Learned Smartphone ISP on Mobile NPUs with Deep Learning,
Mobile AI 2021 Challenge: Report

Andrey Ignatov  Cheng-Ming Chiang  Hsien-Kai Kuo  Anastasia Sycheva
Radu Timofte  Min-Hung Chen  Man-Yu Lee  Yu-Syuan Xu  Yu Tseng
Shusong Xu  Jin Guo  Chao-Hung Chen  Ming-Chun Hsyu  Wen-Chia Tsai
Chao-Wei Chen  Grigory Malivenko  Minsu Kwon  Myungie Lee  Jaeyoon Yoo
Changbeom Kang  Shinjo Wang  Zheng Shaolong  Hao Dejun  Xie Fen
Feng Zhuang  Yipeng Ma  Jingyang Peng  Tao Wang  Fenglong Song
Chih-Chung Hsu  Kwan-Lin Chen  Ngi-Huang Wu  Vishal Chudasama
Kalpesh Prajapati  Heena Patel  Anjali Sarvaiya  Kishor Upla  Kiran Raja
Raghavendra Ramachandra  Christoph Busch  Etienne de Stoutz

Abstract

As the quality of mobile cameras starts to play a crucial role in modern smartphones, more and more attention is now being paid to ISP algorithms used to improve various perceptual aspects of mobile photos. In this Mobile AI challenge, the target was to develop an end-to-end deep learning-based image signal processing (ISP) pipeline that can replace classical hand-crafted ISPs and achieve nearly real-time performance on smartphone NPUs. For this, the participants were provided with a novel learned ISP dataset consisting of RAW-RGB image pairs captured with the Sony IMX586 Quad Bayer mobile sensor and a professional 102-megapixel medium format camera. The runtime of all models was evaluated on the MediaTek Dimensity 1000+ platform with a dedicated AI processing unit capable of accelerating both floating-point and quantized neural networks. The proposed solutions are fully compatible with the above NPU and are capable of processing Full HD photos under 60-100 milliseconds while achieving high fidelity results. A detailed description of all models developed in this challenge is provided in this paper.

1. Introduction

While the quality of modern smartphone cameras increases gradually, many major improvements are currently coming from advanced image processing algorithms used, e.g., to perform noise suppression, accurate color reconstruction or high dynamic range processing. Though the image enhancement task can be efficiently solved with deep learning-based approaches, the biggest challenge here comes from getting the appropriate training data and, in particular, the high-quality ground truth images. The problem of end-to-end mobile photo quality enhancement was first addressed in [16, 17], where the authors proposed to enhance all aspects of low-quality smartphone photos by mapping them to superior-quality images obtained with a high-end reflex camera. The collected DPED dataset was later used in many subsequent competitions [29, 23] and works [51, 41, 14, 13, 38] that have significantly improved the results on this problem. While the proposed methods were quite efficient, they worked with the data produced by smartphones’ built-in ISPs, thus a significant part of information present in the original sensor data was irrecoverably lost after applying many image processing steps. To address this problem, in [31] the authors proposed to work directly with the original RAW Bayer sensor data and learn all ISP steps with a single deep neural network. The experiments conducted on the collected Zurich RAW to RGB dataset containing RAW-RGB image pairs captured by a mobile camera sensor and a high-end DSLR camera demonstrated that the proposed solution was able to get to the level of commercial ISP of the Huawei P20 cameraphone, while these results were later improved in [30, 7, 45, 35, 26]. In this challenge, we take one step further in solving this problem by using more advanced data and by putting additional efficiency-related constraints on the developed solutions.

When it comes to the deployment of AI-based solutions on mobile devices, one needs to take care of the particu-
larities of mobile NPUs and DSPs to design an efficient model. An extensive overview of smartphone AI acceleration hardware and its performance is provided in [27, 24]. According to the results reported in these papers, the latest mobile NPUs are already approaching the results of mid-range desktop GPUs released not long ago. However, there are still two major issues that prevent a straightforward deployment of neural networks on mobile devices: a restricted amount of RAM, and limited and not always efficient support for many common deep learning layers and operators. These two problems make it impossible to process high-resolution data with standard NN models, thus requiring a careful adaptation of each architecture to the restrictions of mobile AI hardware. Such optimizations can include network pruning and compression [6, 20, 37, 39, 43], 16-bit / 8-bit [6, 34, 33, 55] and low-bit [5, 50, 32, 40] quantization, device- or NPU-specific adaptations, platform-aware neural architecture search [11, 46, 54, 52], etc.

While many challenges and works targeted at efficient deep learning models have been proposed recently, the evaluation of the obtained solutions is generally performed on desktop CPUs and GPUs, making the developed solutions not practical due to the above-mentioned issues. To address this problem, we introduce the first Mobile AI Workshop and Challenges, where all deep learning solutions are developed for and evaluated on real mobile devices. In this competition, the participating teams were provided with a new ISP dataset consisting of RAW-RGB image pairs captured with the Sony IMX586 mobile sensor and a professional 102-megapixel Fujifilm camera, and were developing an end-to-end deep learning solution for the learned ISP task. Within the challenge, the participants were evaluating the runtime and tuning their models on the MediaTek Dimensity 1000+ platform featuring a dedicated AI Processing Unit (APU) that can accelerate floating-point and quantized neural networks. The final score of each submitted solution was based on the image reconstruction quality and efficiency of the proposed model. Finally, all developed solutions are fully compatible with the TensorFlow Lite framework [47], thus can be deployed and accelerated on any mobile platform providing AI acceleration through the Android Neural Networks API (NNAPI) [1] or custom TFLite delegates [9].

This challenge is a part of the MAI 2021 Workshop and Challenges consisting of the following competitions:

- Learned Smartphone ISP on Mobile NPUs
- Real Image Denoising on Mobile GPUs [15]
- Quantized Image Super-Resolution on Mobile NPUs [25]
- Real-Time Video Super-Resolution on Mobile GPUs [22]
- Single-Image Depth Estimation on Mobile Devices [18]
- Quantized Camera Scene Detection on Smartphones [19]
- High Dynamic Range Image Processing on Mobile NPUs

The results obtained in the other competitions and the description of the proposed solutions can be found in the corresponding challenge report papers.

2. Challenge

To develop an efficient and practical solution for mobile-related tasks, one needs the following major components:

1. A high-quality and large-scale dataset that can be used to train and evaluate the solution on real (not synthetically generated) data;
2. An efficient way to check the runtime and debug the model locally without any constraints;
3. An ability to regularly test the runtime of the designed neural network on the target mobile platform or device.

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

To handle the problem of image translation from the original RAW photos captured with modern mobile camera sensors to superior quality images achieved by professional full-frame or medium format cameras, a large-scale real-world dataset containing RAW-RGB image pairs was collected. The dataset consists of photos taken in the wild synchronously by a 102-MP Fujifilm medium format camera and the Sony IMX586 Quad Bayer mobile sensor shooting RAW images. The photos were taken during the daytime in a wide variety of places and various illumination and weather conditions. The photos were captured in automatic mode, and the default settings were used for both cameras throughout the whole collection procedure. An example set of collected images can be seen in Fig. 1.

Since the captured RAW-RGB image pairs are not perfectly aligned, they were matched using an advanced dense correspondence algorithm [49], and then smaller patches of size 256×256 px were extracted. The participants were provided with around 24 thousand training RAW-RGB image pairs (of size 256×256×1 and 256×256×3, respectively). It should be mentioned that all alignment operations were performed on RGB Fujifilm images only, therefore RAW photos from the Sony sensor remained unmodified. A comprehensive tutorial demonstrating how to work with the data and how to train a baseline PUNET model on the provided images was additionally released to the participants: [https://github.com/MediaTek-NeuroPilot/mal21-learned-smartphone-isp](https://github.com/MediaTek-NeuroPilot/mal21-learned-smartphone-isp).

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 [24, 27] 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, Samsung Eden and MediaTek Neuron 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 or from the Google Play 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](https://ai-benchmark.com/download)
### 2.3. Runtime Evaluation on the Target Platform

In this challenge, we use the MediaTek Dimensity 1000+ SoC as our target runtime evaluation platform. This chipset contains a powerful APU [36] capable of accelerating floating point, INT16 and INT8 models, being ranked first by AI Benchmark at the time of its release [2]. It should be mentioned that FP16/INT16 inference support is essential for this task as raw Bayer data has a high dynamic range (10-to 14-bit images depending on the camera sensor model).

Within the challenge, the participants were able to upload their TFLite models to the runtime validation server connected to a real device and get instantaneous feedback: the runtime of their solution on the Dimensity 1000+ APU or a detailed error log if the model contains some incompatible operations (ops). The models were parsed and accelerated using MediaTek Neuron delegate\(^3\). The same setup was also used for the final runtime evaluation. The participants were additionally provided with a detailed model optimization guideline demonstrating the restrictions and the most efficient setups for each supported TFLite op.

### 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, to get the runtime on the target platform, 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,
- Structural Similarity Index Measure (SSIM), a proxy for perceptual score,
- The runtime on the target Dimensity 1000+ platform.

The score of each final submission was evaluated based on the next formula:

\[
\text{Final Score} = \text{PSNR} + \alpha \cdot (0.2 - \text{clip}(\text{runtime})),
\]

where:

\[
\alpha = \begin{cases} 
20, & \text{if runtime} \leq 0.2 \\
0.5, & \text{otherwise}
\end{cases},
\]

\[
\text{clip} = \min(\max(\text{runtime}, 0.03), 5).
\]

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 190 registered participants, 9 teams entered the final phase and submitted valid results, TFLite models, codes, executables, and factsheets. Table 1 summarizes the final challenge results and reports PSNR, SSIM, and runtime numbers for each submitted solution on the final test dataset on the target evaluation platform. The proposed methods are described in Section 4, and the team members and affiliations are listed in Appendix A.

<table>
<thead>
<tr>
<th>Team</th>
<th>Author</th>
<th>Framework</th>
<th>Model Size, KB</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>Runtime, ms ↓</th>
<th>Final Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>dh_sp</td>
<td>xushusong001</td>
<td>PyTorch / TensorFlow</td>
<td>21</td>
<td>23.2</td>
<td>0.8467</td>
<td>61</td>
<td>25.98</td>
</tr>
<tr>
<td>AIISP</td>
<td>AIISP</td>
<td>TensorFlow</td>
<td>123</td>
<td>23.73</td>
<td>0.8487</td>
<td>90.8</td>
<td>25.91</td>
</tr>
<tr>
<td>Tuned U-Net</td>
<td>Baseline</td>
<td>PyTorch / TensorFlow</td>
<td>13313</td>
<td>23.30</td>
<td>0.8395</td>
<td>78</td>
<td>25.74</td>
</tr>
<tr>
<td>ENERZAI Research &amp; isp_forever</td>
<td>Minus.Kwon</td>
<td>TensorFlow</td>
<td>9</td>
<td>22.97</td>
<td>0.8392</td>
<td>65</td>
<td>25.67</td>
</tr>
<tr>
<td>NOAHTCV</td>
<td>noahcv</td>
<td>TensorFlow</td>
<td>175</td>
<td>22.78</td>
<td>0.8472</td>
<td>77</td>
<td>25.24</td>
</tr>
<tr>
<td>CVML</td>
<td>jesse1029</td>
<td>TensorFlow</td>
<td>244</td>
<td>23.08</td>
<td>0.8237</td>
<td>94.5</td>
<td>25.19</td>
</tr>
<tr>
<td>ENS Research *</td>
<td>jaryon</td>
<td>PyTorch / TensorFlow</td>
<td>7</td>
<td>22.03</td>
<td>0.8217</td>
<td>76.3</td>
<td>24.5</td>
</tr>
<tr>
<td>EdS</td>
<td>Etienne</td>
<td>TensorFlow</td>
<td>1017</td>
<td>23.41</td>
<td>0.8534</td>
<td>231</td>
<td>23.39</td>
</tr>
</tbody>
</table>

\(^3\)https://github.com/MediaTek-NeuroPilot/tflite-neuron-delegate

---

Table 1. Mobile AI 2021 smartphone ISP challenge results and final rankings. The runtime values were obtained on Full HD (1920×1088) images. Teams dh_sp and AIISP are the challenge winners. Tuned U-Net corresponds to a baseline U-Net model [44] tuned specifically for the target Dimensity 1000+ platform. Team EdS was ranked second in the PIRM 2018 Image Enhancement Challenge [29], its results on this task are provided for the reference. * The second solution from ENERZAI Research team did not participate in the official test phase, its scores are shown for general information only.
3.1. Results and Discussion

All submitted solutions demonstrated a very high efficiency: the majority of models are able to process one Full HD (1920×1088 px) image under 100 ms on the target MediaTek APU. Teams dh isp and AIISP are the winners of this challenge, achieving the best runtime and fidelity results on this task, respectively. These solutions are following two absolutely different approaches. dh isp is using an extremely shallow 3-layer FSRCNN [10]-inspired model with one pixel-shuffle block, and is processing the input image at the original scale. The size of this model is only 21 KB, and it is able to achieve an impressive 16 FPS on the Dimensity 1000+ SoC. In contrast, the solution proposed by AIISP is downsampling the input data and applying a number of convolutional layers and several sophisticated attention blocks to get high fidelity results, while also demonstrating more than 10 FPS on the target platform.

Tuned U-Net is a solid U-Net baseline with several hardware-driven adaptations designed to demonstrate the performance that can be achieved with a common APU-aware tuned deep learning architecture. It is also showing that the weight and the number of layers do not necessarily play key roles in model efficiency if the majority of processing is happening at lower scales. While its size is more than 1000 times larger compared to the model proposed by ENERZAi Research, it demonstrates comparable runtime results on the same hardware. The tremendous model size reduction in the latter case was achieved by using an efficient knowledge transfer approach consisting of the joint training of two (tiny and large) models sharing the same feature extraction block. Another interesting approach was proposed by NOAHTCV which model is processing chroma and texture information separately.

In the final ranking table, one can also find the results of team EdS that was ranked second in the PIRM 2018 Image Enhancement Challenge [29]. This model was deliberately not optimized for the target platform to demonstrate the importance of such fine-tuning. While its fidelity scores are still high, showing the second-best PSNR result, it requires almost 2 seconds to process one image on the Dimensity 1000+ platform. The reason for this is quite straightforward: it is using several ops not yet adequately supported by NNAPI (despite claimed as officially supported). By removing or replacing these ops, the runtime of this model improves by more than 10 times, while the corresponding difference on desktop CPUs / GPUs is less than 10%. This example explicitly shows that the runtime values obtained on common deep learning hardware are not representative when it comes to model deployment on mobile AI silicon: even solutions that might seem to be very efficient can struggle significantly due to the specific constraints of smartphone AI acceleration hardware and frameworks. This makes deep learning development for mobile devices so challenging, though the results obtained in this competition demonstrate that one can get a very efficient model when taking the above aspects into account.

4. Challenge Methods

This section describes solutions submitted by all teams participating in the final stage of the MAI 2021 Learned Smartphone ISP challenge.

4.1. dh isp

Team dh isp proposed a very compact Smallnet architecture illustrated in Fig. 3 that consists of three convolutional and one pixel-shuffle layer. Each convolutional layer is using 3×3 kernels and has 16, 16, and 12 channels, respectively. Tanh activation function is used after the first layer, while the other ones are followed by the ReLU function. The authors especially emphasize the role of the Tanh and pixel-shuffle ops in getting high fidelity results on this task.

The model was first trained with L1 loss only, and then fine-tuned with a combination of L1 and perceptual-based VGG-19 loss functions. The network parameters were optimized with the Adam algorithm using a batch size of 4 and a learning rate of 1e−4 that was decreased within the training.

4.2. AIISP

The authors proposed a Channel Spacial Attention Network (Fig. 4) that achieved the best fidelity results in this challenge. The architecture of this model consists of the following three main parts. In the first part, two convolutional blocks with ReLU activations are used to perform feature extraction and downsize the input RAW data. After that, a series of processing blocks are cascaded. The middle double attention modules (DAM) with skip connections are mainly designed to enhance the spatial dependencies and to highlight the prominent objects in the feature maps. These skip connections are used not only to avoid the vanishing gradient problem but also to keep the similarities between the learned feature maps from different blocks. The last part of the network uses transposed convolution and depth-to-space modules to upscale the feature maps to their target size. Finally, a conventional convolution followed by the sigmoid activation function restores the output RGB image.
The sub-network structure of DAM is shown in Fig. 5. Given the feature maps obtained after applying two convolutions, DAM performs feature recalibration by using two attention mechanisms: spatial attention (SA) and channel attention (CA). The results of these concatenated attentions are then followed by a $1 \times 1$ convolutional layer to yield an adaptive feature refinement. The spatial attention module is designed to learn spatial dependencies in the feature maps. In order to have a distant vision over these maps, a depth-wise dilated convolution with a $5 \times 5$ kernel is used to extract the information. The output of this module is multiplied with the corresponding input feature map to get the final result. Channel attention block uses squeeze-and-excite operations to learn the inter-channel relationship between the feature maps. The squeeze operation is implemented by computing the mean values over individual feature maps. The excite operation is composed of two $1 \times 1$ convolution layers with different channel sizes and activations ($ReLU$ and sigmoid, respectively) and re-calibrates the squeeze output. The output of the module is also obtained by elemental-wise multiplication of the input feature maps and the calibrated descriptor. A more detailed description of the CSANet architecture is provided in [12].

The model is trained with a combination of the Charbonnier loss function (used to approximate $L_1$ loss), perceptual VGG-19 and SSIM losses. The weights of the model are optimized using Adam for 100K iterations with a learning rate of $5e^{-4}$ decreased to $1e^{-5}$ throughout the training. A batch size of 100 was used, the training data was augmented by random horizontal flipping.

4.3. Tuned U-Net Baseline

A U-Net [44] based model was developed to get an effective baseline for this challenge. This model follows the standard U-Net architecture with skip connections, and introduces several hardware-specific adaptations for the target platform such as a reduced number of feature maps, modified convolutional filter sizes, and activation functions, and additional skip connections used to maintain a reasonable accuracy. The model was trained with a combination of $MSE$ and $SSIM$ loss functions using Adam optimizer with a learning rate of $1e^{-4}$.

4.4. ENERGYAi Research

The solution proposed by ENERGYAi Research is inspired by the Once-for-All approach [3] and consists of two models: one super-network and one sub-network. They both share the same Dense-Net-like module, and the difference comes from their top layers: the sub-network has one deconvolution, convolution, and sigmoid layers, while the super-network additionally contains several residual dense blocks as shown in Fig. 6. Both models are first trained jointly using a combination of the Charbonnier and MS-SSIM loss functions. The super-network is then detached after the PSNR score goes above a predefined threshold, and the sub-net is further fine-tuned alone. The model was trained using Adam optimizer with a batch size of 4 and a
learning rate of $1e^{-3}$.

The second model proposed by this team (which did not officially participate in the final test phase) is demonstrated in Fig. 7. It follows the ESRGAN architecture [53] and has a shallow feature extractor and several DenseNet-based residual blocks with separable convolutions followed by a transpose convolution layer. The authors also used an additional channel attention block to boost the fidelity results at the expense of a very slight speed degradation. To choose the most appropriate activation function, the authors applied NAS technique that resulted in selecting the PReLU activations. The model was trained with a combination of the MS-SSIM and $L_1$ losses. It should be also mentioned that the original model was implemented and trained using PyTorch. To avoid the problems related to inefficient PyTorch-to-TensorFlow conversion, the authors developed their own scripts translating the original model architecture and weights to TensorFlow, and then converted the obtained network to TFLite.

![Figure 8. U-Net based network with a channel attention module from isp_forever.](image)

**4.5. isp_forever**

Team isp_forever proposed another truncated U-Net based model for this task that is demonstrated in Fig. 8. The authors augmented their network with a channel attention module and trained the entire model with a combination of $L_1$, SSIM, and VGG-based losses using Adam optimizer with a learning rate of $1e - 4$ and a batch size of 16.

**4.6. NOAHTCV**

This team proposed to decompose the input image into two parts (Fig. 9): chroma part that contains color information, and texture part that includes high-frequency details. The network processes these two parts in separate paths: the first one has a U-Net like architecture and performs patch-level information extraction and tone-mapping, while the second one applies residual blocks for texture enhancement. The outputs from both paths are fused at the end and then upsampled to get the final result. $L_1$ and SSIM losses were used to train the network, its parameters were initialized with Xavier and optimized using Adam with a learning rate of $1e - 4$.

**4.7. ACVLab**

ACVLab proposed a very compact CNN model with a local fusion block. This block consists of two parts: multiple stacked adaptive weight residual units, termed as RRDB
(residual in residual dense block), and a local residual fusion unit (LRFU). The RRDB module can improve the information flow and gradients, while the LRFU module can effectively fuse multi-level residual information in the local fusion block. The model was trained using VGG-based, SSIM, and Smooth $L_1$ losses.

4.8. CVML

![Diagram of the CVML architecture]

Figure 10. CNN architecture proposed by CVML team.

Figure 10 demonstrates the model proposed by team CVML. This architecture is using residual blocks to extract a rich set of features from the input data. Transposed convolution layer is used to upsample the final feature maps to the target resolution. To stabilize the training process, a global residual learning strategy was employed that also helped to reduce the color shift effect. The model was trained to minimize the combination of $L_1$ and SSIM losses and was optimized using Adam with a learning rate of $1e-4$ for 1M iterations.

4.9. EdS

![Diagram of the EdS architecture]

Figure 11. Residual network proposed by team EdS.

EdS proposed a ResNet-based architecture shown in Fig. 11 that was derived from [16]. The main difference consists in using two $4 \times 4$ convolutional layers with stride 2 for going into lower-dimensional space, and additional skip connections for faster training. The network was trained for 33K iterations using the same losses and setup as in [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: [26, 30]
- Perceptual Image Enhancement: [29, 23]
- Bokeh Effect Rendering: [21, 28]
- Image Super-Resolution: [29, 42, 4, 48]

Acknowledgements

We thank MediaTek Inc., AI Witchlabs and ETH Zurich (Computer Vision Lab), the organizers and sponsors of this Mobile AI 2021 challenge.

A. Teams and Affiliations

Mobile AI 2021 Team

**Title:** Mobile AI 2021 Learned Smartphone ISP Challenge

**Members:**
Andrey Ignatov$^{1,3}$ (andrey@vision.ee.ethz.ch), Cheng-Ming Chiang$^{2}$ (jimmy.chiang@mediatek.com), Hsien-Kai Kuo$^{2}$ (hsienkai.kuo@mediatek.com), Anastasia Sycheva$^{1}$, Radu Timofte$^{1,3}$ (radu.timofte@vision.ee.ethz.ch), Min-Hung Chen$^{2}$ (mh.chen@mediatek.com), Man-Yu Lee$^{2}$ (my.lee@mediatek.com), Yu-Syuan Xu$^{2}$ (yu-syuan.xu@mediatek.com), Yu Tseng$^{2}$ (ulia.tseng@mediatek.com)

**Affiliations:**
1. Computer Vision Lab, ETH Zurich, Switzerland
2. MediaTek Inc., Taiwan
3. AI Witchlabs, Switzerland

dh_isp

**Title:** Smallnet for An End-to-end ISP Pipeline

**Members:**
Shusong Xu (13821177832@163.com), Jin Guo

**Affiliations:**
Dahua Technology, China

AIISP

**Title:** CSAnet: High Speed Channel Spacial Attention Network for Mobile ISP [12]

**Members:**
Chao-Hung Chen (ZHChen@itri.org.tw), Ming-Chun
Hsyu, Wen-Chia Tsai, Chao-Wei Chen  
**Affiliations:**  
Industrial Technology Research Institute (ITRI), Taiwan

**U-Net Baseline**  
**Title:** Platform-aware Tuned U-Net Model  
**Members:**  
Grigory Malivenko (grigory.malivenko@gmail.com)  
**Affiliations:**  
Out of competition participation

**ENERZAi Research**  
**Title:** Learning DenseNet by Shrinking Large Network for ISP  
**Members:**  
Minsu Kwon (minsu.kwon@enerzai.com), Myungje Lee, Jaeyoon Yoo, Changbeom Kang, Shinjo Wang  
**Affiliations:**  
ENERZAi, Seoul, Korea  
enerzai.com

**isp_forever**  
**Title:** A Lightweight U-Net Model for Learned Smartphone ISP  
**Members:**  
Zheng Shaolong (master5158@163.com), Hao Dejun, Xie Fen, Feng Zhuang  
**Affiliations:**  
Dahua Technology, China

**NOAHTCV**  
**Title:** Decomposition Network for AutoISP  
**Members:**  
Yipeng Ma (mayipeng@huawei.com), Jingyang Peng, Tao Wang, Fenglong Song  
**Affiliations:**  
Huawei Noah’s Ark Lab, China

**ACVLab**  
**Title:** Lightweight Residual Learning Network for Learned ISP  
**Members:**  
Chih-Chung Hsu (cchsu@gs.ncku.edu.tw), Kwan-Lin Chen, Kwan-Lin Chen, Mei-Hsuang Wu  
**Affiliations:**  
Institute of Data Science, National Cheng Kung University, Taiwan

**CVML**  
**Title:** Mobile compatible Convolution Neural Network for Learned Smartphone ISP  
**Members:**  
Vishal Chudasama (vishalmchudasama88@gmail.com), Kalpesh Prajapati, Heena Patel, Anjuli Sarvaiya, Kishor Upla, Kiran Raja, Raghavendra Ramachandra, Christoph Busch  
**Affiliations:**  
Sardar Vallabhbhai National Institute of Technology, India

**EdS**  
**Title:** Fast learned ISP using Resnet Variant with Transposed Convolutions  
**Members:**  
Etienne de Stoutz (edestoutz@gmail.com)  
**Affiliations:**  
ETH Zurich, Switzerland

**References**  


[30] Andrey Ignatov, Radu Timofte, Zhilu Zhang, Ming Liu, Haolin Wang, Wangmeng Zuo, Jiawei Zhang, Ruimao


