A High-Performance Accelerator for Super-Resolution Processing on Embedded GPU

Wenqian Zhao[1], Qi Sun[1], Yang Bai[1], Wenbo Li[1], Haisheng Zheng[2], Bei Yu[1], Martin D.F. Wong[1]

[1]The Chinese University of Hong Kong
[2]SmartMore
{wqzhao,byu}@cse.cuhk.edu.hk

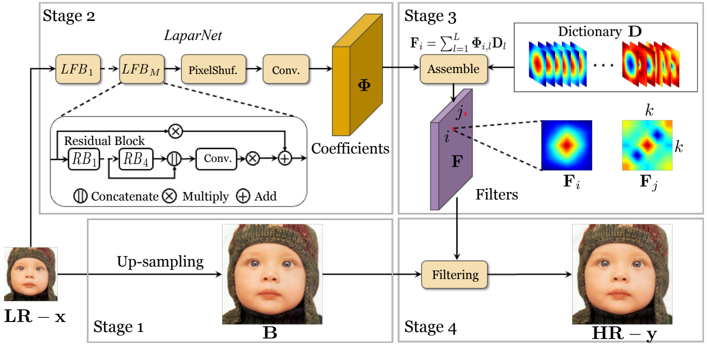
Nov. 1, 2021

Outline

Background

Deep Learning SR Model Architecture

- The architecture of linearly-assembled pixel-adaptive regression network (LAPAR) with four basic stages, i.e., stage 1: *up-sampling*; stage 2: *LaparNet*; stage 3: dictionary assembling; stage 4: filtering.

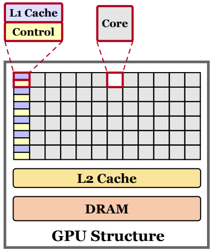


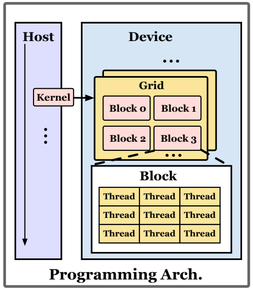
Deep Learning SR Model Architecture

- Stage 1: bilinear up-sampling to upscale input image $\mathbf{x}$
- Stage 2: LaparNet extract features as coefficient matrix Φ from original input $\mathbf{x}$
- Stage 3: dictionary assembling, in which the transformation matrix $\mathbf{F}$ is computed according to ˘ and the pre-defined dictionary $\mathbf{D}$
- Stage 4: filtering, in which the output HR image $\mathbf{y}$ is obtained by applying $\mathbf{F}$ to $\mathbf{B}$, *i.e.*, $\mathbf{y} = \mathbf{F}\mathbf{B}^T$

GPU Programming Architecture

- GPU memory hierarchy and communication mode





- The hardware structure contains a groups of computation cores (streaming processors), multi-level caches, control units and global memory units
- CUDA programming architecture is designed as a wrapper of the hardware
- Each kernel controls a computation grid which can be further divided into multiple blocks.
- Each block is partitioned into a group of threads that can run the same code on different data, synchronously

Overview of our framework

Dictionary Compression: Channel Pruning

$$\begin{aligned} \beta, W = \quad & \arg\min_{\beta,W} \frac{1}{N} \left\| H_{gt} - F_{W,\beta} B^{\top} \right\|_2^2, \\ \text{s.t.} \quad & F_{W,\beta} = \textstyle\sum_{i=0}^{L} \beta_i \Phi D, \\ & \Phi = LaparNet(X, W), \\ & \|\beta\|_0 \le \alpha L. \end{aligned} \tag{1}$$

where N is the size of input batch of images, Φ is the coefficient vector extracted from *LaparNet* with parameters W and H_{gt} is the ground truth high-resolution image. β is the selecting vector on filters of D where $\beta_i = 0$ means the i-th item in the dictionary will be ignored during the compression process.

Dictionary Compression: Channel Pruning

- With two objectives W and β, to solve this optimization problem efficiently, an alternating method including two steps is adopted.
- Step 1: search the suitable selecting vector β corresponding to the required α_t.
- Step 2: tune the parameters $\boldsymbol{W}$ corresponding to the reserved dictionary items with the minimization objective.

Dictionary Compression: Channel Pruning

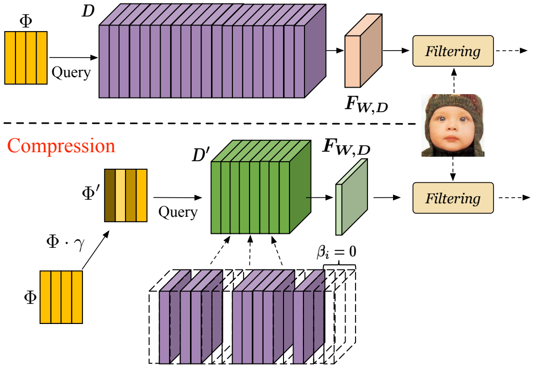
- Step 1 is actually an NP-hard problem. relax the problem to ℓ_1 regulation. solved by utilizing the LASSO regression.

$$\begin{aligned} \beta = \quad & \arg\min_\beta \tfrac{1}{N} \left\| H_{gt} - F_{W,\beta} B^\top \right\|_2^2 + \lambda \|\beta\|_1, \\ & \text{s.t.} \|\beta\|_0 \leq \alpha L. \end{aligned} \tag{2}$$

- Step 2 is to update the parameters

$$W = \arg\min_W \frac{1}{N} \left\| H_{gt} - F_{W,D'} B^\top \right\|_2^2 . \tag{3}$$

Dictionary Compression



Visual illustration of dictionary compression, the upper flow represents original dictionary query and filtering, namely stage 3 + stage 4 in Fig. 2, The flow below demonstrates the compression process of the dictionary query

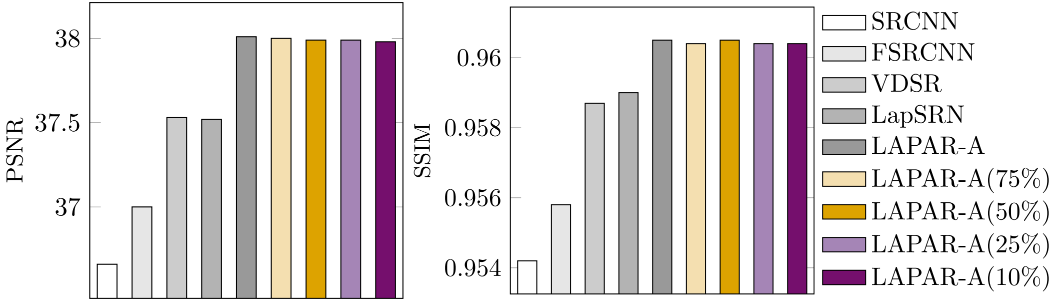
- Dictionary is knowledgeable but may be redundant as well, a distilled compact dictionary is suitable for faster query
- Structured filter pruning of model is hardware friendly, compared with fine-grained unstructured pruning

Dictionary Compression

Algorithm 1 Dictionary Selection Strategy

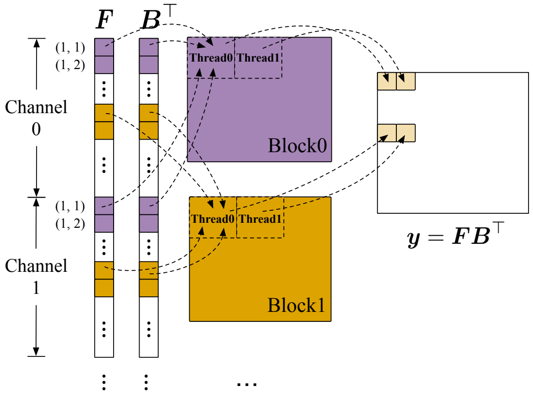
```
1: Input: $\boldsymbol{D} \in \mathbb{R}^{L \times k^2}$, small $\lambda_0$, target $\alpha$, tolerance $\epsilon$;
2: Input: pre-trained $\boldsymbol{W}_0$ , coefficient matrix $\boldsymbol{\Phi}$;
3: $t \leftarrow 0, \alpha_0 \leftarrow 1.0, \beta_0 \leftarrow \mathbf{1} \in \mathbb{R}^L, \gamma_0 \leftarrow \mathbf{1} \in \mathbb{R}^L, \mathscr{L} \leftarrow$ reconstruction error    ▷ Equation (1);
4: repeat
5:     $\alpha_{t+1} \leftarrow \alpha_t - \Delta\alpha$;
6:     $\lambda_{t+1} \leftarrow \lambda_t$;
7:     while $|\beta_{t+1}|_0 > \alpha_{t+1} \cdot L$ do
8:         Fix $\boldsymbol{W}_t$, update $\beta_{t+1} \leftarrow \arg\min_{\boldsymbol{\beta}} \mathscr{L}(\boldsymbol{W}_t, \boldsymbol{\beta D}) + \lambda_{t+1} |\boldsymbol{\beta}|$;    ▷ Equation (2)
9:         $\lambda_{t+1} \leftarrow 2 \cdot \lambda_{t+1}$
10:    end while
11:    $\lambda_{left} \leftarrow 0.5\lambda_{t+1}, \lambda_{right} \leftarrow \lambda_{t+1}$;
12:    while $\left|\alpha_{t+1} \cdot L - \left|\beta_{t+1}\right|_0\right| > \epsilon \cdot L$ do
13:        $\lambda_{t+1} = 1/2(\lambda_{left} + \lambda_{right})$;
14:        Fix $\boldsymbol{W}_t$, update $\beta_{t+1} \leftarrow \arg\min_{\boldsymbol{\beta}} \mathscr{L}(\boldsymbol{W}_t, \boldsymbol{\beta D}) + \lambda_{t+1} |\boldsymbol{\beta}|$;
15:        if $\left|\beta_{t+1}\right|_0 < \alpha_{t+1} \cdot L$ then
16:            $\lambda_{left} \leftarrow \lambda_{t+1}$;
17:        else if $\left|\beta_{t+1}\right|_0 > \alpha_{t+1} \cdot L$ then
18:            $\lambda_{right} \leftarrow \lambda_{t+1}$;
19:        end if
20:    end while
21:    Fix $\beta_{t+1}$, update $\boldsymbol{W}_{t+1} \leftarrow \arg\min_{\boldsymbol{W}} \mathscr{L}(\boldsymbol{W}, \beta_{t+1}\boldsymbol{D})$;
22:    $t = t + 1$;
23: until $\alpha_t \leq \alpha$
```

Dictionary Compression



Single image super-resolution (SISR) performance of our model with different dictionary compression ratios, in comparison with other SR methods. LAPAR-A (Per.%) represents our model with dictionary size shrunk to Per.%. PSNR means peak signal-to-noise ratio. SSIM means structural similarity index measure. PSNR and SSIM are two common metrics to measure the quality of images. The higher the better.

Optimization of Deployments On GPU



An example of the proposed computation engine for image filtering operation

- resources in GPU are limited, which concurrently restricts the computation patterns with respect to the threads, blocks, and etc
- on-chip-memory and l1-cache is limit for each SM.
- Increase thread number will increase parallelism

Optimization of Deployments On GPU

Constrains:

$$
\begin{aligned}
T_r &= (H \times W \times C)/(S \times P \times R), \\
T &\leq \min(T_r, T_{sm}), \\
nx \times ny \times nz &\leq WS \times P \times T,
\end{aligned} \tag{4}
$$

Denote the number of SMs in GPU as S, the number of processing blocks in each SM as P, the size of register file in each processing block as R, the maximum number of threads in each warp as WS. The GPU compute capability constrains the number of warps in each block as smaller than T_{sm}. Denote the three dimensions of the thread block as (nx, ny, nz).

$$
\begin{aligned}
1 &\leq nx \leq H, \\
1 &\leq ny \leq W, \\
1 &\leq nz \leq C.
\end{aligned} \tag{5}
$$

Results

Results

Table: Comparisons on multiple benchmark datasets of our model and other popular SR networks. The dictionary in our model is compressed to 10% of original size for evaluation. Performance metrics are PSNR/SSIM. **Bold**: **best** results

Scale	Method	Set5	Set14	B100	Urban100	Manga109
×2	SRCNNDong, Loy, He, et al. 2014	36.66/0.9542	32.42/0.9063	31.36/0.8879	29.50/0.8946	35.74/0.9661
	FSRCNNDong, Loy, and Tang 2016	37.00/0.9558	32.63/0.9088	31.53/0.8920	29.88/0.9020	36.67/0.9694
	VDSRJ. Kim, J. K. Lee, and K. M. Lee 2016	37.53/0.9587	33.03/0.9124	31.90/0.8960	30.76/0.9140	37.22/0.9729
	DRRNTai, Yang, and Liu 2017	37.74/0.9591	33.23/0.9136	32.05/0.8973	31.23/0.9188	37.92/0.9760
	LapSRNLai et al. 2017	37.52/0.9590	33.08/0.9130	31.80/0.8950	30.41/0.9100	37.27/0.9740
	SRFBN-SLi et al. 2019	37.78/0.9597	33.35/0.9156	32.00/0.8970	31.41/0.9207	38.06/0.9757
	FALSR-AChu et al. 2021	37.82/0.9595	33.55/0.9168	32.12/0.8987	31.93/0.9256	-
	SRMDNFK. Zhang, Zuo, and L. Zhang 2018	37.79/0.9600	33.32/0.9150	32.05/0.8980	31.33/0.9200	-
	Ours	**37.98/0.9604**	**33.59/0.9181**	**32.19/0.8999**	**32.09/0.9281**	**38.60/0.9771**
×3	SRCNNDong, Loy, He, et al. 2014	32.75/0.9090	29.28/0.8209	28.41/0.7863	26.24/0.7989	30.59/0.9107
	FSRCNNDong, Loy, and Tang 2016	33.16/0.9140	29.43/0.8242	28.53/0.7910	26.43/0.8080	30.98/0.9212
	VDSRJ. Kim, J. K. Lee, and K. M. Lee 2016	33.66/0.9213	29.77/0.8314	28.82/0.7976	27.14/0.8279	32.01/0.9310
	DRRNTai, Yang, and Liu 2017	34.03/0.9244	29.96/0.8349	28.95/0.8004	27.53/0.8378	32.74/0.9390
	SelNetChoi and M. Kim 2017	34.27/0.9257	30.30/0.8399	28.97/0.8025	-	-
	CARNAhn, Kang, and Sohn 2018	34.29/0.9255	30.29/0.8407	29.06/0.8034	28.06/0.8493	-
	SRFBN-SLi et al. 2019	34.20/0.9255	30.10/0.8372	28.96/0.8010	27.66/0.8415	33.02/0.9404
	Ours	**34.35/0.9267**	**30.33/0.8420**	**29.11/0.8054**	**28.12/0.8523**	**33.48/0.9439**
×4	SRCNNDong, Loy, He, et al. 2014	30.48/0.8628	27.49/0.7503	26.90/0.7101	24.52/0.7221	27.66/0.8505
	FSRCNNDong, Loy, and Tang 2016	30.71/0.8657	27.59/0.7535	26.98/0.7150	24.62/0.7280	27.90/0.8517
	VDSRJ. Kim, J. K. Lee, and K. M. Lee 2016	31.35/0.8838	28.01/0.7674	27.29/0.7251	25.18/0.7524	28.83/0.8809
	DRRNTai, Yang, and Liu 2017	31.68/0.8888	28.21/0.7720	27.38/0.7284	25.44/0.7638	29.46/0.8960
	LapSRNLai et al. 2017	31.54/0.8850	28.19/0.7720	27.32/0.7280	25.21/0.7560	29.09/0.8845
	CARNAhn, Kang, and Sohn 2018	32.13/0.8937	28.60/0.7806	27.58/0.7349	26.07/0.7837	-
	SRFBN-SLi et al. 2019	31.98/0.8923	28.45/0.7779	27.44/0.7313	25.71/0.7719	29.91/0.9008
	Ours	**32.15/0.8944**	**28.61/0.7817**	**27.59/0.7366**	**26.14/0.7873**	**30.39/0.9072**

Bibliography I

Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn (2018). "Fast, accurate, and lightweight super-resolution with cascading residual network". In: *European Conference on Computer Vision (ECCV)*, pp. 252–268.

Jae-Seok Choi and Munchurl Kim (2017). "A deep convolutional neural network with selection units for super-resolution". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 154–160.

Xiangxiang Chu et al. (2021). "Fast, accurate and lightweight super-resolution with neural architecture search". In: *IEEE International Conference on Pattern Recognition (ICPR)*. IEEE, pp. 59–64.

Chao Dong, Chen Change Loy, Kaiming He, et al. (2014). "Learning a deep convolutional network for image super-resolution". In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 184–199.

Chao Dong, Chen Change Loy, and Xiaoou Tang (2016). "Accelerating the super-resolution convolutional neural network". In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 391–407.

Bibliography II

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee (2016). “Accurate image super-resolution using very deep convolutional networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1646–1654.

Wei-Sheng Lai et al. (2017). “Deep laplacian pyramid networks for fast and accurate super-resolution”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 624–632.

Zhen Li et al. (2019). “Feedback network for image super-resolution”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3867–3876.

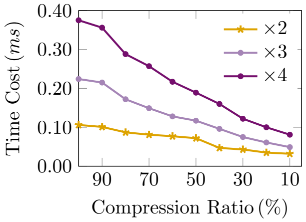
Ying Tai, Jian Yang, and Xiaoming Liu (2017). “Image super-resolution via deep recursive residual network”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3147–3155.

Kai Zhang, Wangmeng Zuo, and Lei Zhang (2018). “Learning a single convolutional super-resolution network for multiple degradations”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3262–3271.

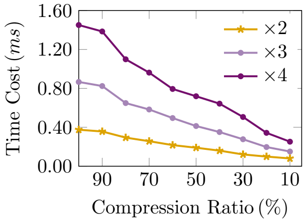
Results

Table: Inference Time (ms) and Acceleration ratios

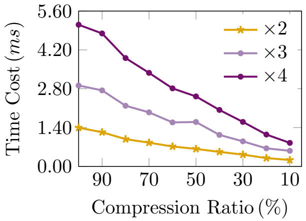
Input size	Scale	NVIDIA GeForce RTX 2080 Ti					NVIDIA Jetson Xavier NX		
		PyTorch	TensorRT	Ours	Acc. (PyTorch)	Acc. (TensorRT)	TensorRT	Ours	Acc. (TensorRT)
64×64	$\times 2$	6.94	1.30	1.02	$\times$680.39%	$\times$127.45%	12.37	9.04	$\times$136.84%
	$\times 3$	8.26	1.94	1.40	$\times$590.00%	$\times$138.57%	22.62	14.28	$\times$158.40%
	$\times 4$	9.86	2.79	1.88	$\times$524.46%	$\times$148.40%	35.83	20.54	$\times$174.44%
128×128	$\times 2$	8.74	3.59	2.66	$\times$328.57%	$\times$134.96%	52.12	37.25	$\times$139.92%
	$\times 3$	13.04	6.19	4.16	$\times$313.46%	$\times$148.80%	90.33	54.26	$\times$166.48%
	$\times 4$	18.07	9.71	6.13	$\times$294.78%	$\times$158.40%	144.34	81.29	$\times$177.56%
180×320	$\times 2$	17.12	12.40	9.25	$\times$185.08%	$\times$134.05%	177.57	124.12	$\times$143.06%
	$\times 3$	30.83	21.66	14.63	$\times$210.73%	$\times$148.05%	325.07	200.02	$\times$162.52%
	$\times 4$	44.69	34.69	22.12	$\times$202.03%	$\times$156.82%	534.99	318.60	$\times$167.92%
360×640	$\times 2$	67.36	50.26	37.47	$\times$179.77%	$\times$134.13%	748.72	530.23	$\times$141.21%
	$\times 3$	105.32	88.45	59.20	$\times$177.90%	$\times$149.41%	1466.91	973.25	$\times$150.72%
	$\times 4$	406.93	141.08	91.09	$\times$540.02%	$\times$154.88%	-	-	-
Average	-	61.43	31.17	**20.91**	$\times$**352.27%**	$\times$**144.49%**	328.26	**214.81**	$\times$**156.28%**



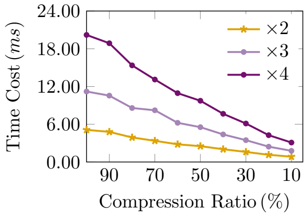
(a) Input Size 64×64



(b) Input Size 128×128



(c) Input Size 180×320



(d) Input Size 360×640

Time consumptions of the dictionary query and filtering with different compression ratios. Different input image sizes and scaling factors (from 2 to 4) are evaluated.

THANK YOU!