arXiv:2208.14655v1 [eess.IV] 31 Aug 2022

XCAT - Lightweight Quantized Single Image Super-Resolution Using Heterogeneous Group Convolutions and Cross Concatenation

Mustafa Ayazoglu and Bahri Batuhan Bilecen

Aselsan Research, Ankara, Turkey
{mayazoglu,batuhanb}@aselsan.com.tr

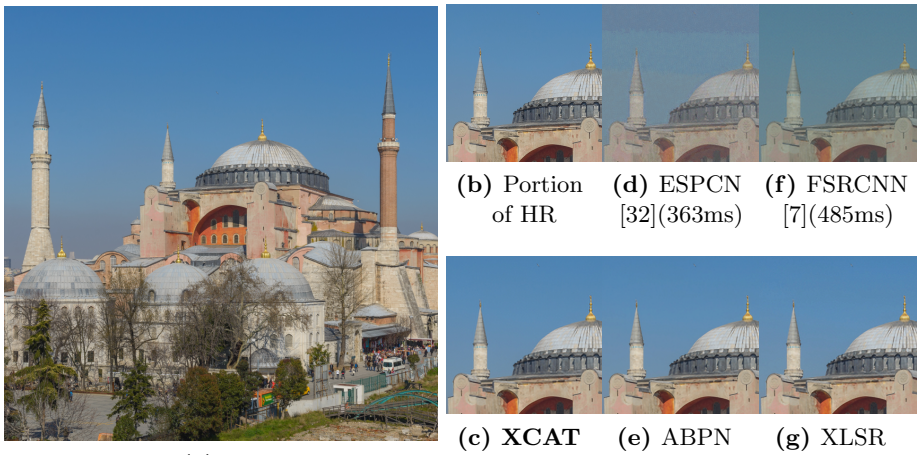
Abstract. We propose a lightweight, single-image super-resolution mobile device network named XCAT, and introduce Heterogeneous Group Convolution Blocks with Cross Concatenations (HXBlock). The heterogeneous split of the input channels to the group convolution blocks reduces the number of operations, and cross concatenation allows for information flow between the intermediate input tensors of cascaded HXBlocks. Cross concatenations inside HXBlocks can also avoid using more expensive operations like 1x1 convolutions. To further prevent expensive tensor copy operations, XCAT utilizes non-trainable convolution kernels to apply upsampling operations. Designed with integer quantization in mind, XCAT also utilizes several techniques in training, like intensity-based data augmentation. Integer quantized XCAT operates in real-time on Mali-G71 MP2 GPU with 320ms, and on Synaptics Dolphin NPU with 30ms (NCHW) and 8.8ms (NHWC), suitable for real-time applications.

Keywords: Single image super-resolution, quantization, group convolutions, mobile AI

1 Introduction

Super-resolution (SR) is an extensively studied computer vision problem that aims to generate higher resolution (HR) image(s) given lower resolution (LR) image(s). In single image super-resolution (SISR), a single image; and in multi-image super-resolution (MISR), multiple images are utilized to generate a single HR image. In either case, image super-resolution is an ill-posed problem, since there is no unique solution. This ill-posed problem has been attempted to be solved via classical methods [9] and deep-learning based methods [32][6]; however, many new methods based on deep-learning are still being developed, most of which purely focus on data fidelity.

However, for the SR method to be practically applicable, the runtime is as important as the method's PSNR performance. Due to its practical importance, recent literature studies on SR focus on deployability, runtime, quantization, and efficiency, as well as PSNR of the method [2][8][3][13][26]. Yet, achieving



(a) HR **(b)** Portion of HR **(c) XCAT (320ms)** **(d)** ESPCN [32](363ms) **(e)** ABPN [8](600ms) **(f)** FSRCNN [7](485ms) **(g)** XLSR [2](370ms)

Fig. 1: Comparative results of UINT8 quantized models on DIV2K (Val image: 890). (d) and (f) yield both visually and numerically worse results than the rest. Visual results of (c), (e), and (g) are indistinguishable to the naked eye; however, XCAT runs the fastest

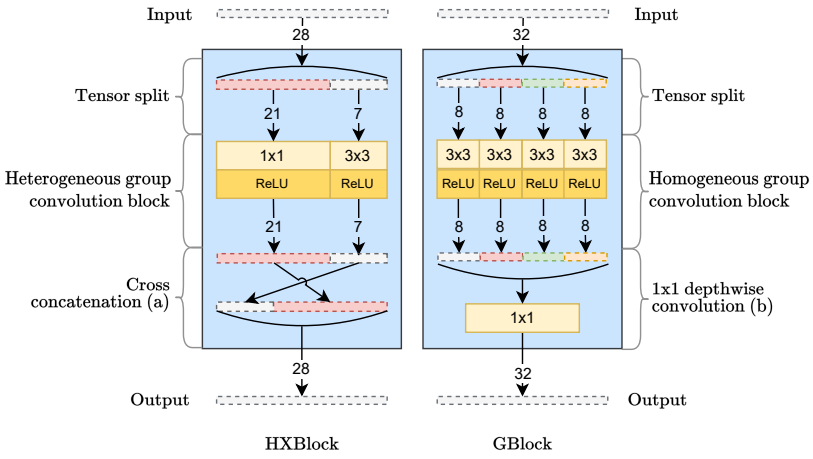


Fig. 2: Proposed HXBlock (left) vs. GBlock [2] (right). Dashed rectangles represent tensors. Group convolutions inside HXBlock are **heterogeneous** compared to GBlock's. HXBlock uses **cross concatenation (a)** instead of depth wise 1x1 convolutions (b), which provides information flow through different convolutional kernels when HXBlocks are cascaded. For HXBlock, using (a) results in a significant runtime performance increase in return of a small PSNR drop compared to using (b)

real-time performance with satisfactory visual quality during the quantization process further complicates the problem and careful network design is needed.

In this study, we focus on the efficiency and mobile deployment, in the scope of *Mobile AI & AIM 2022 Real-Time Image Super-Resolution Challenge* [14]. Our model, named **XCAT**, is a SISR network incorporating the proposed HXBlock and modifying both new and existing techniques from the literature for providing a quantization-aware, robust, real-time performance model, suitable for mobile devices.

Our work makes the following contributions:

1. **HXBlocks**, which are heterogeneous grouped convolutions with cross concatenation layers for allowing information flow with almost no computational cost through different convolutional kernels of the group. Relevant studies done on HXBlocks have shown that they can be replaced with traditional group convolutions with little sacrifice from PSNR, but a significant gain in run time performance.
2. A method for nearest neighborhood up sampling method with fixed 2D convolutional kernels to replace expensive tensor copy operations on mobile devices, which makes the model robust to quantization.
3. An efficient, mobile device friendly, single image super-resolution network named **XCAT**.

2 Related Works

DNN-based single image super-resolution. First deep-learning-based SISR algorithm was proposed by Dong et. al. as SRCNN [6]. Later, as a speed improvement on SRCNN, FSRCNN [7] was developed; which introduced a deconvolutional layer at the end of the network, replaced ReLU with a PReLU activation layer, and reformulated SRCNN by adopting smaller filter sizes but more mapping layers. Shi et. al. [32] introduced a novel, efficient sub-pixel convolutional layer (also known as depth to space), which is actually widely used in many fast SR networks right now. VDSR [19], EDSR [27], and WDSR [39] continued the development of deep-learning-based SR by increasing the number of parameters, in exchange for accuracy with speed.

With the recent developments in computer vision and deep learning, concepts like attention mechanism [30], generative adversarial networks [25][35], recursive & residual networks [20][33][1], and distillation layers [12][11][3][29] also started to take part inside SR network architectures. GANs and networks with attention mechanism mostly generate a high-quality SR image by sacrificing speed, whereas RNNs and distilling networks try to decrease the computational load.

Group convolutions. Group convolutions consist of groups of multiple convolutional kernels placed within the same layer. The motivation behind group convolutions emerged with AlexNet [22], desiring to distribute the model over multiple GPUs to overcome hardware limitations. Later on, besides the increase in speed in AlexNet, group convolutions are also observed to improve classification accuracy when groups are accompanied by skipped connections with

ResNetX [36]. ShuffleNet [41] introduced shuffling the intermediate tensors between group convolution blocks to increase feature extraction. DeepRoots [16] and more recent studies use different convolutional kernels inside groups, such as 1x1 depth wise convolutions [31] and dilated convolutions [38][42]. In addition, unitary [43] and interleaved [40] group convolutions also offer different perspectives on how to extract various features from input images. Usage of group convolutions due to their efficiency on super-resolution problems is also present [2][3][18].

Model Optimization. Hardware limitations and specifications may require the model to be optimized via different techniques, such as quantization, pruning, clustering, network architecture search (NAS), and many more. **Quantization** refers to converting floating point values to integers, hence decreasing memory usage and computational cost when re-accessing and/or updating the mentioned values, at a cost of decreasing the precision. Quantization is particularly useful on neural network models since it can decrease inference times without sacrificing much inference accuracy if done correctly [17]. Models also can be quantized after quantization-aware training in floating point precision [34][21], as well as training the network directly with low precision multiplications [5]. Removing layers from a model having a minor effect on inference is called **pruning** [37][10], and **clustering** is the method of decreasing the number of unique weights by grouping weights and assigning the centroid values for each group. All of these methods try to decrease processor utilization or memory usage, or both. Besides optimizing an existing network structure, finding the most possible optimal network structure in search space is also a study area, known as network architecture search [44].

3 Method

In this section, XCAT is defined by its overall architecture and its components. Details about the training techniques and the quantization procedure are explained thoroughly as well.

3.1 XCAT's Architecture

As seen from Fig. 3, XCAT consists of 3 individual convolutional layers with trainable 3x3 kernels, a single 1x1 convolutional layer with a fixed "identity kernel" to simulate nearest neighborhood upsampling operation, m HXBlocks, a tensor addition layer, followed by a depth to space (D2S) layer and a clipped ReLU activation layer. Input and output of XCAT are LR and SR, respectively, where SR has x3 resolution of LR.

Each key component of XCAT will be detailed with their reasoning:

Group convolutions with heterogeneous filter groups and varying kernels. Group convolutions, which include multiple convolutional kernels per layer, are known to be able to extract and learn more varying features compared to a single kernel [23]. XCAT inherits this idea of group convolution blocks to

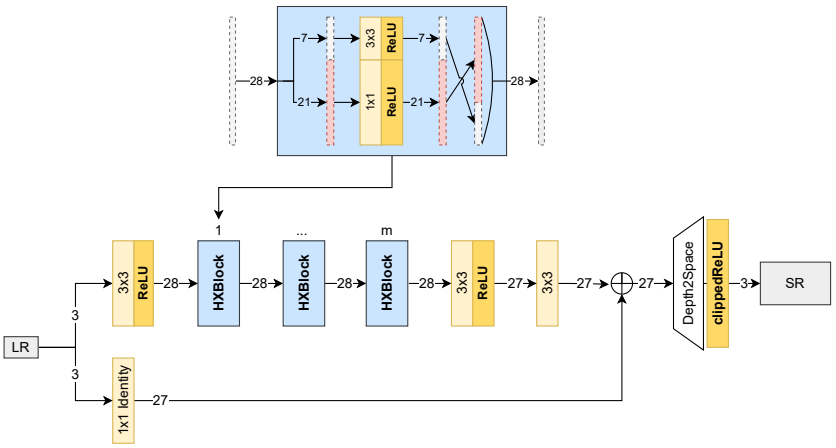


Fig. 3: Network structure of XCAT. Numbers on arrows denote channel numbers, and numbers inside blocks represent kernel sizes of convolutional blocks. Dashed blocks represent tensors, whereas non-dashed ones represent operators like convolution and normalization. Convolution with 1x1 identity kernel performs the upsampling method visualized in Fig. 4. XCAT has m HXBlocks, where m=2 for this study

replace single-layered convolutions in a repeated manner. However, as opposed to initial approaches [36], convolutional layers inside the group convolution blocks in XCAT have **different layer dimensions** and **different kernel sizes** (Fig. 3). This allows to pass the same source information between different convolutional layers and allows for less computationally demanding feature extraction. The input tensor is split into two parts in channel dimension, one processed by 1x1 and the other by 3x3 convolutional kernels. 1x1 convolution "blends" the pointwise information from previous HXBlocks and extracts inter-channel features, whereas 3x3 convolution considers in-channel correlation as well. In addition, a relevant study of Lee et al.'s [24] logarithmic filter groups in shallow CNNs shows the positive effect of dividing group convolution input tensors unevenly.

Cross concatenation. First group convolutions in AlexNet [22] ended with max pooling layers. However, group convolution designs such as DeepRoots [16] started utilizing low-dimensional embeddings (like 1x1 convolutions) at the end of the groups, with the inspiration taken from Lin. et. al. [28] and Cogswell et. al. [4]. This was done to decrease the computational cost and number of parameters without compromising accuracy. Later on, several efficiency-oriented SR networks like XLSR [2] and IMDeception [3] also utilized group convolutions ending with 1x1 depth-wise convolutions.

In XCAT, instead of using 1x1 depth wise convolutions for increasing the spatial receptive field of each output of a group convolution block, the output tensor of each group convolution block is **cross concatenated**. The inspiration

came from ShuffleNet [41] and Swin Transformer [29]; where channel shuffling and convolutional layers are inserted between group convolutions in the former, and window partitions are cyclic shifted to enable information flow between windows in the latter. Each cross concatenation in XCAT corresponds to a circular shift of one-fourth of the input tensor (Fig. 3). This cyclic procedure allows the information to pass through from 1x1 and 3x3 convolutions inside XCAT's group convolution blocks, hence having more chance for feature extraction.

It is worthy to note that ShuffleNet [41]'s channel shuffling is similar to the XCAT's; however, XCAT has a cross concatenation operation represented with cyclic shifts, whereas ShuffleNet has a shuffle operation dividing and reorganizing tensors into many small partitions. This reorganization operation is reflected onto the target device (Synaptics Dolphin NPU) as reshape and transpose operations, which take much longer to process compared to XCAT's simpler yet effective approach.

During the experiments, it is observed that replacing cross concatenation operations with 1x1 convolutions in XCAT increases the run time per frame, but does not increase the PSNR test score considerably, making it less practical for mobile networks.

Depth to space (D2S) operation. Shi et al. [32]'s pixel shuffling (depth to space operator) is inserted at the end of the network, which aims to implement sub-pixel convolutions in an efficient manner and is proven to increase PSNR score in super-resolution problems in many studies.

Nearest neighborhood upsampling with fixed kernel convolutions. We observed that providing the low-resolution input image to D2S with accompanying feature tensors increases the robustness, as opposed to only providing the extracted feature tensors to D2S. With this motivation, XCAT also adds repeated input image tensors (where each channel of the input image is repeated 9 times) to feature tensors and provides them to D2S. From the perspective of D2S, this operation is equivalent to the nearest neighborhood upsampling.

A relevant study done by Du et. al. named ABPN [8] also utilizes the nearest neighborhood upsampling to be fed to the D2S block. However, it uses tensor copy operations while repeating and concatenating the input image in the upsampling process, which are indeed expensive for mobile devices. For a better alternative, a convolutional layer of 3 input channels and 27 output channels is used, with a 1x1 non-trainable kernel which is set to serve the same purpose as a tensor copy (Fig. 4). One point to note is that when this 1x1 kernel is set as trainable, it gets affected by the quantization process and yields lower visual quality results.

3.2 Training and Quantization Details

XCAT is trained in floating point precision and quantized afterward. However, it is trained and designed with quantization in mind, with several techniques to avoid PSNR decrease:

Intensity-augmented training. To minimize the PSNR difference while quantizing the FP32 model to its UINT8, intensity values of the training images

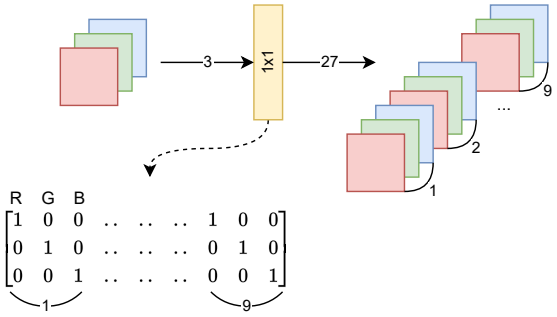


Fig. 4: Tensor copy operations done with convolutions. The identity kernel of the 1x1 convolution is set in such a way that it reproduces the input tensor of 3 channels 9 times, generating an output tensor of 27 channels

are scaled with randomly chosen constants among (1, 0.7, 0.5). We have observed that this strategy helped with quantization and avoided signal degradation, as stated in [2].

Clipped ReLU. As proven and explained in [2], using clipped ReLU at the end of the network allows better quantization while keeping the performance in the real-time range.

Representative dataset selection. TensorFlow Lite requires a representative dataset while quantizing a floating point Keras or TensorFlow model. As a rule of thumb, this dataset consists of entire training images. However, it is observed that selecting a subset of all training images as the representative dataset affects the final PSNR test score of the quantized UINT8 model immensely. Hence, to find the most suitable representative dataset, a linear search is applied to all DIV2K training images, generating single image representative datasets. For each representative dataset (or rather an image), XCAT is quantized and PSNR test scores are measured. The highest scoring quantized XCAT model is chosen as the best.

Training details. XCAT is trained twice in floating point precision and then quantized. Training details are as follows:

- DIV2K dataset is used for the first training, and Flickr2K dataset is added alongside for the second training (fine tuning).
- Intensity, rotation, random crop, and flip augmentations are used while setting up the dataset for both of the training. HR images are cropped to 96x96 patches.
- XCAT is trained with 50 epochs and 16 batches. Each epoch contains 10000 mini-batches.
- For the first training:
  - Charbonnier loss is used, where $C(x) = \sqrt{(x^2 + \epsilon^2)}$ and $\epsilon = 0.1$. Charbonnier loss is the smoother version of L1-loss having better convergence characteristics than L2-loss.

  - Adam optimizer is used with initial learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e^{-8}$.
  - Warm-up scheduler is used: Starting from the initial learning rate, at each epoch, the learning rate is increased up to $25e^{-4}$ until the 5th epoch. After the 5th epoch, the learning rate is linearly decreased at the end of each epoch, where at the last epoch it decreases to e^{-4}.
- For the second training for fine-tuning:
  - Mean square error is used as the loss.
  - Adam optimizer is used with an initial learning rate = 0.0001, and the same beta and epsilon parameters.
  - Warm-up scheduler is used again, but with the new initial learning rate, and the maximum learning rate of $12.5e^{-4}$ instead of $25e^{-4}$.

4 Experimental Results

During the development of XCAT, many modified versions were created and tested. Numerical results of XCAT models and the ablation study done for HXBlocks are given in Tab. 1. Comparative visual results are given in Fig. 5.

To choose the most successful model, we used the score function in (1), which is officially published in the competition's evaluation criteria.

$$Score = \frac{2^{2(PSNR_{(UINT8)}-30)}}{t_{(UINT8)}10^{-5}} \tag{1}$$

Comparative study. Tab. 4 and Fig. 5 reveal that the network architecture's suitability for the quantization procedure plays a big role in producing high-quality, super-resolved images. Despite XLSR and ABPN having higher PSNR FP32 scores compared to XCAT, after the quantization, all three yielded similar visual results and closer UINT8 PSNR scores to each other.

Ablation study. In Tab. 1, increasing layer number/sizes and parameter numbers increased the PSNR score and decreased run time performance (E-G, G-L, B-C). Decreasing number of groups had a negative effect on PSNR; however, the positive effect on runtime surpassed (A-E). Using (I) dynamic kernels as opposed to not using (J) had a significant runtime boost with PSNR scores almost unchanged. Different heterogeneous divisions of filters (G-H) are also tried. Logically, when the input size of the 3x3 convolution layer increased, the PSNR score also increased. However, the penalty of runtime overcame the positive benefits of the PSNR raise. Replacing cross concatenation layers with 1x1 convolutions (XCAT-B) had the same effect as in the previous case.

Tab. 3 shows the effect of using cross concatenation instead of directly concatenating the intermediate tensors in HXBlocks, as well as using different tensor divisions and number of HXBlocks. It is proven that using cross-concatenation allows for better information flow and increases the PSNR score, as opposed to using direct concatenation. This effect is more visible when the number of HXBlocks increase.

Table 1: Different XCAT-based models and their performance. Runtime is evaluated on Mali-G71 MP2 GPU via AI Benchmark 5 [15]. Score is the metric function described in (1). Note that the Config column describes the differences among models. m is the number of HXBlocks. X/Y shows the splitting ratio, where X+Y is the total number of channels of HXBlocks. axa/bxb represents the convolutional kernels inside the group convolution blocks, where the tensor with dimension X passes through axa, and Y through bxb

Model	**Config**	**PSNR FP32/UINT8**	**Runtime (ms)**	**Score**
XCAT	m=2, 21/7, 1x1/3x3 2 stage training	29.88/29.81	320	**240**
A	m=2, 21/7, 1x1/3x3 1 stage training	29.85/29.79	320	233
B	m=2, 21/7, 1x1/3x3 Cross Cat → 1x1 Conv.	29.92/29.84	340	235
C	m=2, 21/7, 3x3/3x3 Cross Cat → 1x1 Conv.	**30.04/29.96**	780	121
D	m=2 HXBlock → 3x3 Conv.	30.04/29.97	770	124
E	m=4, 21/7, 1x1/3x3	29.98/29.89	370	232
F	m=4, 21/7, 1x1/3x3 Conv after last HXBlock: 3x3 → 1x1	29.82/29.75	300	236
G	m=4, 21/7, 1x1/3x3 Conv after last HXBlock: removed	29.81/29.72	290	234
H	m=4, 16/12, 1x1/3x3 Conv after last HXBlock: removed	29.87/29.76	300	238
I	m=4, 7/21, 1x1/3x3 Conv after last HXBlock: removed	30.03/29.88	520	163
J	m=4, 7/21, 3x3/3x3 Conv after last HXBlock: removed	30.04/29.87	550	152
K	m=3, 7/21, 1x1/3x3 Conv after last HXBlock: removed	29.95/29.81	430	179
L	m=4, 16/4, 1x1/3x3 Conv after last HXBlock: removed	29.61/29.45	**205**	228
M	m=4, 16/4, 1x1/3x3 Replaced Add with Concat	29.63/29.15	298	103

Table 2: PSNR test scores of XCAT and other algorithms with public datasets. All models are FP32. To be consistent with the rest of the algorithms; XLSR, ABPN, and XCAT's PSNR results are calculated using Luminance (Y) channel rather than RGB channels, except for DIV2K (*We performed our own training since the pre-trained FP32 model from the authors performed poorly, around 15dB for DIV2K(Val))

Dataset	**Scale**	**Bicubic**	**FSRCNN**	**ESPCN**	**XLSR**	**ABPN***	**XCAT** (proposed)
Set5	x3	30.44	32.73	32.59	33.09	33.45	33.02
Set14	x3	27.63	29.30	29.18	29.59	29.73	29.54
B100	x3	27.13	28.26	28.18	28.45	28.56	28.42
Urban100	x3	24.43	26.03	25.87	26.48	26.73	26.38
Manga109	x3	26.87	30.21	29.70	31.13	31.47	31.12
DIV2K(Val)	x3	28.82	29.67	29.54	30.10	30.10	29.88

Table 3: Effect of cross concatenation versus straight concatenation (no shuffling of the tensors while concatenating), the number of HXBlocks, and the tensor divisions. All models are based on XCAT. All parameter changes are mentioned on the table, and the rest are kept the same among all models. Runtime is evaluated on Mali-G71 MP2 GPU via AI Benchmark 5 [15]. Score is the metric function described in (1). Split and kernel definitions are stated in Tab. 1

Split	**Kernel**	**m (# of HXBlocks)**	**Cross Concat**	**PSNR** (FP32)	**PSNR** (UINT8)	**Runtime** (ms)	**Score**
21/7	1x1/3x3	2	✓	29.88	29.81	320	240
21/7	1x1/3x3	2	×	29.87	29.79	320	233
21/7	1x1/3x3	4	✓	29.98	29.89	370	232
21/7	1x1/3x3	4	×	29.96	29.87	370	232
21/7	1x1/3x3	8	✓	30.04	29.97	480	200
21/7	1x1/3x3	8	×	30.01	29.89	480	179
21/7	1x1/3x3	12	✓	30.07	29.97	580	165
21/7	1x1/3x3	12	×	30.04	29.72	580	117
24/8	1x1/3x3	6	✓	30.02	29.92	410	218
24/8	1x1/3x3	6	×	30.01	29.89	410	209
56/8	1x1/3x3	4	✓	30.08	30.03	620	168
56/8	1x1/3x3	4	×	30.04	29.99	620	159

Fig. 5: Comparative results of UINT8 quantized models on DIV2KVal dataset. The proposed method is applied for (c)'s representative dataset, whereas all DIV2KVal images are used for (d) and (f)'s. (e) and (g) are the pre-trained quantized models provided by the authors. Visual results of (c), (e), and (g) are indistinguishable; however, XCAT runs faster

Table 4: PSNR (dB) drops for DIV2K(Val and Test, x3) before and after quantization, number of parameters, and runtime scores (ms) on Mali-G71 MP2 via AI Benchmark 5 [15] and Synaptics NPU (*FP32 and UINT8 scores are taken from the paper. In addition, the authors' pre-trained .tflite model gave a concatenation error on AI Benchmark 4 & 5, about the source tensor not being able to be used multiple times. Hence, the model code is altered for ABPN, where the relevant tensor is manually hard-copied and concatenated) ($^+$Tested in NCHW format)

Metric	**FSRCNN** [7]	**ESPCN** [32]	**XLSR** [2]	**ABPN** [8]*	**XCAT** (proposed)
Val, FP32 PSNR	29.67	29.54	30.10	30.22	**29.88**
Val, UINT8 PSNR	18.52	17.50	29.82	30.09	**29.81**
ΔPSNR	11.15	12.04	0.28	0.13	**0.07**
Test, UINT8 PSNR	-	-	29.58	29.87	**29.67**
# of parameters	25K	31K	22K	42K	**16K**
Synaptics Runtime$^+$	-	-	44.8	36.9	**8.8**
Mali Runtime	485	363	370	600	**320**
Score	0.003	0.061	210	188	**240**

5 Conclusions & Future Studies

This study proposes a lightweight, quantized single image super-resolution network named XCAT, submitted to *Mobile AI & AIM 2022 Real-Time Image Super-Resolution Challenge*. XCAT offers **heterogeneous group convolution blocks** which includes convolutional kernels with different kernels and input & output tensor sizes. Compared to other studies which include group convolutions ending with 1x1 layers, **cross concatenating** the intermediate tensors between group convolutions offer runtime efficiency with tolerable sacrifice from PSNR test scores. To further increase runtime performance on mobile devices, upsampling done by tensor copy operations by default is replaced by a 1x1 convolutional layer with a non-trainable kernel. XCAT is also shown to be robust to quantization, with a decrease of 0.07dB from FP32 to the UINT8 model.

Comparative experimental results on slightly modified XCAT models reveal that the design choices proposed in this study offer the model to be deployed on mobile devices efficiently. To further prove the effectiveness of the proposed method, XCAT is evaluated with standardized datasets in comparison to other mobile-friendly super-resolution networks. Visual results indicate that XCAT can produce super-resolved images nearly identical to the other slower networks' outputs. Although HXBlock is designed for super-resolution problems, we believe that it can help many heavy models to facilitate running on mobile devices.

References

1. Ahn, N., Kang, B., Sohn, K.: Fast, accurate, and, lightweight super-resolution with cascading residual network. CoRR **abs/1803.08664** (2018)
2. Ayazoglu, M.: Extremely lightweight quantization robust real-time single-image super resolution for mobile devices. CoRR **abs/2105.10288** (2021)
3. Ayazoglu, M.: Imdeception: Grouped information distilling super-resolution network (2022)
4. Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., Batra, D.: Reducing overfitting in deep networks by decorrelating representations (11 2015)
5. Courbariaux, M., Bengio, Y., David, J.P.: Training deep neural networks with low precision multiplications (12 2015)
6. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 184–199. Springer International Publishing, Cham (2014)
7. Dong, C., Loy, C.C., Tang, X.: Accelerating the super-resolution convolutional neural network. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 391–407. Springer International Publishing, Cham (2016)
8. Du, Z., Liu, J., Tang, J., Wu, G.: Anchor-based plain net for mobile image super-resolution (2021)
9. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 349–356 (2009)
10. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 1398–1406 (2017)
11. Hui, Z., Gao, X., Yang, Y., Wang, X.: Lightweight image super-resolution with information multi-distillation network. pp. 2024–2032 (10 2019)
12. Hui, Z., Wang, X., Gao, X.: Fast and accurate single image super-resolution via information distillation network. pp. 723–731 (06 2018)
13. Ignatov, A., Timofte, R., Denna, M., Younes, A., Lek, A., Ayazoglu, M., Liu, J., Du, Z., Guo, J., Zhou, X., Jia, H., Yan, Y., Zhang, Z., Chen, Y., Peng, Y., Lin, Y., Zhang, X., Zeng, H., Zeng, K., Wang, S.: Real-time quantized image super-resolution on mobile npus, mobile ai 2021 challenge: Report. pp. 2525–2534 (06 2021)
14. Ignatov, A., Timofte, R., Denna, M., Younes, A., et al.: Efficient and accurate quantized image super-resolution on mobile npus, mobile ai & aim 2022 challenge: Report. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops (2022)
15. Ignatov, A., Timofte, R., Kulik, A., Yang, S., Wang, K., Baum, F., Wu, M., Xu, L., Van Gool, L.: Ai benchmark: All about deep learning on smartphones in 2019 (10 2019)
16. Ioannou, Y., Robertson, D., Cipolla, R., Criminisi, A.: Deep roots: Improving cnn efficiency with hierarchical filter groups (07 2017)
17. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. pp. 2704–2713 (06 2018)
18. Jain, V., Bansal, P., Kumar Singh, A., Srivastava, R.: Efficient single image super resolution using enhanced learned group convolutions (08 2018)

19. Kim, J., Lee, J.K., Lee, K.M.: Accurate image super-resolution using very deep convolutional networks. CoRR **abs/1511.04587** (2015)
20. Kim, J., Lee, J.K., Lee, K.M.: Deeply-recursive convolutional network for image super-resolution. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1637–1645 (2016)
21. Krishnamoorthi, R.: Quantizing deep convolutional networks for efficient inference: A whitepaper (06 2018)
22. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012)
23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012)
25. Ledig, C., Theis, L., Huszar, F., Caballero, J., Aitken, A.P., Tejani, A., Totz, J., Wang, Z., Shi, W.: Photo-realistic single image super-resolution using a generative adversarial network. CoRR **abs/1609.04802** (2016)
26. Li, Y., Zhang, K., Timofte, R., Van Gool, L., Kong, e.a.: Ntire 2022 challenge on efficient super-resolution: Methods and results (2022)
27. Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced deep residual networks for single image super-resolution. CoRR **abs/1707.02921** (2017)
28. Lin, M., Chen, Q., Yan, S.: Network in network (2013)
29. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows (2021)
30. Niu, B., Wen, W., Ren, W., Zhang, X., Yang, L., Wang, S., Zhang, K., Cao, X., Shen, H.: Single image super-resolution via a holistic attention network (2020)
31. Schwarz Schuler, J.P., Romaní, S., Abdel-nasser, M., Rashwan, H., Puig, D.: Grouped pointwise convolutions reduce parameters in convolutional neural networks. Mendel **28**, 23–31 (06 2022)
32. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1874–1883 (2016)
33. Tai, Y., Yang, J., Liu, X.: Image super-resolution via deep recursive residual network. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2790–2798 (2017)
34. Vanhoucke, V., Senior, A., Mao, M.: Improving the speed of neural networks on cpus (01 2011)
35. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C.C., Qiao, Y., Tang, X.: Esrgan: Enhanced super-resolution generative adversarial networks (2018)
36. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5987–5995 (2017)
37. Xu, S., Huang, A., Chen, L., Zhang, B.: Convolutional neural network pruning: A survey. In: 2020 39th Chinese Control Conference (CCC). pp. 7458–7463 (2020)

38. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions (05 2016)
39. Yu, J., Fan, Y., Yang, J., Xu, N., Wang, Z., Wang, X., Huang, T.S.: Wide activation for efficient and accurate image super-resolution. CoRR **abs/1808.08718** (2018)
40. Zhang, T., Qi, G.J., Xiao, B., Wang, J.: Interleaved group convolutions. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 4383–4392 (2017)
41. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6848–6856 (2018)
42. Zhang, Z., Wang, X., Jung, C.: Dcsr: Dilated convolutions for single image super-resolution. IEEE Transactions on Image Processing **28**(4), 1625–1635 (2019)
43. Zhao, R., Hu, Y., Dotzel, J., De Sa, C., Zhang, Z.: Building efficient deep neural networks with unitary group convolutions. pp. 11295–11304 (06 2019)
44. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8697–8710 (2018)