Learned Smartphone ISP on Mobile GPUs with Deep Learning, Mobile AI & AIM 2022 Challenge: Report

Andrey Ignatov, Radu Timofte, Shuai Liu, Chaoyu Feng, Furui Bai, Xiaotao Wang, Lei Lei, Ziyao Yi, Yan Xiang, Zibin Liu, Shaoqing Li, Keming Shi, Dehui Kong, Ke Xu, Minsu Kwon, Yaqi Wu, Jiesi Zheng, Zhihao Fan, Xun Wu, Feng Zhang, Albert No, Minhyeok Cho, Zewen Chen, Xiaze Zhang, Ran Li, Juan Wang, Zhiming Wang, Marcos V. Conde, Ui-Jin Choi, Georgy Perevozchikov, Egor Ershov, Zheng Hui, Mengchuan Dong, Xin Lou, Wei Zhou, Cong Pang, Haina Qin, and Mingxuan Cai *

Abstract. The role of mobile cameras increased dramatically over the past few years, leading to more and more research in automatic image quality enhancement and RAW photo processing. In this Mobile AI challenge, the target was to develop an efficient end-to-end AI-based image signal processing (ISP) pipeline replacing the standard mobile ISPs that can run on modern smartphone GPUs using TensorFlow Lite. The participants were provided with a large-scale Fujifilm UltraISP dataset consisting of thousands of paired photos captured with a normal mobile camera sensor and a professional 102MP medium-format FujiFilm GFX100 camera. The runtime of the resulting models was evaluated on the Snapdragon's 8 Gen 1 GPU that provides excellent acceleration results for the majority of common deep learning ops. The proposed solutions are compatible with all recent mobile GPUs, being able to process Full HD photos in less than 20-50 milliseconds while achieving high fidelity results. A detailed description of all models developed in this challenge is provided in this paper.

Keywords: Mobile AI Challenge, Learned ISP, Mobile Cameras, Photo Enhancement, Mobile AI, Deep Learning, AI Benchmark

1 Introduction

Nowadays, the cameras are ubiquitous mainly due to the tremendous success and adoption of modern smartphones. Over the years, the quality of the smart-

*Andrey Ignatov *(andrey@vision.ee.ethz.ch)* and Radu Timofte *(radu.timofte@uni-wuerzburg.de)* are the main Mobile AI & AIM 2022 challenge organizers . The other authors participated in the challenge.

Appendix A contains the authors' team names and affiliations.

Mobile AI 2022 Workshop website:
https://ai-benchmark.com/workshops/mai/2022/

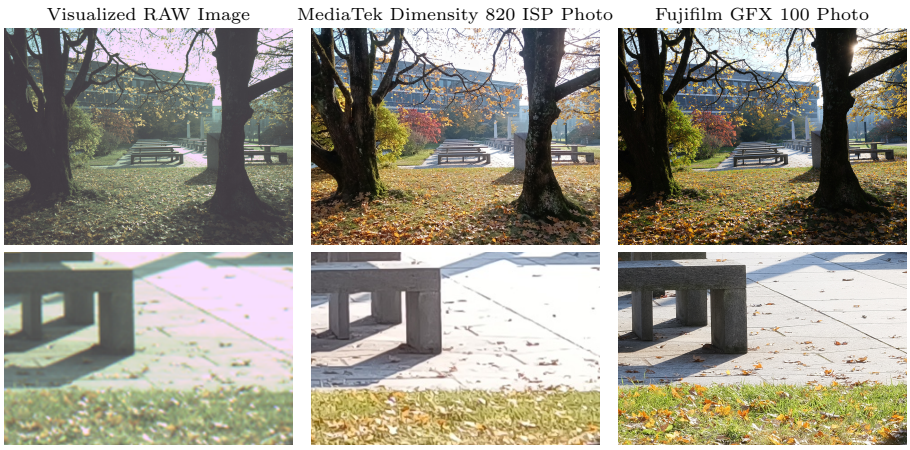

Fig. 1. Example set of full-resolution images (top) and crops (bottom) from the collected Fujifilm UltraISP dataset. From left to right: original RAW visualized image, RGB image obtained with MediaTek's built-in ISP system, and Fujifilm GFX100 target photo.

phone cameras continuously improved due to advances in both hardware and software. Currently, due to their versatility, the critical improvements are coming from the advanced image processing algorithms employed, *e.g.*, to perform color reconstruction or adjustment, noise removal, super-resolution, high dynamic range processing. The image enhancement task can be effectively solved with deep learning-based approaches. The critical part is the acquisition of appropriate (paired) low and high-quality ground truth images for training. For the first time the end-to-end mobile photo quality enhancement problem was tackled in [19, 20]. The authors proposed to directly map the images from a low-quality smartphone camera to the higher-quality images from a high-end DSLR camera. The introduced DPED dataset was later employed in many competitions [36, 28] and works [68, 53, 61, 16, 15, 49] that significantly advanced the research on this problem. The major shortcoming of the proposed methods is that they are working on the images produced by cameras' built-in ISPs and, thus, they are not using a significant part of the original sensor data lost in the ISP pipeline. In [39] the authors proposed to replace the smartphone ISP with a deep neural network learned to map directly the RAW Bayer sensor data to the higher-quality images captured by a DSLR camera. For this, a *Zurich RAW to RGB* dataset containing RAW-RGB image pairs from a mobile camera sensor and a high-end DSLR camera was collected. The proposed learned ISP reached the quality level of commercial ISP system of the Huawei P20 camera phone, and these results were further improved in [37, 7, 60, 43, 33]. In this challenge, we use a more advanced FujiFlim UltraISP dataset [27, 23] and additional efficiency-related constraints on the developed solutions. We target deep learning solutions capable to run on mobile GPUs. This is the second installment after the challenge conducted in conjunction with the Mobile AI 2021 CVPR workshop [18].

The deployment of AI-based solutions on portable devices usually requires an efficient model design based on a good understanding of the mobile processing units (*e.g.*CPUs, NPUs, GPUs, DSP) and their hardware particularities, including their memory constraints. We refer to [34, 30] for an extensive overview of mobile AI acceleration hardware, its particularities and performance. As shown in these works, the latest generations of mobile NPUs are reaching the performance of older-generation mid-range desktop GPUs. Nevertheless, a straightforward deployment of neural networks-based solutions on mobile devices is impeded by (i) a limited memory (*i.e.*, restricted amount of RAM) and (ii) a limited or lacking support of many common deep learning operators and layers. These impeding factors make the processing of high resolution inputs impossible with the standard NN models and require a careful adaptation or re-design to the constraints of mobile AI hardware. Such optimizations can employ a combination of various model techniques such as 16-bit / 8-bit [4, 42, 41, 73] and low-bit [3, 67, 40, 52] quantization, network pruning and compression [4, 25, 47, 51, 55], device- or NPU-specific adaptations, platform-aware neural architecture search [14, 63, 72, 69], *etc* .

The majority of competitions aimed at efficient deep learning models use standard desktop hardware for evaluating the solutions, thus the obtained models rarely show acceptable results when running on real mobile hardware with many specific constraints. In this *Mobile AI challenge*, we take a radically different approach and propose the participants to develop and evaluate their models directly on mobile devices. The goal of this competition is to design a fast and performant deep learning-based solution for the learned smartphone ISP problem. For this, the participants were provided with the Fujifilm UltraISP dataset consisting of thousands of paired photos captured with a normal mobile camera sensor and a professional 102MP medium-format FujiFilm GFX100 camera. The efficiency of the proposed solutions was evaluated on the Snapdragon 8 Gen 1 mobile platform capable of accelerating floating-point and quantized neural networks. All solutions developed in this challenge are fully compatible with the TensorFlow Lite framework [64], thus can be efficiently executed on various Linux and Android-based IoT platforms, smartphones and edge devices.

This challenge is a part of the *Mobile AI & AIM 2022 Workshops and Challenges* consisting of the following competitions:

- Learned Smartphone ISP on Mobile GPUs
- Power Efficient Video Super-Resolution on Mobile NPUs [29]
- Quantized Image Super-Resolution on Mobile NPUs [32]
- Efficient Single-Image Depth Estimation on Mobile Devices [24]
- Realistic Bokeh Effect Rendering on Mobile GPUs [38]
- Super-Resolution of Compressed Image and Video [74]
- Reversed Image Signal Processing and RAW Reconstruction [6]
- Instagram Filter Removal [45]

The results and solutions obtained in the previous *MAI 2021 Challenges* are described in our last year papers:

- Learned Smartphone ISP on Mobile NPUs [18]

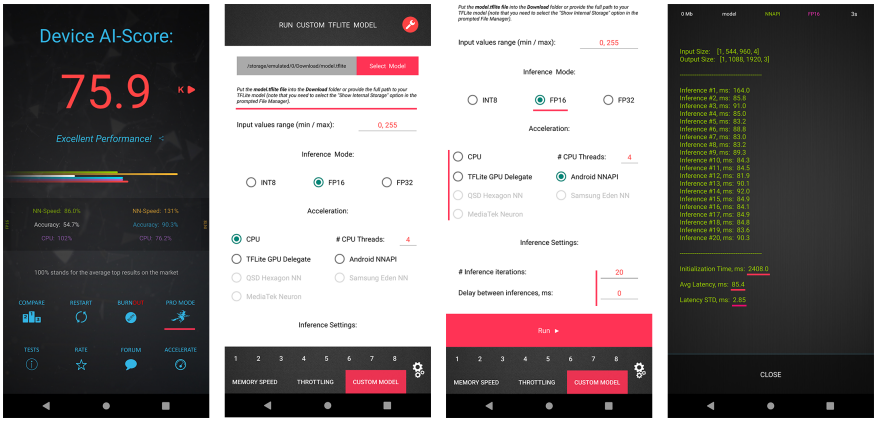

Fig. 2. Loading and running custom TensorFlow Lite models with AI Benchmark application. The currently supported acceleration options include Android NNAPI, TFLite GPU, Hexagon NN, Qualcomm QNN, MediaTek Neuron and Samsung ENN delegates as well as CPU inference through TFLite or XNNPACK backends. The latest app version can be downloaded at https://ai-benchmark.com/download.

- Real Image Denoising on Mobile GPUs [17]
- Quantized Image Super-Resolution on Mobile NPUs [31]
- Real-Time Video Super-Resolution on Mobile GPUs [57]
- Single-Image Depth Estimation on Mobile Devices [21]
- Quantized Camera Scene Detection on Smartphones [22]

2 Challenge

In order to design an efficient and practical deep learning-based solution for the considered task that runs fast on mobile devices, one needs the following tools:

1. A large-scale high-quality dataset for training and evaluating the models. Real, not synthetically generated data should be used to ensure a high quality of the obtained model;
2. An efficient way to check the runtime and debug the model locally without any constraints as well as the ability to check the runtime on the target evaluation platform.

This challenge addresses all the above issues. Real training data, tools, and runtime evaluation options provided to the challenge participants are described in the next sections.

2.1 Dataset

In this challenge, we use the Fujifilm UltraISP dataset collected using the Fujifilm GFX100 medium format 102 MP camera capturing the target high-quality images, and a popular Sony IMX586 Quad Bayer mobile camera sensor that can be found in tens of mid-range and high-end mobile devices released in the past 3 years. The Sony sensor was mounted on the MediaTek Dimensity 820 development board, and was capturing both raw and processed (by its built-in ISP system) 12MP images. The Dimensity board was rigidly attached to the Fujifilm camera, and they were shooting photos synchronously to ensure that the image content is identical. The dataset contains over 6 thousand daytime image pairs captured at a wide variety of places with different illumination and weather conditions. An example set of full-resolution photos from the Fujifilm UltraISP dataset is shown in Fig. 1. As the collected RAW-RGB image pairs were not perfectly aligned, they were initially matched using the state-of-the-art deep learning based dense matching algorithm [66] to extract 256×256 pixel patches from the original photos. It should be mentioned that all alignment operations were performed on Fujifilm RGB images only, therefore RAW photos from the Sony sensor remained unmodified, exhibiting the same values as read from the sensor.

2.2 Local Runtime Evaluation

When developing AI solutions for mobile devices, it is vital to be able to test the designed models and debug all emerging issues locally on available devices. For this, the participants were provided with the *AI Benchmark* application [30, 34] that allows to load any custom TensorFlow Lite model and run it on any Android device with all supported acceleration options. This tool contains the latest versions of *Android NNAPI, TFLite GPU, Hexagon NN, Qualcomm QNN, MediaTek Neuron* and *Samsung ENN* delegates, therefore supporting all current mobile platforms and providing the users with the ability to execute neural networks on smartphone NPUs, APUs, DSPs, GPUs and CPUs.

To load and run a custom TensorFlow Lite model, one needs to follow the next steps:

1. Download AI Benchmark from the official website[1] or from the Google Play[2] and run its standard tests.
2. After the end of the tests, enter the *PRO Mode* and select the *Custom Model* tab there.
3. Rename the exported TFLite model to *model.tflite* and put it into the *Download* folder of the device.
4. Select mode type *(INT8, FP16, or FP32)*, the desired acceleration/inference options and run the model.

These steps are also illustrated in Fig. 2.

[1] https://ai-benchmark.com/download
[2] https://play.google.com/store/apps/details?id=org.benchmark.demo

Team	Author	Framework	Model Size, MB	PSNR↑	SSIM↑	CPU Runtime, ms ↓	GPU Runtime, ms ↓	Final Score
MiAlgo	mialgo_ls	TensorFlow	0.014	23.33	0.8516	**135**	**6.8**	**14.87**
ENERZAi	MinsuKwon	TensorFlow	0.077	23.8	0.8652	208	18.9	10.27
HITZST01	Jaszheng	Keras / TensorFlow	0.060	23.89	0.8666	712	34.3	6.41
MINCHO	Minhyeok	TensorFlow	0.067	23.65	0.8658	886	41.5	3.8
ENERZAi	MinsuKwon	TensorFlow	4.5	24.08	0.8778	45956	212	1.35
HITZST01	Jaszheng	Keras / TensorFlow	1.2	**24.09**	0.8667	4694	482	0.6
JMU-CVLab	nanashi	Keras / TensorFlow	0.041	23.22	0.8281	3487	182	0.48
rainbow	zheng222	TensorFlow	1.0	21.66	0.8399	277	28	0.36
CASIA 1st	Zevin	PyTorch / TensorFlow	205	**24.09**	**0.884**	14792	1044	0.28
MiAlgo	mialgo_ls	PyTorch / TensorFlow	117	23.65	0.8673	15448	1164	0.14
DANN-ISP	gosha20777	TensorFlow	29.4	23.1	0.8648	97333	583	0.13
Multimedia *	lillythecutie	PyTorch/ OpenVINO	0.029	23.96	0.8543	293	11.4	21.24
SKD-VSP	dongdongdong	PyTorch / TensorFlow	78.9	24.08	0.8778	> 10 min	Failed	N.A.
CHannel Team	sawyerk2212	PyTorch / TensorFlow	102.0	22.28	0.8482	> 10 min	Failed	N.A.
MicroISP 1.0 [27]	Baseline	TensorFlow	0.152	23.87	0.8530	973	23.1	9.25
MicroISP 0.5 [27]	Baseline	TensorFlow	0.077	23.60	0.8460	503	15.6	9.43
PyNET-V2 Mobile [23]	Baseline	TensorFlow	3.6	24.72	0.8783	8342	194	3.58

Table 1. Mobile AI 2022 learned smartphone ISP challenge results and final rankings. The runtime values were obtained on Full HD (1920×1088) resolution images on the Snapdragon 8 Gen 1 mobile platform. The results of the MicroISP and PyNET-V2 Mobile models are provided for the reference. * The solution submitted by team *Multimedia* had corrupted weights due to incorrect model conversion, this issue was fixed after the end of the challenge.

2.3 Runtime Evaluation on the Target Platform

In this challenge, we use the the *Qualcomm Snapdragon 8 Gen 1* mobile SoC as our target runtime evaluation platform. The considered chipset demonstrates very decent AI Benchmark scores and can be found in the majority of flagship Android smartphones released in 2022. It can efficiently accelerate floating-point networks on its Adreno 730 GPU with a theoretical FP16 performance of 5 TFLOPS. The models were parsed and accelerated using the TensorFlow Lite GPU delegate [46] demonstrating the best performance on this platform when using general deep learning models. All final solutions were tested using the aforementioned AI Benchmark application.

2.4 Challenge Phases

The challenge consisted of the following phases:

I. *Development:* the participants get access to the data and AI Benchmark app, and are able to train the models and evaluate their runtime locally;
II. *Validation:* the participants can upload their models to the remote server to check the fidelity scores on the validation dataset, and to compare their results on the validation leaderboard;
III. *Testing:* the participants submit their final results, codes, TensorFlow Lite models, and factsheets.

2.5 Scoring System

All solutions were evaluated using the following metrics:

– Peak Signal-to-Noise Ratio (PSNR) measuring fidelity score,

Team	Author	Framework	Model Size, MB	PSNR↑	SSIM↑	MOS Score
HITZST01	Jaszheng	Keras / TensorFlow	1.2	**24.09**	0.8667	3.1
ENERZAi	MinsuKwon	TensorFlow	4.5	24.08	0.8778	3.1
CASIA 1st	Zevin	PyTorch / TensorFlow	205	**24.09**	**0.884**	3.0
Multimedia	lillythecutie	PyTorch/ OpenVINO	0.029	23.96	0.8543	3.0
HITZST01	Jaszheng	Keras / TensorFlow	0.060	23.89	0.8666	3.0
MINCHO	Minhyeok	TensorFlow	0.067	23.65	0.8658	3.0
ENERZAi	MinsuKwon	TensorFlow	0.077	23.8	0.8652	2.8
DANN-ISP	gosha20777	TensorFlow	29.4	23.1	0.8648	2.8
MiAlgo	mialgo_ls	TensorFlow	0.014	23.33	0.8516	2.5
JMU-CVLab	nanashi	Keras / TensorFlow	0.041	23.22	0.8281	2.3
MiAlgo	mialgo_ls	PyTorch / TensorFlow	117	23.65	0.8673	2.2

Table 2. Mean Opinion Scores (MOS) of all solutions submitted during the final phase of the MAI 2022 challenge and achieving a PSNR score of at least 23 dB. Visual results were assessed based on the reconstructed 12MP full resolution images.

- Structural Similarity Index Measure (SSIM), a proxy for perceptual score,
- The runtime on the target Snapdragon 8 Gen 1 platform.

In this challenge, the participants were able to submit their final models to two tracks. In the first track, the score of each final submission was evaluated based on the next formula (C is a constant normalization factor):

$$\text{Final Score} = \frac{2^{2 \cdot \text{PSNR}}}{C \cdot \text{runtime}},$$

In the second track, all submissions were evaluated only based on their visual results as measured by the corresponding Mean Opinion Scores (MOS). This was done to allow the participants to develop larger and more powerful models for the considered task.

During the final challenge phase, the participants did not have access to the test dataset. Instead, they had to submit their final TensorFlow Lite models that were subsequently used by the challenge organizers to check both the runtime and the fidelity results of each submission under identical conditions. This approach solved all the issues related to model overfitting, reproducibility of the results, and consistency of the obtained runtime/accuracy values.

3 Challenge Results

From the above 140 registered participants, 11 teams entered the final phase and submitted valid results, TFLite models, codes, executables, and factsheets. The proposed methods are described in Section 4, and the team members and affiliations are listed in Appendix A.

3.1 Results and Discussion

Tables 1 and 2 demonstrate the fidelity, runtime and MOS results of all solutions submitted during the final test phase. Models submitted to the 1st and 2nd challenge tracks were evaluated together since the participants had to upload the corresponding TensorFlow Lite models in both cases. In the 1st tack, the overall best results were achieved by team *Multimedia*. The authors proposed a novel *eReopConv* layer that consists of a large number of convolutions that are fused during the final model exporting stage to improve the its runtime while maintaining the fidelity scores. This approach turned out to be very efficient as the model submitted by this team was able to achieve one of the best PSNR and MOS scores as well as a runtime of less than 12 ms on the target Snapdragon platform. Unfortunately, the original TFLite file submitted by this team had corrupted weights caused by incorrect model conversion (the initial PyTorch model was converted to TFLite via ONNX), and this issue was fixed only after the end of the challenge.

The best runtime on the Snapdragon 8 Gen 1 was achieved by team *MiAlgo*, which solution is able to process one Full HD resolution photo on its GPU under 7 milliseconds. This efficiency was achieved due to a very shallow structure of the proposed 3-layer neural network, which architecture was inspired by the last year MAI challenge winner [18]. The visual results obtained by this model are also satisfactory, though significantly fall behind the results of the solution from team *Multimedia*. The second best result in the 1st challenge track was obtained by team *ENERZAi* that proposed a UNet-based model with channel attention blocks. The final structure of this solution was obtained using the neural architecture search modified to take the computational complexity of the model as an additional key penalty parameter.

After evaluating the visual quality of the proposed models, we obtained quite similar MOS scores for half of the submissions. A detailed inspection of the results revealed that their overall quality can be generally considered as relatively comparable, though none of the models was able to perform an ideal image reconstruction: each model had some issues either with color rendition or with noise suppression / texture rendering. Solutions with a MOS score of less than 2.8 were usually having several issues or were exhibiting noticeable image corruptions. These results highlighted again the difficulty of an accurate assessment of the results obtained in learned ISP task as the conventional fidelity metrics are often not indicating the real image quality.

4 Challenge Methods

This section describes solutions submitted by all teams participating in the final stage of the MAI 2022 Learned Smartphone ISP challenge.

4.1 MiAlgo

For track 1, team MiAlgo proposed a smaller three-convolution structure based on last year's MAI 2021 winning solution [18] (Fig. 3). The authors reduced

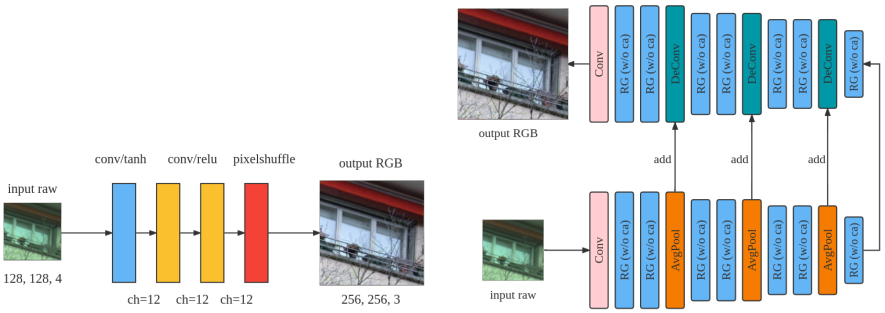

Fig. 3. Model architectures proposed by team MiAlgo for the 1st (left) and 2nd (right) challenge tracks.

the number of convolutional channels from 16 to 12, which also decreased the inference time by about 1/4. Besides that, the authors additionally used the distillation technique to remove the misalignment of some raw-RGB pairs in order to make the model converge better, which improved the PSNR score by about 0.3 dB. The model was trained for 10K epochs with L1 loss. The parameters were optimized with the Adam [44] algorithm using a batch size of 32 and a learning rate of 1e−4 that was decreased within the training.

For track 2, the authors proposed a 4-level UNet-based structure (Fig. 3, right). Several convolutional layers in the UNet [58] architecture were replaced with a residual group (RG, without channel attention layer) from RCAN [76] to enhance the reconstruction ability of the network. The authors used average pooling for the down-sampling layer and deconvolution for the up-sampling layer. The number of channels for each model level is 32, 64, 128, and 256 respectively. The model was first trained for 2K epochs with L1 loss, and then fine-tuned for about 2K epochs with the L1 and VGG loss functions. The initial learning rate was set to 1e–4 and was decreased within the training.

4.2 Multimedia

Inspired by a series of research on model re-parameterization from Ding [10, 9, 8], team Multimedia proposed an *enormous Re-parameter Convolution (eReopConv)* layer to replace the standard convolution. eReopConv has a large structure during training to learn superior proprieties but transforms into a less structured block during inference and retains these proprieties. The training and inference structures are shown in Fig. 4: unlike the RepConv in the RepVGG [10] that contain a 3×3 and 1×1 convolution layers in two branches during the training procedure, this method retains multi-branch convolution layers with different kernel sizes. *E.g.,* eRepConv with 5×5 kernel size has ten convolution layers with a kernel size ranging from 1×1 to 5×5 to collect enough information from different receptive fields. During the inference procedure, the training parameters are re-parameterized by continuous linear transforms.

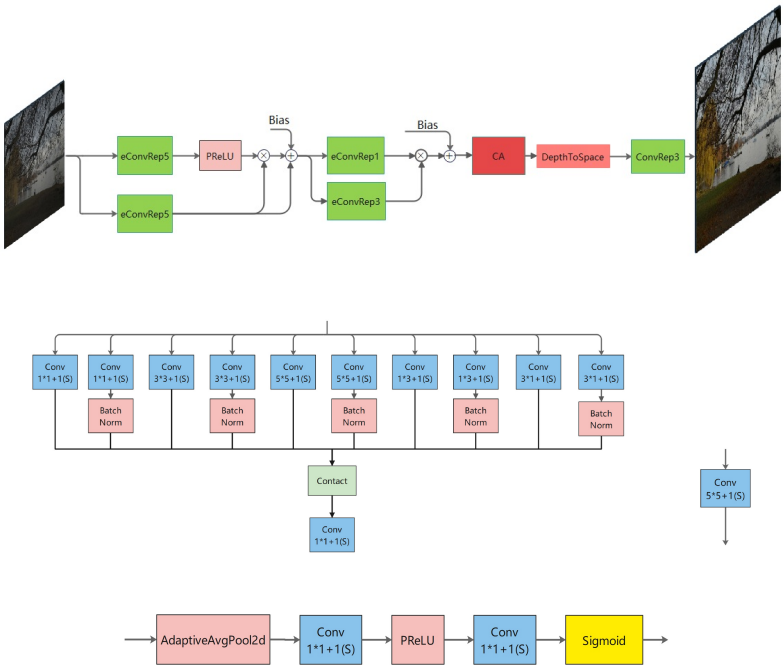

Fig. 4. The overall model architecture (top), the structure of the *eReopConv* block during the training and inference stages (middle), and the architecture of the CA module (bottom) proposed by team Multimedia.

As many spatial features are extracted by the eRepConv, a spatial attention mechanism could effectively improve the network performance. However, classical spatial attention blocks like [70] would increase the computational complexity because of the max pooling and average pooling operations. To save the extra costs, the authors utilize another eRepconv to produce a fine-granularity spatial attention block that has a specific attention matrix for each spatial feature in different channels. A learnable parameter is also added as a bias to expand this nonlinear expressivity. Although the fine-granularity spatial attention improves the performance, too many multiple operators and all channels fused by one convolution operator will inevitably make the model hard to train. To fix this, channel attention is added at the end of the network to help the network to converge since it could identify the importance of each channel.

Besides the Charbonnier loss and the cosine similarity loss, the authors propose to use an additional patch loss to enhance the image quality by considering the patch level information. In the patch loss, the ground truth y and the generated y' images are divided into a set of patches $\{y_{p1}, y_{p2}, ..., y_{pn}\}$ and $\{y'_{p1}, y'_{p2}, ...y'_{pn}\}$, then the mean and the variance of the difference between the generated and the ground truth patches is calculated and used as a weight for

each pixel in the patch. The exact formulation of this loss function is:

$$L_P = \sum_{i=0}^{n} e^{mean(y_{pi} - y'_{pi})^{-1} + var(y_{pi} - y'_{pi})^{-1}} \times |y - y'|, \quad (1)$$

where *mean* represents how big the difference between the two images is, and *var* reflects whether the two images have similar changing rates.

The model is trained in two stages. During the warming-up training stage, besides the RAW-to-RGB transformation task, the model also learns to perform masked raw data recovery: raw image is divided into a set of 3×3 patches, 50% of them are randomly masked out, and the network is trained to recover the masked patches. This MAE-liked [13] strategy has two benefits. First, defective pixels often occur in the raw data, and it is quite useful to be able to recover these pixels. Secondly, since it is harder for a network to perform these two tasks simultaneously, the resulting model is usually more robust. The learning rate of the warming-up stage is set to 1e–6, and the model is trained with the L1 loss only. Next, in the second normal train stage the learning rate is set to 1e–3 and decayed with a cosine annealing scheduler, and the model is optimized with the Adam for only 350 epochs.

4.3 ENERZAi

For track 1, team ENERZAi first constructed a search space for the target model architecture based on the UNet [58] and Hourglass architectures. Since a right balance between image colors is important when constructing RGB images from raw data, a channel attention module [76] was also included in the model space. Optimal model design (Fig. 5) was obtained using an architecture search algorithm that was based on the evolutionary algorithm [56] and was taking latency into account by penalizing computationally heavy models.

The authors used L1 loss, MS-SIM and ResNet50-based perceptual loss functions to train the model. In addition, the authors developed a differentiable approximate histogram feature inspired by the Histogan [1] paper to consider the color distribution of the constructed RGB image. For each R, G, and B channel, the histogram differences between the constructed image and the target image were calculated and added to the total loss. The Adam optimizer with a learning rate of 0.001 was used for training the model, and the learning rate was halved each several epochs.

For track 2 (Fig. 5, bottom), the authors used the same approach but without penalizing computationally heavy models, which resulted in the increased number of convolutional layers and channel sizes.

4.4 HITZST01

Team HITZST01 proposed the RFD-CSA architecture [71] for the considered problem demonstrated in Fig. 6. The model consists of there modules: the Source Features Module, the Enhance Features Module and the Upsample Features

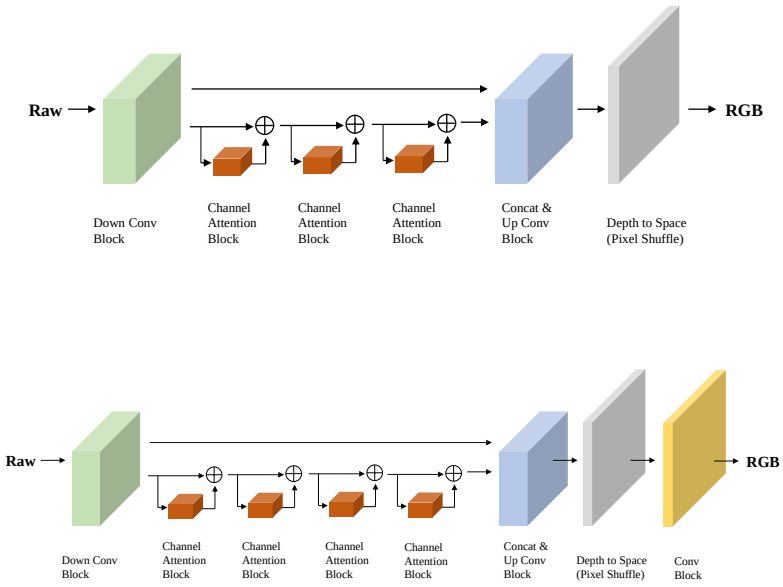

Fig. 5. Model designs obtained by team ENERZAi for the 1st (top) and 2nd (bottom) challenge tracks using the neural architecture search.

Module. The purpose of the Source Features Module is to extract rough features from the original raw images. This module is based on the ECB/Conv3 architecture proposed in [75] with the re-parameterization technique used to boost the performance while keeping the architecture efficient.

The Enhance Features Module is designed to extract effective features at multiple model levels. This module consists of n lightweight multi-level feature extraction structures (with n equal to 2 and 3 for models submitted to the 1st and 2nd challenge tracks, respectively). These structures are modified Residual feature distillation blocks (RFDB) proposed in [50], where the CCA-layer in RFDB is replaced with the CSA block. The authors additionally added a long-term residual connection in this module to avoid the performance degradation caused by model's depth and to boost its efficiency. The Upsample Features Module generates the final image reconstruction results. To improve the performance, ECB/Conv3 blocks are added at the beginning and at the end of this module. The model was trained with a combination of the Charbonnier and SSIM losses using the Adam optimizer with the initial learning rate set to 1e–4 and halved every 200 epochs.

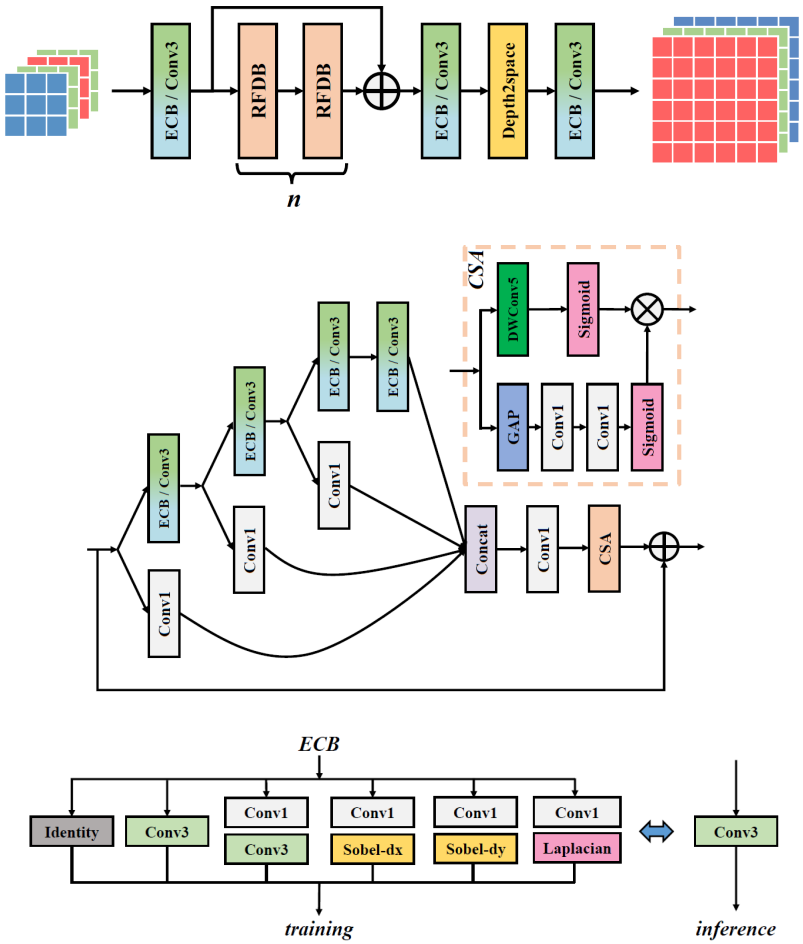

Fig. 6. The RFD-CSA architecture proposed by team HITZST01.

4.5 MINCHO

The architecture of the model developed by team MINCHO consists of the two main parts (Fig. 7). The first part is used to extract image features and consists of one convolutional blocks with $ReLU$ activations, depthwise convolutional blocks, pointwise convolutional blocks with $ReLU$ activations, and a skip connection. The second part of the model is the ISP part that is based on the Smallnet architecture [18] with three convolutional and one pixel-shuffle layer. The model was first trained with L1 loss only, and then fine-tuned with a combination of the L2, perceptual-based VGG-19 and SSIM loss functions. Model parameters were optimized using the ADAM algorithm with $\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-8}$, a learning rate of 10^{-4}, and a batch size of 32.

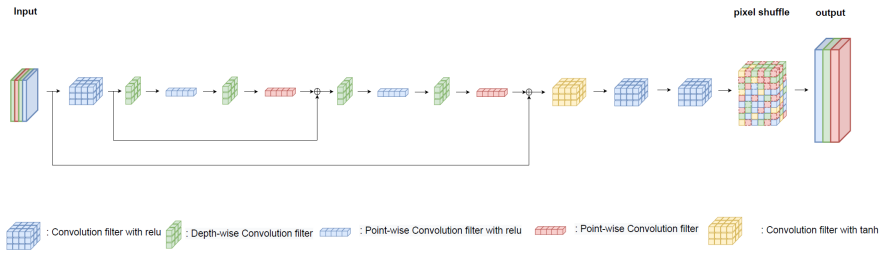

Fig. 7. The architecture proposed by team MINCHO.

4.6 CASIA 1st

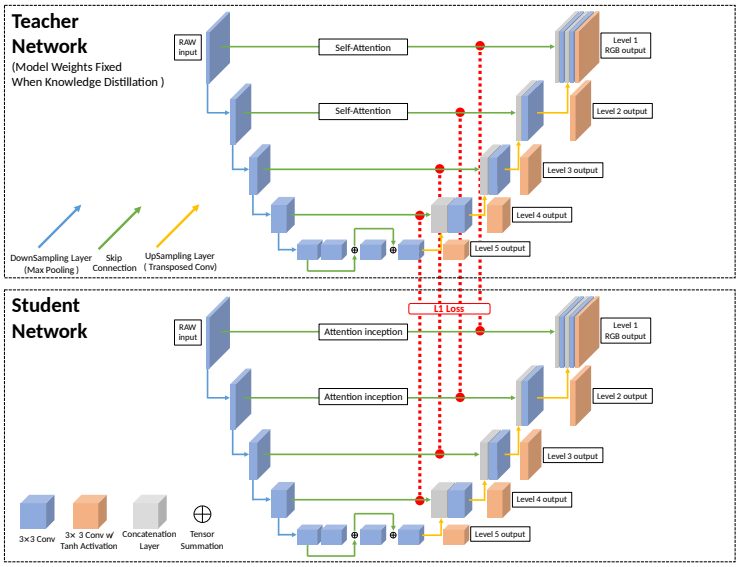

Fig. 8. An overview of the model and teacher-guided training strategy proposed by team CASIA 1st.

Team CASIA 1st proposed a two-stage teacher-guided model training strategy for the learned ISP problem (Fig. 8). At the first stage, the authors trained a teacher network (TN) to get good PSNR and SSIM scores. They used the PUNET network [18] as a baseline to develop an attention-aware based Unet (AA-Unet) architecture utilizing a self-attention module. Inspired by [39], the authors applied a multi-loss constraint to features at different model levels.

In the second stage, the authors designed a relatively tiny student network (SN) to inherit knowledge from the teacher network through model distillation.

The student network is very similar to the teacher model. Inspired by [62], the authors connected three attention modules (CBAM, ECA, and SEA) in parallel to form the attention inception module that was used to replace all self-attention modules in the TN to reduce the time complexity. Since the self-attention and attention inception modules have the same input/output shapes, the model distillation strategy was straightforward: the feature maps from each level in the TN model were extracted and used as soft-labels to train SN.

The models were trained with the MSE, L1, SSIM, VGG and edge [59] loss functions. When training the TN, they applied MSE loss to level 2, 3 and 4, SSIM loss to level 2 and 3, and MSE, SSIM, VGG, Edge loss to level 1. When training SN, they fixed the model weight of TN and applied L1 loss to each level with the soft-label. Similar to TN, MSE, SSIM and VGG loss are also applied to level 1. Both TN and SN model parameters were optimized using Adam with a batch size of 12, a learning rate of 5e–5 and a weight decay of 5e–5.

4.7 JMU-CVLab

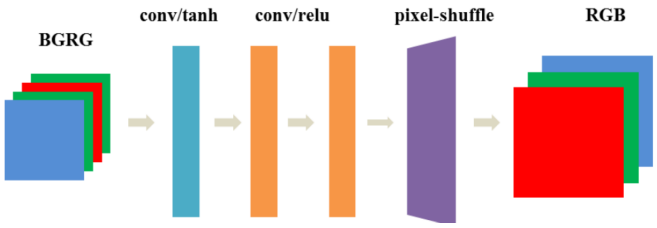

Fig. 9. Model architecture used by the JMU-CVLab team.

Team JMU-CVLab based their solution on the Smallnet model developed in the previous MAI 2021 challenge [18] (Fig. 9). The authors replaced the PixelShuffle layer with the Depth2Space op and added two well-known non-linear ISP operations: tone mapping and gamma correction [5]. These operations were applied to the final RGB reconstruction result followed by an additional CBAM attention block [70] applied to allow the model to learn more complex features. The model was trained to minimize the L1 and SSIM losses for 60 epochs with the Adam optimizer. A batch size of 32 with a learning rate of 0.0001 was used, basic augmentations including flips and rotations were applied to the training data.

4.8 DANN-ISP

Team DANN-ISP developed a domain adaptation method for the considered problem, which is illustrated in Fig. 10. The authors trained their model to generate RGB images with both source and target domains as inputs. Two

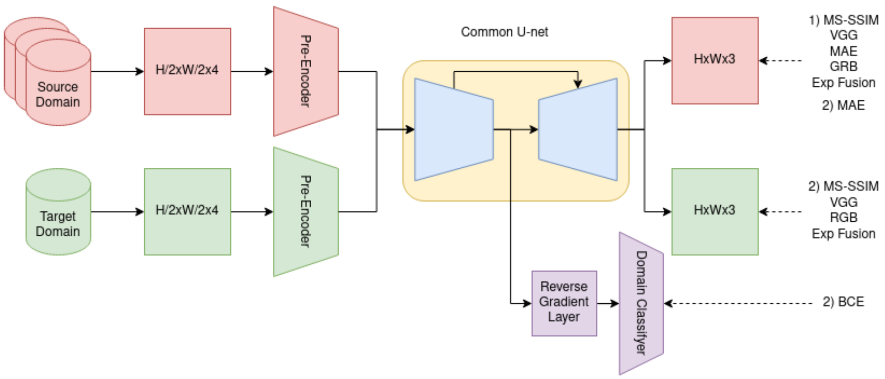

Fig. 10. Domain adaptation strategy developed by team DANN-ISP.

pre-encoders were used to reduce the significant domain gap between different cameras by extracting individual and independent features from each one. The architecture of the pre-encoders consisted of three 3×3 convolutional layers with 8, 16 and 32 filters, respectively. Next, the authors used a lightweight U-Net-like [58] autoencoder with three downsampling and four upsampling blocks. It takes 32-channel features from each pre-encoder as input and produces two outputs: a 3-channel RGB image and a 256-dimensional feature vector from its bottleneck. Finally, a binary domain classifier [12] with an inverse gradient [11] was used to reduce the gap between domains and increase the performance of the model. This classifier was constructed from the global average pooling layer and two dense layers at the end.

First, the model was pre-trained using only source domain data (in this case – using the Zurich-RAW-to-RGB dataset [39]) with the L1, MS-SSIM, VGG-based, color and exposure fusion losses. At the second stage, both source and target domain data were used together, and the model was trained to minimize a combination of the above losses plus the binary domain classifier loss (cross-entropy).

4.9 Rainbow

Team rainbow proposed a U-Net based AWBUnet model for this task. In traditional ISP pipelines, demosaicing can be solved by a neural network naturally, and white balance needs to be determined based on the input image. Therefore, the authors propose an RGB gain module (Fig. 11) consisting of a global average pooling (GAP) layer and three fully connected layers to adjust the white balance (RG channels) and brightness (RGB channels). The model was trained using a mini-batch size of 256, random horizontal flips were applied for data augmentation. The model was trained to minimizing the MSE loss function using the Adam optimizer. The initial learning rate was set to 2e–4 and halved every 100K iterations.

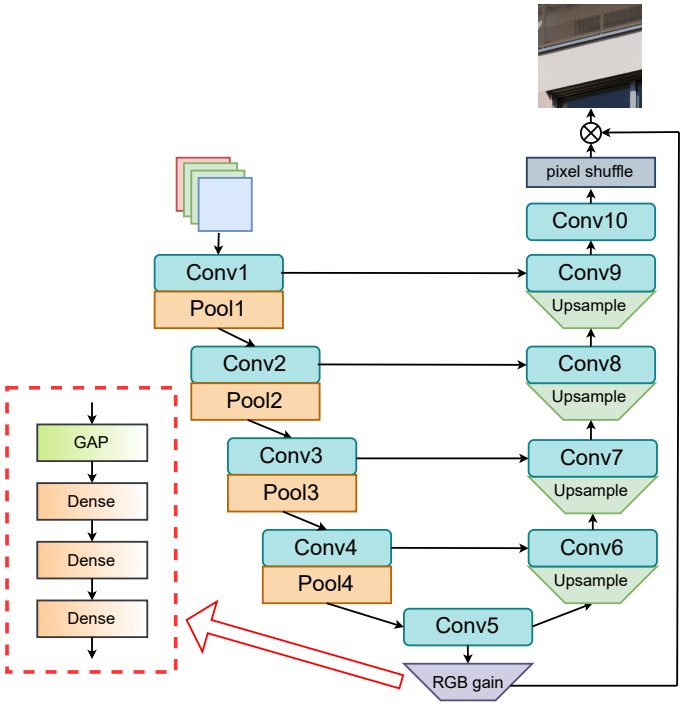

Fig. 11. AWBUnet model proposed by team rainbow.

4.10 SKD-VSP

Team SKD-VSP applied the discrete wavelet transform to divide the image into high-frequency and low-frequency parts, and used two different networks to process these two parts separately (Fig. 12). After processing, the inverse discrete wavelet transform was used to restore the final image. The architecture of the considered networks is based on the EDSR [48] model. Since the low-frequency part contains most of the global image information, the authors used 16 residual blocks with 256 channels in each block. For the high frequency part, 8 residual blocks with 64 channels were used due to its simple structure. When DWT operates on the image, the high-frequency part is divided into high-frequency information in three directions: horizontal, vertical and diagonal, so the input and output shape of the high-frequency network is $3 \times 3 \times \frac{H}{2} \times \frac{W}{2}$.

The authors used the L1 Loss for training the high frequency part, while for the low frequency part a combination of the L1, perceptual VGG-based and SSIM losses were utilized. Adam optimizer was used to train the model for 100 epochs with the initial learning rate set to 1e–3 and attenuated by a factor of 0.5 every 25 epochs.

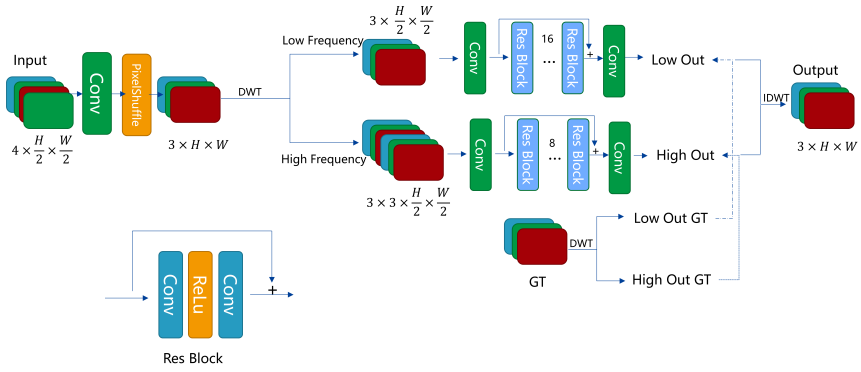

Fig. 12. The architecture of the model proposed by team SKD-VSP. DWT stands for the discrete wavelet transform.

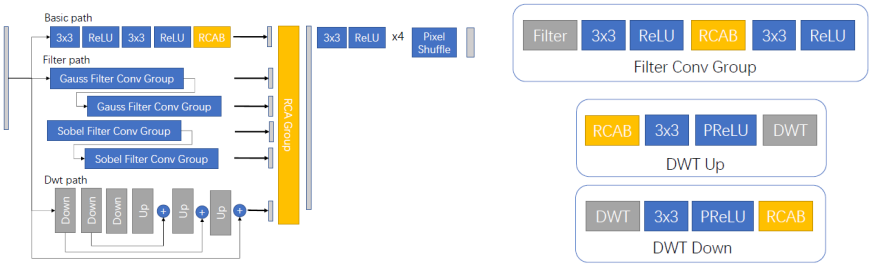

Fig. 13. GaUss-DWT model architecture proposed by the CHannel Team.

4.11 CHannel Team

CHannel Team proposed a multi-path GaUss-DWT model, which architecture is illustrated in Fig. 13. The model uses three paths to extract different information aspects from raw images: the basic path, the filter path, and the DWT path. The basic path consists of two convolutional layers and a residual attention block. This simple structure provides the model with a basic understanding of the image. The filter path consists of two different filters: Gauss and Sobel that extract chroma and texture information separately. Each convolutional group contains a fixed filter, two convolutional layers, and a residual channel attention block. The DWT path was inspired by the MW-ISPNet [37] model that attains high fidelity results in the AIM 2020 challenge. However, its structure was simplified by replacing its residual channel attention groups with residual channel attention blocks. This modification allowed to reduce the runtime of the model

while attaining satisfactory results. Finally, a residual channel attention group integrates the extracted information from all three paths. Its output is then processed by four convolutional layers and a pixel shuffle layer to get the final image result. The model was trained with a combination of the perceptual VGG-based, MSE, L1, MS-SSIM and edge loss functions. Model parameters were optimized using the Adam algorithm with a learning rate of 1e–4 and a batch size of 8.

5 Additional Literature

An overview of the past challenges on mobile-related tasks together with the proposed solutions can be found in the following papers:

- Learned End-to-End ISP: [33, 37, 27, 23]
- Perceptual Image Enhancement: [36, 28]
- Bokeh Effect Rendering: [26, 35]
- Image Super-Resolution: [36, 54, 2, 65]

Acknowledgements

We thank the sponsors of the Mobile AI and AIM 2022 workshops and challenges: AI Witchlabs, MediaTek, Huawei, Reality Labs, OPPO, Synaptics, Raspberry Pi, ETH Zürich (Computer Vision Lab) and University of Würzburg (Computer Vision Lab).

A Teams and Affiliations

Mobile AI 2022 Team

Title:
Mobile AI 2022 Learned Smartphone ISP Challenge
Members:
Andrey Ignatov[1,2] *(andrey@vision.ee.ethz.ch)*, Radu Timofte[1,2,3]
Affiliations:
[1] Computer Vision Lab, ETH Zurich, Switzerland
[2] AI Witchlabs, Switzerland
[3] University of Wuerzburg, Germany

MiAlgo

Title:
3Convs and BigUNet for Smartphone ISP
Members:

Shuai Liu (liushuai21@xiaomi.com), Chaoyu Feng, Furui Bai, Xiaotao Wang, Lei Lei
Affiliations:
Xiaomi Inc., China

Multimedia

Title:
FGARepNet: A real-time end-to-end ISP network based on Fine-Granularity attention and Re-parameter convolution
Members:
Ziyao Yi (yi.ziyao@sanechips.com.cn), Yan Xiang, Zibin Liu, Shaoqing Li, Keming Shi, Dehui Kong, Ke Xv
Affiliations:
Sanechips Co. Ltd, China

ENERZAi Research

Title:
Latency-Aware NAS and Histogram Feature Loss
Members:
Minsu Kwon (minsu.kwon@enerzai.com)
Affiliations:
ENERZAi, Seoul, Korea
enerzai.com

HITZST01

Title:
Residual Feature Distillation Channel Spatial Attention Network for ISP on Smartphones [71]
Members:
Yaqi Wu[1] *(titimasta@163.com)*, Jiesi Zheng[2], Zhihao Fan[3], Xun Wu[4], Feng Zhang
Affiliations:
[1] Harbin Institute of Technology, China
[2] Zhejiang University, China
[3] University of Shanghai for Science and Technology, China
[4] Tsinghua University, China

MINCHO

Title:
Mobile-Smallnet: Smallnet with MobileNet blocks for an end-to-end ISP Pipeline
Members:
Albert No (albertno@hongik.ac.kr), Minhyeok Cho
Affiliations:
Hongik University, Korea

CASIA 1st

Title:
Learned Smartphone ISP Based On Distillation Acceleration
Members:
Zewen Chen[1] *(chenzewen2022@ia.ac.cn)*, Xiaze Zhang[2], Ran Li[3], Juan Wang[1], Zhiming Wang[4]
Affiliations:
[1] Institute of Automation, Chinese Academy of Sciences, China
[2] School of Computer Science, Fudan University, China
[3] Washington University in St. Louis
[4] Tsinghua University, China

JMU-CVLab

Title:
Shallow Non-linear CNNs as ISP
Members:
Marcos V. Conde (marcos.conde-osorio@uni-wuerzburg.de), Ui-Jin Choi
Affiliations:
University of Wuerzburg, Germany

DANN-ISP

Title:
Learning End-to-End Deep Learning Based Image Signal Processing Pipeline Using Adversarial Domain Adaptation
Members:
Georgy Perevozchikov (perevozchikov.gp@phystech.edu), Egor Ershov
Affiliations:
Moscow Institute of Physics and Technology, Russia

Rainbow

Title:
Auto White Balance UNet for Learned Smartphone ISP
Members:
Zheng Hui (huizheng.hz@alibaba-inc.com)
Affiliations:
Alibaba DAMO Academy, China

SKD-VSP

Title:
IFS Net-Image Frequency Separation Residual Network
Members:
Mengchuan Dong (mengchuan61@gmail.com), Wei Zhou, Cong Pang
Affiliations:
ShanghaiTech University, China

CHannel Team

Title:
GaUss-DWT net
Members:
Haina Qin (qinhaina2020@ia.ac.cn), Mingxuan Cai
Affiliations:
Institute of Automation, Chinese Academy of Sciences, China

References

1. Afifi, M., Brubaker, M.A., Brown, M.S.: Histogan: Controlling colors of gan-generated and real images via color histograms. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 7941–7950 (2021)
2. Cai, J., Gu, S., Timofte, R., Zhang, L.: Ntire 2019 challenge on real image super-resolution: Methods and results. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
3. Cai, Y., Yao, Z., Dong, Z., Gholami, A., Mahoney, M.W., Keutzer, K.: Zeroq: A novel zero shot quantization framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13169–13178 (2020)
4. Chiang, C.M., Tseng, Y., Xu, Y.S., Kuo, H.K., Tsai, Y.M., Chen, G.Y., Tan, K.S., Wang, W.T., Lin, Y.C., Tseng, S.Y.R., et al.: Deploying image deblurring across mobile devices: A perspective of quality and latency. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 502–503 (2020)

5. Conde, M.V., McDonagh, S., Maggioni, M., Leonardis, A., Pérez-Pellitero, E.: Model-based image signal processors via learnable dictionaries. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 481–489 (2022)
6. Conde, M.V., Timofte, R., et al.: Reversed Image Signal Processing and RAW Reconstruction. AIM 2022 Challenge Report. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops (2022)
7. Dai, L., Liu, X., Li, C., Chen, J.: Awnet: Attentive wavelet network for image isp. arXiv preprint arXiv:2008.09228 (2020)
8. Ding, X., Hao, T., Tan, J., Liu, J., Han, J., Guo, Y., Ding, G.: Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4510–4520 (2021)
9. Ding, X., Xia, C., Zhang, X., Chu, X., Han, J., Ding, G.: Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition. arXiv preprint arXiv:2105.01883 (2021)
10. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: Repvgg: Making vgg-style convnets great again. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13733–13742 (2021)
11. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: International conference on machine learning. pp. 1180–1189. PMLR (2015)
12. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The journal of machine learning research **17**(1), 2096–2030 (2016)
13. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16000–16009 (2022)
14. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1314–1324 (2019)
15. Huang, J., Zhu, P., Geng, M., Ran, J., Zhou, X., Xing, C., Wan, P., Ji, X.: Range scaling global u-net for perceptual image enhancement on mobile devices. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. pp. 0–0 (2018)
16. Hui, Z., Wang, X., Deng, L., Gao, X.: Perception-preserving convolutional networks for image enhancement on smartphones. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. pp. 0–0 (2018)
17. Ignatov, A., Byeoung-su, K., Timofte, R.: Fast camera image denoising on mobile gpus with deep learning, mobile ai 2021 challenge: Report. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2021)
18. Ignatov, A., Chiang, J., Kuo, H.K., Sycheva, A., Timofte, R.: Learned smartphone isp on mobile npus with deep learning, mobile ai 2021 challenge: Report. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2021)
19. Ignatov, A., Kobyshev, N., Timofte, R., Vanhoey, K., Van Gool, L.: Dslr-quality photos on mobile devices with deep convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3277–3285 (2017)
20. Ignatov, A., Kobyshev, N., Timofte, R., Vanhoey, K., Van Gool, L.: Wespe: weakly supervised photo enhancer for digital cameras. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 691–700 (2018)

21. Ignatov, A., Malivenko, G., Plowman, D., Shukla, S., Timofte, R.: Fast and accurate single-image depth estimation on mobile devices, mobile ai 2021 challenge: Report. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2021)
22. Ignatov, A., Malivenko, G., Timofte, R.: Fast and accurate quantized camera scene detection on smartphones, mobile ai 2021 challenge: Report. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2021)
23. Ignatov, A., Malivenko, G., Timofte, R., Tseng, Y., Xu, Y.S., Yu, P.H., Chiang, C.M., Kuo, H.K., Chen, M.H., Cheng, C.M., Van Gool, L.: Pynet-v2 mobile: Efficient on-device photo processing with neural networks. In: 2021 26th International Conference on Pattern Recognition (ICPR). IEEE (2022)
24. Ignatov, A., Malivenko, G., Timofte, R., et al.: Efficient single-image depth estimation on mobile devices, mobile ai & aim 2022 challenge: Report. In: European Conference on Computer Vision (2022)
25. Ignatov, A., Patel, J., Timofte, R.: Rendering natural camera bokeh effect with deep learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 418–419 (2020)
26. Ignatov, A., Patel, J., Timofte, R., Zheng, B., Ye, X., Huang, L., Tian, X., Dutta, S., Purohit, K., Kandula, P., et al.: Aim 2019 challenge on bokeh effect synthesis: Methods and results. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 3591–3598. IEEE (2019)
27. Ignatov, A., Sycheva, A., Timofte, R., Tseng, Y., Xu, Y.S., Yu, P.H., Chiang, C.M., Kuo, H.K., Chen, M.H., Cheng, C.M., Van Gool, L.: Microisp: Processing 32mp photos on mobile devices with deep learning. In: European Conference on Computer Vision (2022)
28. Ignatov, A., Timofte, R.: Ntire 2019 challenge on image enhancement: Methods and results. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
29. Ignatov, A., Timofte, R., Chiang, C.M., Kuo, H.K., Xu, Y.S., Lee, M.Y., Lu, A., Cheng, C.M., Chen, C.C., Yong, J.Y., Shuai, H.H., Cheng, W.H., et al.: Power efficient video super-resolution on mobile npus with deep learning, mobile ai & aim 2022 challenge: Report. In: European Conference on Computer Vision (2022)
30. Ignatov, A., Timofte, R., Chou, W., Wang, K., Wu, M., Hartley, T., Van Gool, L.: Ai benchmark: Running deep neural networks on android smartphones. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. pp. 0–0 (2018)
31. Ignatov, A., Timofte, R., Denna, M., Younes, A.: Real-time quantized image super-resolution on mobile npus, mobile ai 2021 challenge: Report. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2021)
32. 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: European Conference on Computer Vision (2022)
33. Ignatov, A., Timofte, R., Ko, S.J., Kim, S.W., Uhm, K.H., Ji, S.W., Cho, S.J., Hong, J.P., Mei, K., Li, J., et al.: Aim 2019 challenge on raw to rgb mapping: Methods and results. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 3584–3590. IEEE (2019)
34. 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.

In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 3617–3635. IEEE (2019)
35. Ignatov, A., Timofte, R., Qian, M., Qiao, C., Lin, J., Guo, Z., Li, C., Leng, C., Cheng, J., Peng, J., et al.: Aim 2020 challenge on rendering realistic bokeh. In: European Conference on Computer Vision. pp. 213–228. Springer (2020)
36. Ignatov, A., Timofte, R., Van Vu, T., Minh Luu, T., X Pham, T., Van Nguyen, C., Kim, Y., Choi, J.S., Kim, M., Huang, J., et al.: Pirm challenge on perceptual image enhancement on smartphones: Report. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. pp. 0–0 (2018)
37. Ignatov, A., Timofte, R., Zhang, Z., Liu, M., Wang, H., Zuo, W., Zhang, J., Zhang, R., Peng, Z., Ren, S., et al.: Aim 2020 challenge on learned image signal processing pipeline. arXiv preprint arXiv:2011.04994 (2020)
38. Ignatov, A., Timofte, R., et al.: Realistic bokeh effect rendering on mobile gpus, mobile ai & aim 2022 challenge: Report (2022)
39. Ignatov, A., Van Gool, L., Timofte, R.: Replacing mobile camera isp with a single deep learning model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 536–537 (2020)
40. Ignatov, D., Ignatov, A.: Controlling information capacity of binary neural network. Pattern Recognition Letters **138**, 276–281 (2020)
41. 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. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2704–2713 (2018)
42. Jain, S.R., Gural, A., Wu, M., Dick, C.H.: Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. arXiv preprint arXiv:1903.08066 (2019)
43. Kim, B.H., Song, J., Ye, J.C., Baek, J.: Pynet-ca: enhanced pynet with channel attention for end-to-end mobile image signal processing. In: European Conference on Computer Vision. pp. 202–212. Springer (2020)
44. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
45. Kınlı, F.O., Menteş, S., Özcan, B., Kirac, F., Timofte, R., et al.: Aim 2022 challenge on instagram filter removal: Methods and results. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops (2022)
46. Lee, J., Chirkov, N., Ignasheva, E., Pisarchyk, Y., Shieh, M., Riccardi, F., Sarokin, R., Kulik, A., Grundmann, M.: On-device neural net inference with mobile gpus. arXiv preprint arXiv:1907.01989 (2019)
47. Li, Y., Gu, S., Gool, L.V., Timofte, R.: Learning filter basis for convolutional neural network compression. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5623–5632 (2019)
48. Lim, B., Son, S., Kim, H., Nah, S., Mu Lee, K.: Enhanced deep residual networks for single image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 136–144 (2017)
49. Liu, H., Navarrete Michelini, P., Zhu, D.: Deep networks for image-to-image translation with mux and demux layers. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. pp. 0–0 (2018)
50. Liu, J., Tang, J., Wu, G.: Residual feature distillation network for lightweight image super-resolution. In: European Conference on Computer Vision. pp. 41–55. Springer (2020)

51. Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, K.T., Sun, J.: Metapruning: Meta learning for automatic neural network channel pruning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3296–3305 (2019)
52. Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., Cheng, K.T.: Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In: Proceedings of the European conference on computer vision (ECCV). pp. 722–737 (2018)
53. Lugmayr, A., Danelljan, M., Timofte, R.: Unsupervised learning for real-world super-resolution. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 3408–3416. IEEE (2019)
54. Lugmayr, A., Danelljan, M., Timofte, R.: Ntire 2020 challenge on real-world image super-resolution: Methods and results. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 494–495 (2020)
55. Obukhov, A., Rakhuba, M., Georgoulis, S., Kanakis, M., Dai, D., Van Gool, L.: T-basis: a compact representation for neural networks. In: International Conference on Machine Learning. pp. 7392–7404. PMLR (2020)
56. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence. vol. 33, pp. 4780–4789 (2019)
57. Romero, A., Ignatov, A., Kim, H., Timofte, R.: Real-time video super-resolution on smartphones with deep learning, mobile ai 2021 challenge: Report. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2021)
58. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
59. Seif, G., Androutsos, D.: Edge-based loss function for single image super-resolution. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1468–1472. IEEE (2018)
60. Silva, J.I.S., Carvalho, G.G., Santos, M.S., Santiago, D.J., de Albuquerque, L.P., Battle, J.F.P., da Costa, G.M., Ren, T.I.: A deep learning approach to mobile camera image signal processing. In: Anais Estendidos do XXXIII Conference on Graphics, Patterns and Images. pp. 225–231. SBC (2020)
61. de Stoutz, E., Ignatov, A., Kobyshev, N., Timofte, R., Van Gool, L.: Fast perceptual image enhancement. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. pp. 0–0 (2018)
62. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI conference on artificial intelligence (2017)
63. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2820–2828 (2019)
64. TensorFlow-Lite: https://www.tensorflow.org/lite
65. Timofte, R., Gu, S., Wu, J., Van Gool, L.: Ntire 2018 challenge on single image super-resolution: Methods and results. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 852–863 (2018)
66. Truong, P., Danelljan, M., Van Gool, L., Timofte, R.: Learning accurate dense correspondences and when to trust them. arXiv preprint arXiv:2101.01710 (2021)

67. Uhlich, S., Mauch, L., Cardinaux, F., Yoshiyama, K., Garcia, J.A., Tiedemann, S., Kemp, T., Nakamura, A.: Mixed precision dnns: All you need is a good parametrization. arXiv preprint arXiv:1905.11452 (2019)
68. Vu, T., Van Nguyen, C., Pham, T.X., Luu, T.M., Yoo, C.D.: Fast and efficient image quality enhancement via desubpixel convolutional neural networks. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. pp. 0–0 (2018)
69. Wan, A., Dai, X., Zhang, P., He, Z., Tian, Y., Xie, S., Wu, B., Yu, M., Xu, T., Chen, K., et al.: Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12965–12974 (2020)
70. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: Cbam: Convolutional block attention module. In: Proceedings of the European conference on computer vision (ECCV). pp. 3–19 (2018)
71. yaqi wu, zheng, J., zhihao Fan, Wu, X., Zhang, F.: Residual feature distillation channel spatial attention network for isp on smartphone. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops (2022)
72. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10734–10742 (2019)
73. Yang, J., Shen, X., Xing, J., Tian, X., Li, H., Deng, B., Huang, J., Hua, X.s.: Quantization networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7308–7316 (2019)
74. Yang, R., Timofte, R., et al.: Aim 2022 challenge on super-resolution of compressed image and video: Dataset, methods and results. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops (2022)
75. Zhang, X., Zeng, H., Zhang, L.: Edge-oriented convolution block for real-time super resolution on mobile devices. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 4034–4043 (2021)
76. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image super-resolution using very deep residual channel attention networks. In: Proceedings of the European conference on computer vision (ECCV). pp. 286–301 (2018)