolution," in *IEEE TMC*, 2023, pp. 1-15.

[41] R. Lee, S. I. Venieris, L. Dudziak, S. Bhattacharya, and N. D. Lane, "MobiSR: Efficient on-device super-resolution through heterogeneous mobile processors," in *ACM MobiCom*, 2019, pp. 1–16.

[42] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han, "NEMO: Enabling Neural-enhanced Video Streaming on Commodity Mobile Devices," in *ACM MobiCom*, 2020, pp. 1-14.

[43] J. Yi, S. Kim, J. Kim, and S. Choi, "Supremo: Cloud-assisted low-latency super-resolution in mobile devices," *IEEE TMC*, vol. 21, no. 5, 2022, pp. 1847–1860.

[44] 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," *ACM IMWUT*, vol. 5, no. 1, 2021, pp. 1-20.

[45] M. Ayazoglu, "Extremely Lightweight Quantization Robust Real- Time Single-Image Super Resolution for Mobile Devices," in *IEEE CVPR Workshops*, 2021, pp. 2472–2479.

[46] 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 *EMDL Workshops*, 2019.

[47] 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 IMC*, 2021, pp. 658-672.

[48] A. Ignatov et al., "AI Benchmark: All About Deep Learning on Smartphones in 2019," in *IEEE ICCV Workshops*, 2019.

[49] Qualcomm, "Snapdragon Neural Processing Engine," https://developer.qualcomm.com/sites/default/files/docs/snpe/.

[50] S. I. Venieris and C. -S. Bouganis, "fpgaConvNet: Mapping Regular and Irregular Convolutional Neural Networks on FPGAs," in *IEEE TNNLS*, vol. 30, no. 2, 2019, pp. 326-342.

[51] Z. He, H. Huang, M. Jiang, Y. Bai, and G. Luo, "FPGA-Based Real-Time Super-Resolution System for Ultra High Definition Videos," in *IEEE FCCM*, 2018, pp. 181–188.

[52] C. Dong, C.e Loy, and X. Tang, "Accelerating the Super-Resolution Convolutional Neural Network," In *ECCV*, 2016, pp. 391-407.

[53] Y. Kim, J. Choi, and M. Kim, "A Real-Time Convolutional Neural Network for Super-Resolution on FPGA with Applications to 4K UHD 60 fps Video Services," in *IEEE TCSVT*, 2018, pp. 2521-2534.