IEEE INFOCOM
IEEE International Conference on Computer Communications
20-23 May 2024 // Vancouver, Canada

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

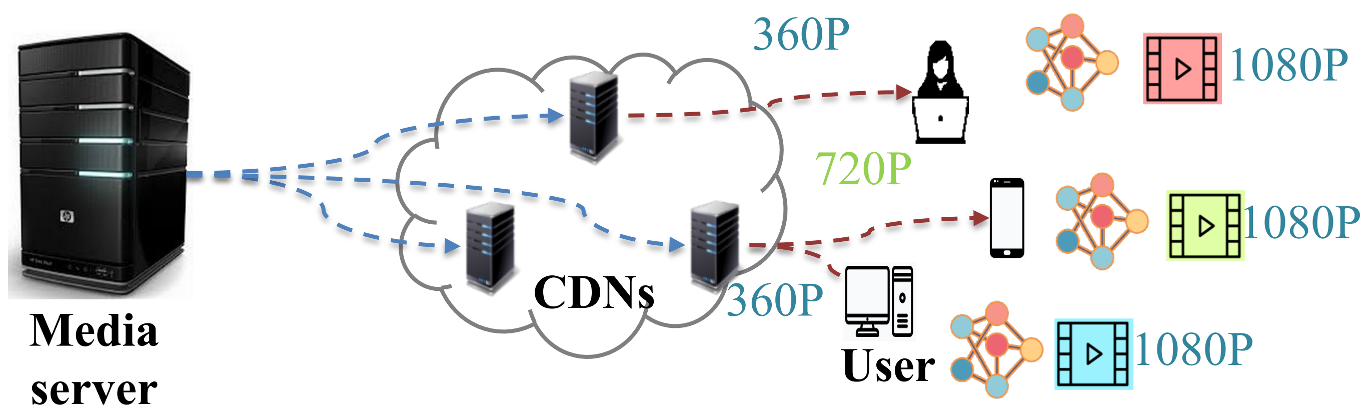
Ning Chen, Sheng Zhang, Yu Liang, Jie Wu, Yu Chen,
Yuting Yan, Zhuzhong Qian and Sanglu Lu

Outline

- Problem Background
  - Edge Collaborative SR with Parallel Offloading
    - **Key steps**: image dividing, tile-based offloading
  - Challenges
    - SR difficulty, device priori information
- Related Works
- Motivations
- System Model
  - TileSR: Parallel Offloading Tiles to Accelerate Inference
- Experiments

Background: **Single**-Device SR

- On-device video super-resolution
  - **Unreliable network** leads to **low-quality** video
  - Local SR to enhance the video quality



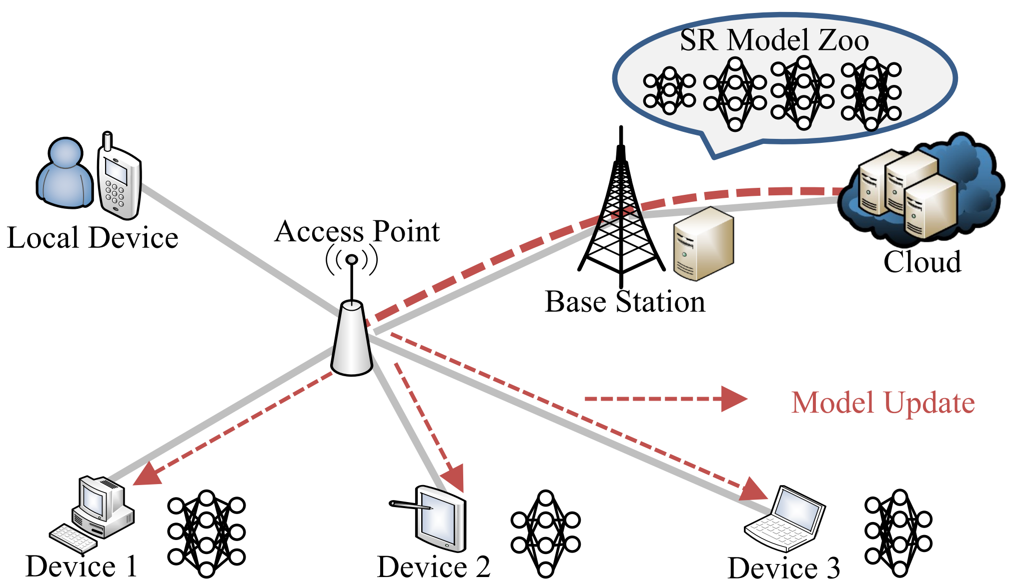
Background: **Single**-Device SR

- On-device video super-resolution
- Challenges: local resource constrained
  – Computation → long inference delay
  – Memory → hard to store intermediate data
  – Energy → hard to support long-term inference

SR model	270p (s)	360p (s)	540p (s)	720p (s)
MSRN X2	0.075	0.167	0.302	0.922
MSRN X3	0.083	0.182	0.321	1.003
MSRN X4	0.091	0.237	0.404	1.118
RCAN X2	0.269	0.595	1.057	3.191
RCAN X3	0.277	0.610	1.085	3.261
RCAN X4	0.287	0.632	1.149	3.356

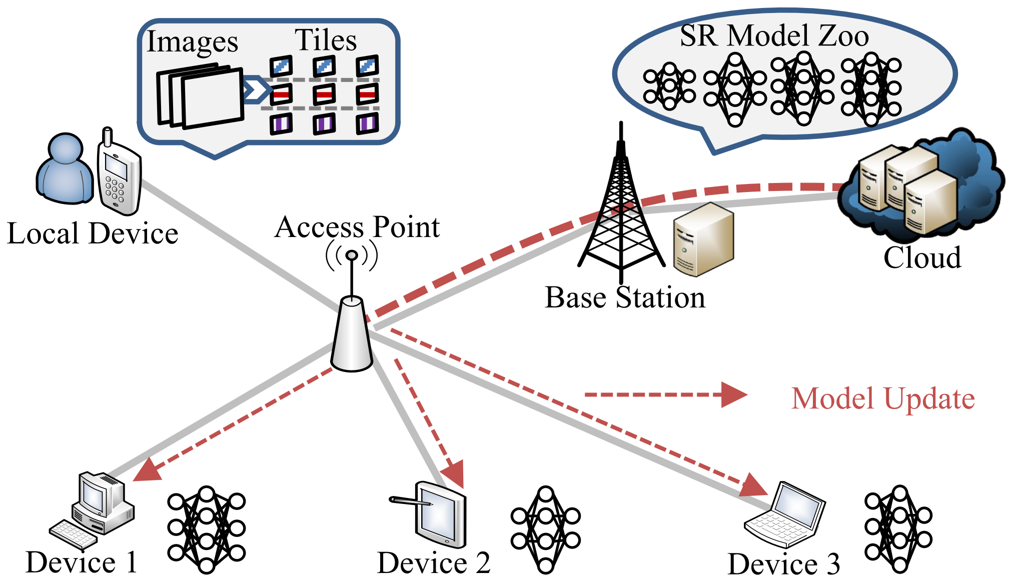
Background: **Multi**-Device SR

- Multi-device inference for image/video SR
  - Model downloading from cloud



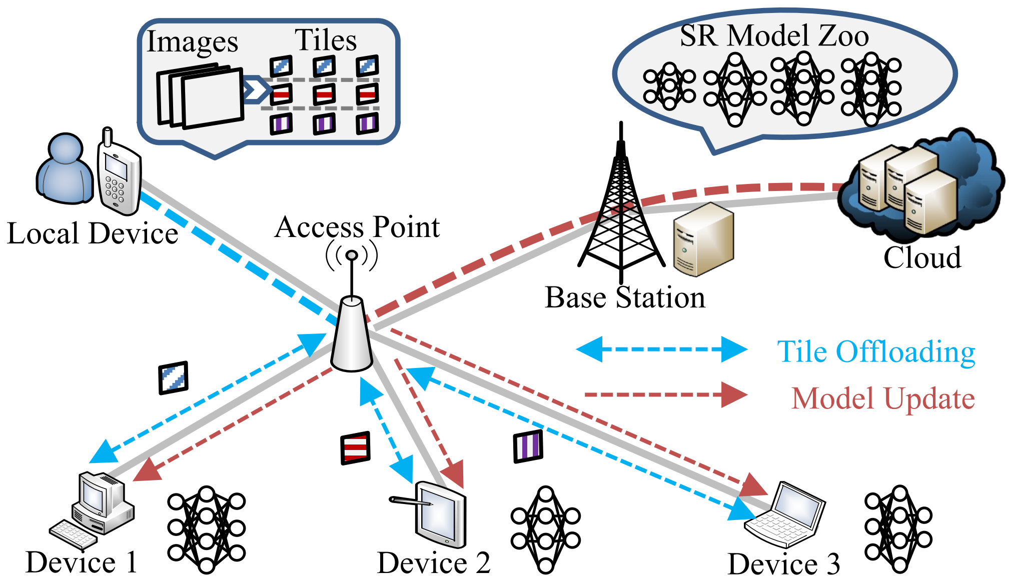
Background: **Multi**-Device SR

- Multi-device inference for image/video SR
  - Model downloading from cloud
  - Image dividing → multiple tiles



Background: **Multi**-Device SR

- Multi-device collaboration for image/video SR
  - Model downloading from cloud
  - Image dividing → multiple tiles
  - Parallel offloading → tile inference

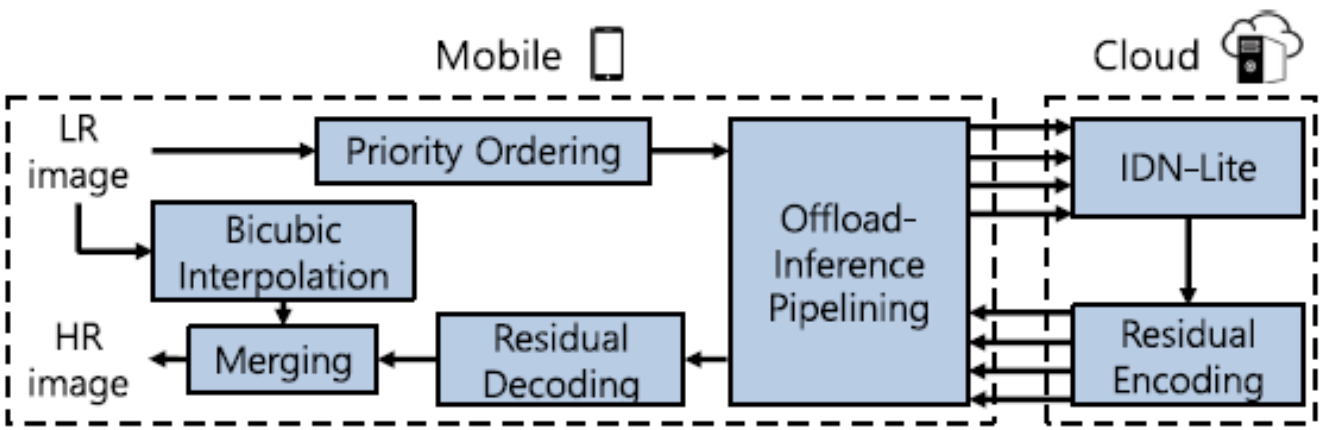


Background: **Multi**-Device SR

- Multi-device collaboration for image/video SR
- Challenges
  - How to divide? even or uneven?
  - How to offload? without prior offload reward

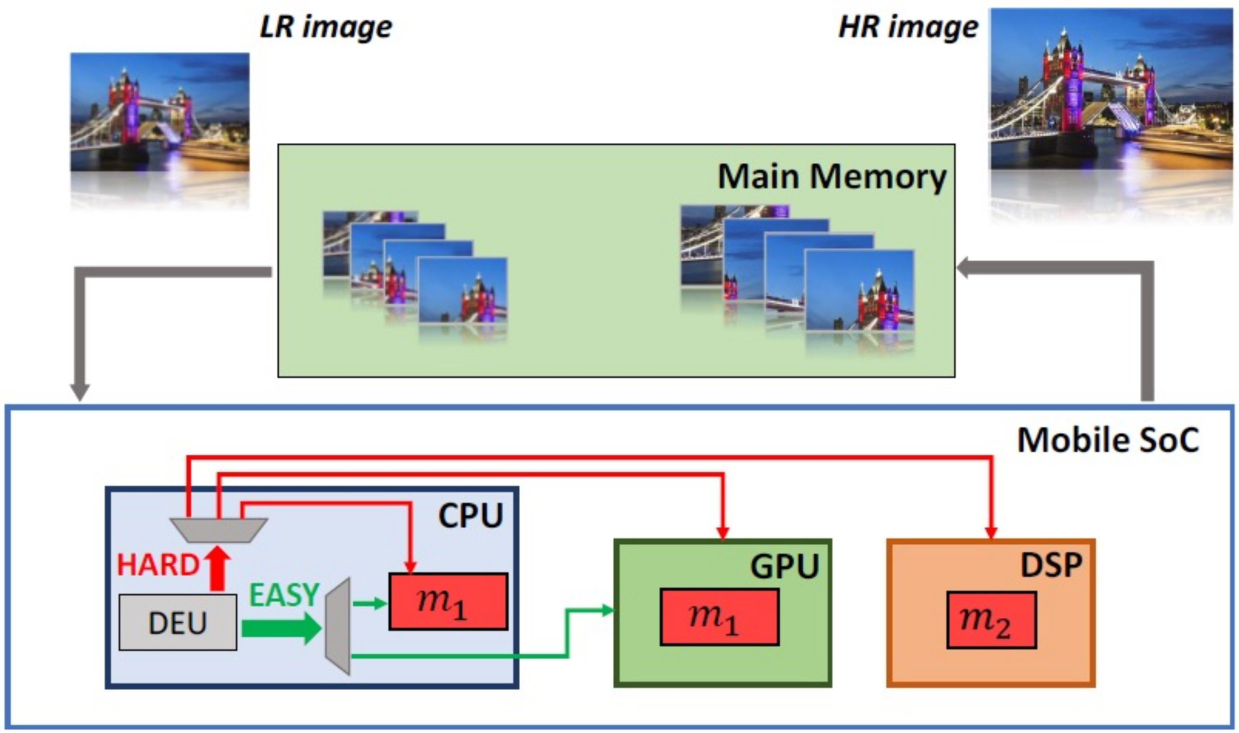
Related Works

- Supremo *TMC 2022*
  - **Layer**: edge-cloud collaboration
  - **Methods**: based on the "edge" feature



Related Works

- MobiSR *MobiCom 2019*
  - **Methods**: dispatch images to diverse processors

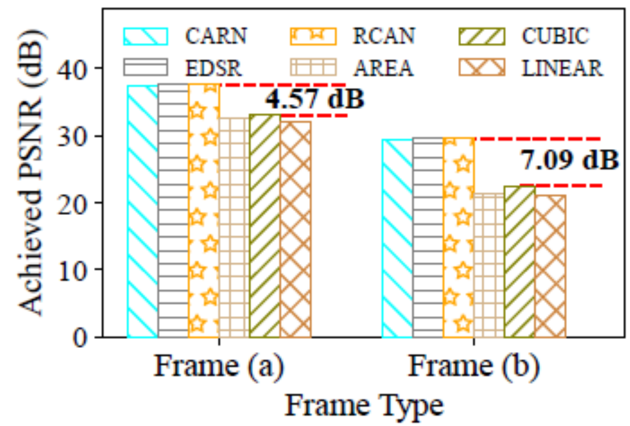


Motivation: Difficulty Analysis

- Visual structural complexity

(a) Structured image

(b) Unstructured image



  - Utilize two methods, interpolation and CNN-based
    - Higher structure, higher upscale quality (PSNR)
    - Higher structure, it benefits less from CNN inference

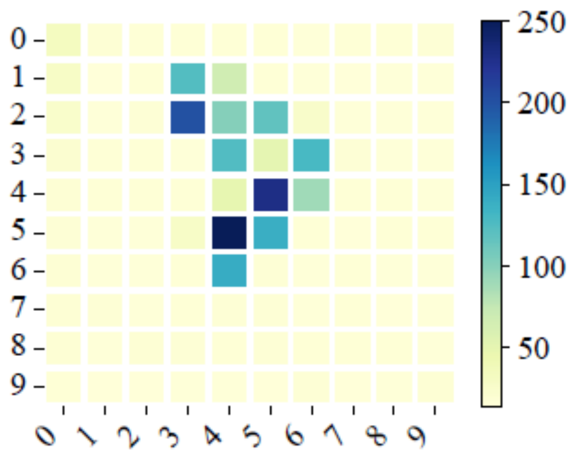
How to measure the structure information?

Motivation: Difficulty Analysis

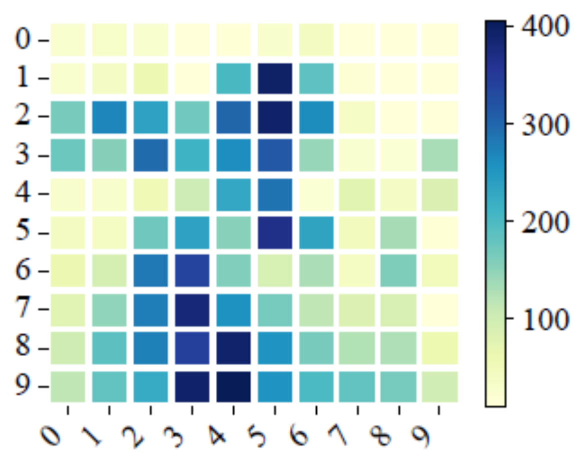
- Pixel Variant Matrix ***PV_f*** for image f,

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

- Mean Pixel Variant $mPV_f = \sum_{i=1}^{W}\sum_{j=1}^{H} PV_f[i,j]/(WH)$
  - This definition applies to **tile** as well



(c) mPV heatmap for image (a)

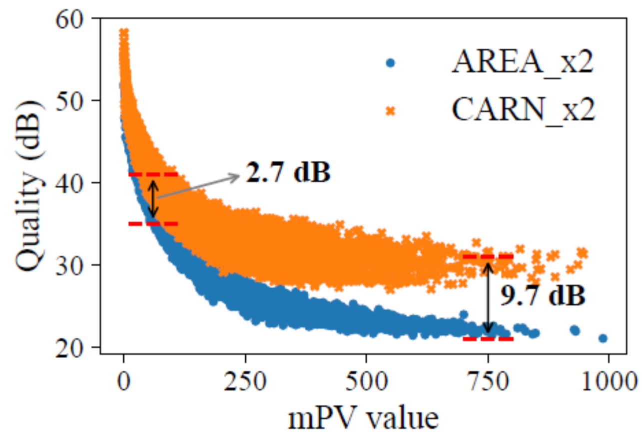


(d) mPV heatmap for image (b)

Motivation: Difficulty Analysis

- Mean Pixel Variant $mPV_f = \sum_{i=1}^{W} \sum_{j=1}^{H} PV_f[i,j]/(WH)$
- Motivative insights
  - Tiles with low-mPV achieve high and similar PSNR using CNN-based and interpolation-based methods;
  - Tiles with high-mPV get significant higher PSNR using CNN-based method than interpolation.

Dataset	DIV2K
Interpolation method	AREA_x2
CNN model	CARN_x2
Dividing method	20 x 20



Tiles Generation

- Mean Pixel Variant $mPV_f = \sum_{i=1}^{W} \sum_{j=1}^{H} PV_f[i,j]/(WH)$
- Key Idea: only offload high-mPV tiles
  - Calculate the upscaling difficulties for each tile;
  - Select the **Top-K** tiles with highest mPV;
  - **Parallel offload** the selected tiles to involved devices for CNN-based inference;
  - **Locally** upscale other tiles via interpolation.

Next we consider the parallel offloading policy!

Parallel Offloading: **System Model**

- Multi-device parallel offloading problem
  - Decision: $x_t = (x_{1,t}, \cdots, x_{K,t})$
  - Maximize the overall offload reward
  - Subject to the resource and latency constraints

$$\mathbb{P}: \max_{\{x_t | t \in \mathcal{T}\}} \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{K}} \underline{\mathbb{E}\left[\tilde{u}_{n,x_{k,t}}(t)\right]}$$

No System-side info.
i.i.d random reward

$$\text{s.t.} \sum_{k \in \mathcal{K}_{d,t}} c_{k,d} \le C_d, \forall d \in \mathcal{D}, \forall t \in \mathcal{T},$$

Resource constraint

$$\max\left\{L_{k,x_{k,t}} | k \in \mathcal{K}\right\} \le L_{max}, \forall t \in \mathcal{T}.$$

Latency constraint

Parallel Offloading: **System Model**

- Multi-device parallel offloading problem
  - Decision: $x_t = (x_{1,t}, \cdots, x_{K,t})$
  - Proportional fairness maximization

$$\mathbb{P}_1: \max_{\{x_t | t \in \mathcal{T}\}} \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{K}} \underline{\mathbb{E}\left[\tilde{r}_{k,x_{k,t}}(t)\right]}$$

No System-side info.
i.i.d random reward

$$\text{s.t. } \tilde{r}_{k,x_{k,t}}(t) = \ln\left(1 + \eta \tilde{u}_{k,x_{k,t}}(t)\right),$$

Fair reward distribution

$$\sum_{k \in \mathcal{K}_{d,t}} c_{k,d} \le C_d, \forall d \in \mathcal{D}, \forall t \in \mathcal{T},$$

Resource constraint

$$\max\left\{L_{k,x_{k,t}} | k \in \mathcal{K}\right\} \le L_{max}, \forall t \in \mathcal{T}.$$

Latency constraint

Parallel Offloading: **Algorithm**

- Reward $\tilde{r}_{n,x_{k,t}}(t)$ is uncertain but bounded
  - Decentralized multi-agent multi-armed bandit
    - **Exploration**: obtain the offloading reward mapping
    - **Packing**: get the final tile offloading scheme
    - **Exploitation**: utilize the packing result

Algorithm 1 Decentralized Online Tile Offloading

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

$R^{(\pi)} \leftarrow \left\{\tilde{r}_{k,d}^{(\pi)} = 0, \forall k \in \mathcal{K}, \forall d \in \mathcal{D}\right\}, \forall \pi \in \Pi;$

$\Xi^{(\pi)} \leftarrow \left\{\chi_k^{(\pi)} = 0, \forall k \in \mathcal{K}\right\}, \forall \pi \in \Pi;$

for epoch $\pi = 1$ to π_T **do**

    Invoke **Alg. 2**: $\tilde{R}^{(\pi)}$ = **Exploring**$\left(R^{(\pi)}, T_{explore}\right)$;

    Invoke **Alg. 3**: $\tilde{\Xi}^{(\pi)}$ = **Packing**$\left(\tilde{R}^{(\pi)}, \Xi^{(\pi)}\right)$;

    for each of the remaining $T_{exploit}$ time slots **do**

        Offloading tiles to device by **Exploiting** $\tilde{\Xi}^{(\pi)}$;

Parallel Offloading: **Algorithm**

- Reward Exploration and Exploitation
  - Group-based offloading in a round-robin fashion
    - Target device $x_{k,t} = \lfloor((k+t)\%(\underline{K}D))/\underline{K}\rfloor + 1,$
    - Reward update $\tilde{r}_{k,d}^{\pi} \leftarrow \tilde{r}_{k,d}^{\pi-1} + \left(\tilde{r}_{k,d} - \tilde{r}_{k,d}^{\pi-1}\right)/\pi$

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 $d = \lfloor((k+t)\%(\underline{K}D))/\underline{K}\rfloor + 1$;

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

Parallel Offloading: **Algorithm**

- Tile packing based on reward map
  - For each slot t, it can be modeled as

$$\mathbb{P}_2 : \max_{\boldsymbol{x}_t} \underbrace{\sum\nolimits_{k\in\mathcal{K}} \tilde{r}_{k,x_{k,t}}}_{} \quad \textbf{Value}$$

$$\text{s.t.} \sum\nolimits_{k\in\mathcal{K}_{d,t}} \underbrace{c_{k,d} \le C_d, \forall d \in \mathcal{D},}_{} \quad \textbf{Volume}$$

$$\max\left\{L_{k,x_{k,t}} | k \in \mathcal{K}\right\} \le L_{max},$$

Multi-Knapsack Problem (MKP)

Parallel Offloading: **Algorithm**

- Tile packing upon MKP
  - Iteratively update the tiles for each device

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

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

$R_1 \leftarrow \tilde{R}^{(\pi)}$, $\widehat{S} \leftarrow \varnothing$, $d \leftarrow 1$;

while $d \leq D$ **do**

    Run algorithm $\Pi(R_d, C_d)$ and return S_d;

    for tile $i \in S_d$ **do**

        if $\exists d', 1 \leq d' < d$ s.t. $i \in S_{d'}$ **then**

            Update $S_{d'} \leftarrow S_{d'} \setminus \{i\}$;

    Update decision $\widehat{S} \leftarrow \widehat{S} \bigcup \{S_d\}$;

    for tile k to K **do**

        for device j to D **do**

$$R_d^1[k,j] = \begin{cases} R_d[k,j], & \text{if } k \in S_d \text{ or } j = d, \\ 0, & \text{otherwise}, \end{cases}$$

    Set $R_d^2 \leftarrow R_d - R_d^1$, $R_{d+1} \leftarrow R_d^2$, $d \leftarrow d+1$;

Output: $\widehat{S}$

Slove single-knapsack problem for device *d*

Update the decision set

Update the profit matrix

Parallel Offloading: **Algorithm**

- Tile packing upon MKP
  - Example: 4 tiles, 3 devices

	d_1	d_2	d_3
k_1	1	2	2
k_2	2	3	4
k_3	2	2	4
k_4	1	1	4

Demand

d_1	d_2	d_3
2	3	4

Capacity

	d_1	d_2	d_3
k_1	5	1	4
k_2	2	2	2
k_3	12	25	30
k_4	20	10	14

Reward

$$\begin{array}{cc} & \begin{array}{ccc} d_1 & d_2 & d_3 \end{array} \\ R_1 = & \begin{bmatrix} \boxed{5} & 1 & 4 \\ 2 & 2 & 2 \\ 12 & 25 & 30 \\ \boxed{20} & 10 & 14 \end{bmatrix} \end{array}$$

$$R_1^1 = \begin{bmatrix} 5 & 5 & 5 \\ 2 & 0 & 0 \\ 12 & 0 & 0 \\ 20 & 20 & 20 \end{bmatrix}$$

$$R_1^2 = \begin{bmatrix} 0 & -4 & -1 \\ 0 & 2 & 2 \\ 0 & 25 & 30 \\ 0 & -10 & -6 \end{bmatrix}$$

1st recursion

$$\begin{array}{cc} & \begin{array}{cc} d_2 & d_3 \end{array} \\ R_2 = & \begin{bmatrix} -4 & -1 \\ 2 & 2 \\ \boxed{25} & 30 \\ -10 & -6 \end{bmatrix} \end{array}$$

$$R_2^1 = \begin{bmatrix} -4 & 0 \\ 2 & 0 \\ 25 & 25 \\ -10 & 0 \end{bmatrix}$$

$$R_2^2 = \begin{bmatrix} 0 & -1 \\ 0 & 2 \\ 0 & 5 \\ 0 & -6 \end{bmatrix}$$

2nd recursion

$$\begin{array}{cc} & d_3 \\ R_3 = & \begin{bmatrix} -1 \\ 2 \\ \boxed{5} \\ -6 \end{bmatrix} \end{array}$$

$$R_3^1 = \begin{bmatrix} -1 \\ 2 \\ 5 \\ -6 \end{bmatrix}$$

$$R_3^2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

3rd recursion

Experiments

- Experimental setup
  - Edge device and CNN models

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]

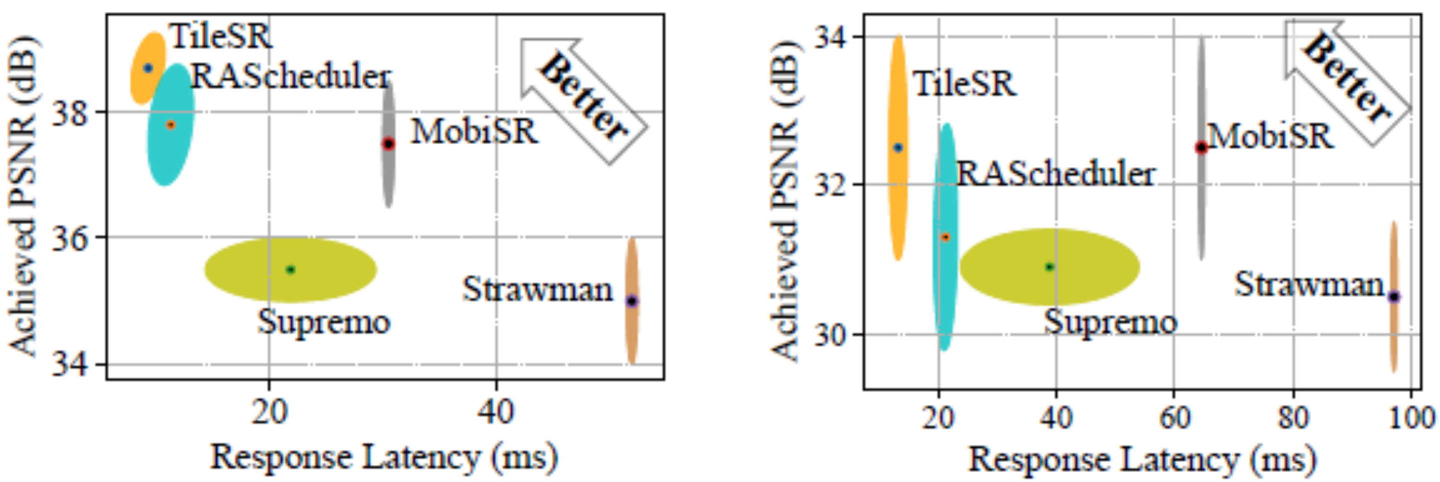
  - Datasets: DIV2K/Set5/Set13/YouTube-NBA

Experiments

- Experimental setup
  - Edge device and CNN models
  - Datasets: DIV2K/Set5/Set13/YouTube-NBA
  - Comparing schemes
    - **Supremo**: offloads tiles to cloud
    - **MobiSR**: utilizes local multiple diverse processor
    - **Strawman**: utilizes CARN for local SR
    - **RAScheduler**: offloads tiles based the computation

Experiments

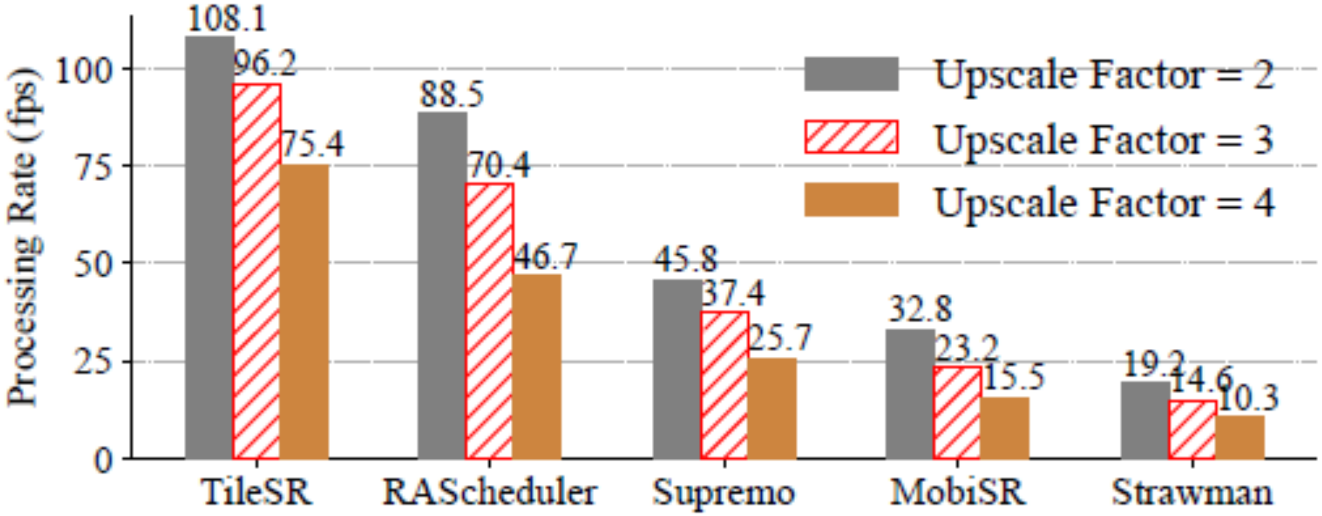
- Tradeoff between latency and quality
  - **Latency**: 17.77%, 57.63%, 69.66%, and 82.2%
  - **SR quality**: 2.38%, 9%, 3.2%, and 10.57%



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

Experiments

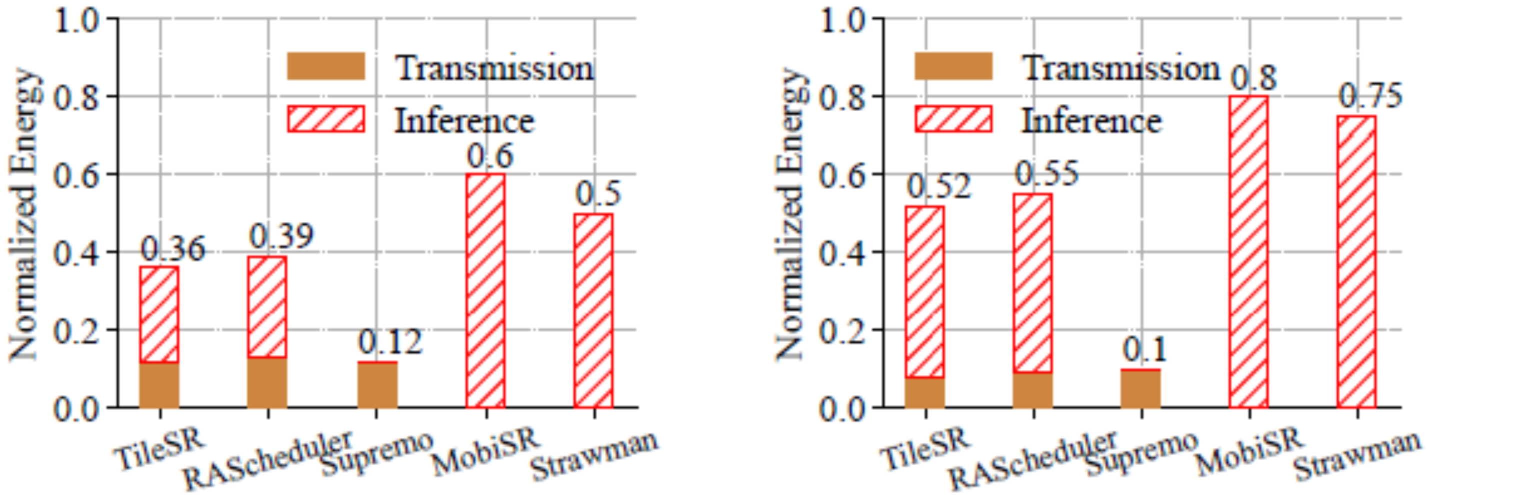
- Processing Rate (fps)
  - 0.22x, 1.36x, 2.3x, and 4.62x



(c) Processing rate in fps of Tile and other alternatives

Experiments

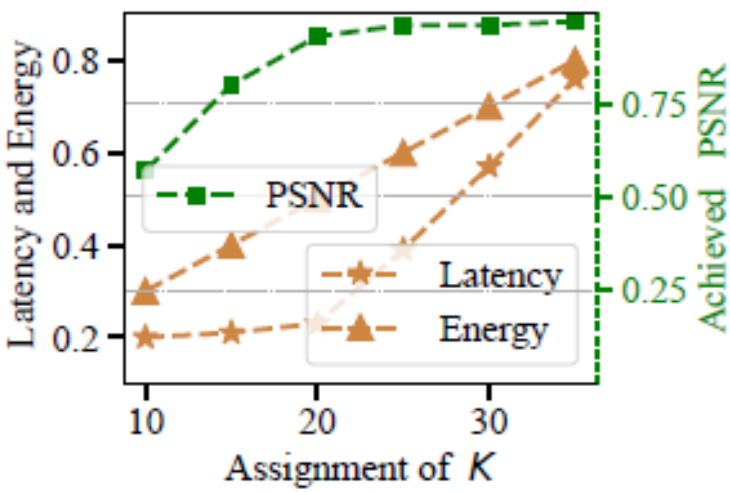
- Energy efficiency
  - 7.6%，40%，28%



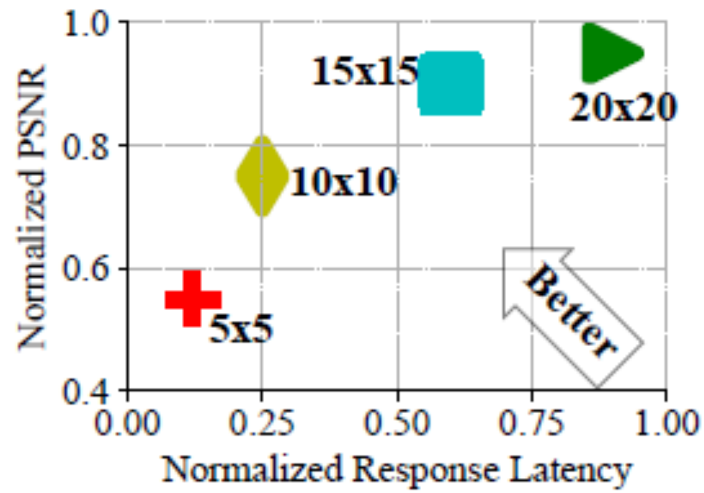
(a) Under upscaling factor x2 (b) Under upscaling factor x4

Experiments

- Tile settings
  - Tile quantity: 20；Tile size: 15x15



(a) Impact of tile quantity K



(b) Impact of tile size

Conclusion

- Multi-device inference for image SR
  - SR difficulty analysis
    - Method: Top-K tiles are offloaded for CNN inference, others are executed locally.
  - Tile parallel offloading
    - Decentralized multi-agent multi-armed bandit
    - Exploration: obtain the offloading reward map
    - Packing (MKP): iteratively solve a single-knapsack problem for each device

Thank You !

Q & A

Ning Chen

Nanjing University