Fast and Accurate Single-Image Depth Estimation on Mobile Devices, Mobile AI 2021 Challenge: Report

Andrey Ignatov Grigory Malivenko David Plowman Samarth Shukla Radu Timofte
Ziyu Zhang Yicheng Wang Zilong Huang Guozhong Luo Gang Yu Bin Fu
Yiran Wang Xingyi Li Min Shi Ke Xian Zhiguo Cao Jin-Hua Du
Pei-Lin Wu Chao Ge Jiaoyang Yao Fangwen Tu Bo Li Jung Eun Yoo
Kwanggyoon Seo Jialei Xu Zhenyu Li Xianming Liu Junjun Jiang
Wei-Chi Chen Shayan Joya Huanhuan Fan Zhaobing Kang Ang Li
Tianpeng Feng Yang Liu Chuannan Sheng Jian Yin Fausto T. Benavides

Abstract

Depth estimation is an important computer vision problem with many practical applications to mobile devices. While many solutions have been proposed for this task, they are usually very computationally expensive and thus are not applicable for on-device inference. To address this problem, we introduce the first Mobile AI challenge, where the target is to develop an end-to-end deep learning-based depth estimation solutions that can demonstrate a nearly real-time performance on smartphones and IoT platforms. For this, the participants were provided with a new large-scale dataset containing RGB-depth image pairs obtained with a dedicated stereo ZED camera producing high-resolution depth maps for objects located at up to 50 meters. The runtime of all models was evaluated on the popular Raspberry Pi 4 platform with a mobile ARM-based Broadcom chipset. The proposed solutions can generate VGA resolution depth maps at up to 10 FPS on the Raspberry Pi 4 while achieving high fidelity results, and are compatible with any Android or Linux-based mobile devices. A detailed description of all models developed in the challenge is provided in this paper.

1. Introduction

A wide spread of various depth-guided problems related to augmented reality, gesture recognition, object segmentation, autonomous driving and bokeh effect rendering tasks has created a strong demand for fast and efficient single-image depth estimation approaches that can run on portable low-power hardware. While many accurate deep learning-based solutions have been proposed for this problem in the past [46, 16, 14, 47, 48, 42, 15, 10], they were optimized for high fidelity results only while not taking into account computational efficiency and mobile-related constraints, which is essential for tasks related to image processing [23, 24, 37] on mobile devices. This results in solutions requiring powerful high-end GPUs and consuming gigabytes of RAM when processing even low-resolution input data, thus being incompatible with resource-constrained mobile hardware. In this challenge, we change the current depth estimation benchmarking paradigm by using a new depth estimation dataset collected in the wild and by imposing additional efficiency-related constraints on the designed solutions.

When it comes to the deployment of AI-based solutions on portable devices, one needs to take care of the particularities of mobile CPUs, NPUs and GPUs to design an efficient model. An extensive overview of mobile AI acceleration hardware and its performance is provided in [33, 30]. 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 a 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 [11, 26, 45, 49, 53], 16-bit / 8-bit [11, 40, 39, 73] and low-bit [9, 65, 38, 50] quantization, device- or NPU-specific adaptations, platform-aware neural architecture search [20, 60, 70, 66], *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

* Andrey Ignatov, Grigory Malivenko, David Plowman and Radu Timofte are the Mobile AI 2021 challenge organizers *(andrey@vision.ee.ethz.ch, grigory.malivenko@gmail.com, david.plowman@raspberrypi.com, radu.timofte@vision.ee.ethz.ch)*. The other authors participated in the challenge. Appendix A contains the authors' team names and affiliations.

Mobile AI 2021 Workshop website:
https://ai-benchmark.com/workshops/mai/2021/

Figure 1. The original RGB image and the corresponding depth map obtained with the ZED 3D camera.

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 low-power devices. In this competition, the participating teams were provided with a novel depth estimation dataset containing over 8 thousand RGB-depth image pairs collected in the wild with a stereo ZED 3D camera. Within the challenge, the participants were evaluating the runtime and tuning their models on the Raspberry Pi 4 ARM based single-board computer used as a target platform for many embedded machine learning projects. The final score of each submitted solution was based on the runtime and fidelity results, thus balancing between the image reconstruction quality and efficiency of the proposed model. Finally, all developed solutions are fully compatible with the TensorFlow Lite framework [62], thus can be deployed and accelerated on any mobile platform providing AI acceleration through the Android Neural Networks API (NNAPI) [5] or custom TFLite delegates [12].

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

- Learned Smartphone ISP on Mobile NPUs [22]
- Real Image Denoising on Mobile GPUs [21]
- Quantized Image Super-Resolution on Edge SoC NPUs [31]
- Real-Time Video Super-Resolution on Mobile GPUs [28]
- Single-Image Depth Estimation on Mobile Devices
- Quantized Camera Scene Detection on Smartphones [25]
- 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 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;
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 get real and diverse data for the considered challenge, a novel dataset consisting of RGB-depth image pairs was collected using the ZED stereo camera[1] capable of shooting 2K images. It demonstrates an average depth estimation error of less than 0.2m for objects located closer than 8 meters [55], while more coarse predictions are also available for distances of up to 50 meters. Around 8.3K image pairs were collected in the wild over several weeks in a variety of places. For this challenge, the obtained images were downscaled to VGA resolution (640×480 pixels) that is typically used on mobile devices for different depth-related tasks. The original RGB images were then considered as inputs, and the corresponding 16-bit depth maps — as targets. A sample RGB-depth image pair from the collected dataset is demonstrated in Fig. 1.

[1] https://www.stereolabs.com/zed/

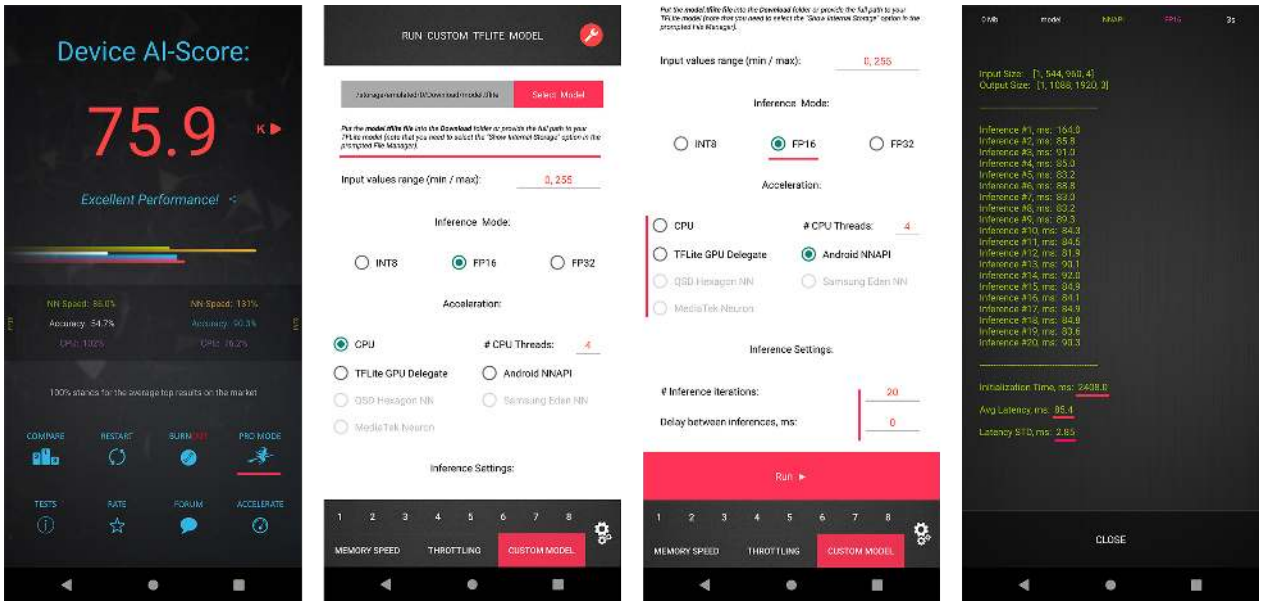

Figure 2. Loading and running custom TensorFlow Lite models with AI Benchmark application. The currently supported acceleration options include Android NNAPI, TFLite GPU, Hexagon NN, Samsung Eden and MediaTek Neuron delegates as well as CPU inference through TFLite or XNNPACK backends. The latest app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

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, 33] 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[2] or from the Google Play[3] 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.

[2] https://ai-benchmark.com/download

[3] https://play.google.com/store/apps/details?id=org.benchmark.demo

2.3. Runtime Evaluation on the Target Platform

In this challenge, we use the *Raspberry Pi 4* single-board computer as our target runtime evaluation platform. It is based on the *Broadcom BCM2711* chipset containing four Cortex-A72 ARM cores clocked at 1.5 GHz and demonstrates AI Benchmark scores comparable to entry-level Android smartphone SoCs [6]. The Raspberry Pi 4 supports the majority of Linux distributions, Windows 10 IoT build as well as Android operating system. In this competition, the runtime of all solutions was tested using the official TensorFlow Lite 2.5.0 Linux build [63] containing many important performance optimizations for the above chipset, the default *Raspberry Pi OS* was installed on the device. Within the challenge, the participants were able to upload their TFLite models to the runtime validation server connected to a real Raspberry Pi 4 board and get instantaneous feedback: the runtime of their solution or an error log if the model contains some incompatible operations. The same setup was also used for the final runtime evaluation.

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 plat-

Team	Author	Framework	Model Size, MB	si-RMSE↓	RMSE↓	LOG10↓	REL↓	Runtime, ms ↓	Final Score
Tencent GY-Lab	Parkzyzhang	PyTorch / TensorFlow	3.4	0.2836	3.56	0.1121	0.2690	**97**	**129.41**
SMART	KX_SMART	PyTorch / TensorFlow	15.0	0.2602	3.25	0.1043	0.2678	1197	14.51
Airia-Team1	dujinhua	TensorFlow	64.9	0.2408	3.00	0.0904	0.2389	1933	11.75
YTL	Jacob.Yao	PyTorch / TensorFlow	56.2	0.2902	3.91	0.1551	0.4700	1275	8.98
CFL2	jey	PyTorch / TensorFlow	9.6	0.2761	9.68	2.3393	0.9951	772	5.5
HIT-AIIA	zhyl	Keras / TensorFlow	56.0	**0.2332**	**2.72**	**0.0831**	**0.2189**	6146	4.11
weichi	weichi	TensorFlow	0.5	0.4659	7.56	0.4493	0.5992	582	1.72
MonoVision Palace	shayanj	TensorFlow	15.3	0.3543	4.16	0.1441	0.3862	3466	1.36
3dv oppo	fanhuanhuan	PyTorch / TensorFlow	187	0.2678	5.96	0.3300	0.5152	26494	0.59
MegaUe	faustChok	Keras / TensorFlow	118	0.3737	9.08	0.9605	0.8573	9392	0.38

Table 1. MAI 2021 Monocular Depth Estimation challenge results and final rankings. The runtime values were obtained on 640×480 px images on the Raspberry Pi 4 device. Team *Tencent GY-Lab* is the challenge winner, the best fidelity results are obtained by team *HIT-AIIA*.

form, 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:

- Root Mean Squared Error (RMSE) measuring the absolute depth estimation accuracy,
- Scale Invariant Root Mean Squared Error (si-RMSE) measuring the quality of relative depth estimation (relative position of the objects),
- Average $\log_{10}$ and Relative (REL) errors [48],
- The runtime on the target Raspberry Pi 4 device.

The score of each final submission was evaluated based on the next formula (C is a constant normalization factor):

$$\text{Final Score} = \frac{2^{-20 \cdot \text{si-RMSE}}}{C \cdot \text{runtime}},$$

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 above 140 registered participants, 10 teams entered the final phase and submitted valid results, TFLite models, codes, executables and factsheets. Table 1 summarizes the final challenge results and reports si-RMSE, RMSE, LOG10 and REL measures and runtime numbers for each submitted solution on the final test dataset and on the target evaluation platform. The proposed methods are described in section 4, and the team members and affiliations are listed in Appendix A.

3.1. Results and Discussion

All proposed solutions are relying on the encoder-decoder based architecture as it allows both to perform heavy image manipulations and to reduce the computational complexity of the model by doing the majority of processing at lower scales / resolutions. Nearly all models used standard image classification models in their encoder module extracting features from the input images. Teams *Tencent GY-Lab*, *SMART*, *Airia-Team1* and *CFL2* adopted MobileNets for this as they are already optimized for low-power devices and can achieve a very good runtime on the majority of mobile platforms. The best fidelity results were, however, obtained by team *HIT-AIIA* that used the EfficientNet-B1 network for feature generation. To improve the models' accuracy, skip connections between the encoder and decoder blocks were added in almost all architectures. Another popular approach resulting in better depth prediction was to use knowledge distillation: a larger model was first trained for the same task, and then its outputs or intermediate features were used as additional targets for the final small network. In particular, this approach was used by the challenge winner, team *Tencent GY-Lab*, that outperformed all other methods by a huge margin, being able to get both good fidelity scores and to achieve more than 10 FPS on the target Raspberry Pi 4 device. Notably, this solution is a magnitude faster than the FastDepth [69] model known as one of the most efficient ones for this task.

To further benchmark the efficiency of the designed solutions, we additionally tested their performance on several popular smartphone chipsets. The runtime results demonstrated in Table 2 were measured with the AI Benchmark using the TFLite GPU delegate [43] compatible with all mobile devices supporting OpenCL or OpenGL 3.0+. In almost all cases, the runtime of the proposed networks is less than half a second except for the solution from *3dv oppo*: due to the issues caused by PyTorch to TFLite conversion, it contains several ops supported neither by TFLite delegates nor by Android NNAPI, thus this model was executed on CPU, same as networks from *Airia-Team1* and *CFL2*. The solution from team *Tencent GY-Lab* demonstrated more than 75 FPS on all considered SoCs, thus be-

Mobile SoC GPU	Snapdragon 888 Adreno 660, ms	Snapdragon 855 Adreno 640, ms	Dimensity 1000 Mali-G77 MP9, ms	Dimensity 800 Mali-G57 MP4, ms	Exynos 2100 Mali-G78 MP14, ms	Exynos 990 Mali-G77 MP11, ms	Kirin 990 5G Mali-G76 MP16, ms	Kirin 980 Mali-G76 MP10, ms
Tencent GY-Lab	3.5	5.7	8.6	13	5.7	12	8.8	9.3
SMART	33	60	65	106	37	53	48	58
Airia-Team1 *	283	321	295	447	248	270	337	351
YTL	35	70	71	104	36	52	54	65
CFL2 *	121	179	186	277	117	170	179	188
HIT-AIIA	95	175	149	320	101	137	142	183
weichi	7.1	11	23	43	13	18	18	22
MonoVision Palace	77	128	119	247	71	97	101	129
3dv oppo *	3672	4346	4053	4832	4071	3649	3753	4107
MegaUe	141	288	245	547	182	234	209	266

Table 2. The speed of the proposed solutions on several popular mobile GPUs. The runtime was measured with the AI Benchmark app using the TFLite GPU delegate [43]. * Solutions from teams *Airia-Team1*, *CFL2* and *3dv oppo* are not compatible with neither TFLite delegates nor Android NNAPI due to the issues related to PyTorch → TFLite conversion, thus were executed on mobile CPUs.

ing able to generate depth maps in real-time on all modern chipsets, including the low-end ones. We can conclude that this architecture is now defining a new efficiency standard for depth estimation on mobile and embedded systems. The model from team *HIT-AIIA*, demonstrating the best accuracy in this challenge, is able to achieve at least 7 FPS on all tested SoCs, thus being applicable for tasks where the precision of the predicted depth maps is critical. It should be also mentioned that all models were additionally tested on NPUs / DSPs of the considered chipsets, though the results were either the same or worse since not all TFLite layers and operations are currently optimized for specialized AI hardware.

4. Challenge Methods

This section describes solutions submitted by all teams participating in the final stage of the MAI 2021 Monocular Depth Estimation challenge.

4.1. Tencent GY-Lab

Team Tencent GY-Lab proposed a U-Net like architecture presented in Fig. 3, where a MobileNet-V3 [20] based encoder is used for dense feature extraction. To reduce the amount of computations, the input image is first resized from 640×480 to 160×128 pixels and then passed to the encoder module consisting of five blocks. The outputs of each block are processed by the Feature Fusion Module (FFM) that concatenates them with the decoder feature maps to get better fidelity results. The authors use one additional *nearest neighbor* resizing layer on top of the model to upscale the output to the target resolution. Knowledge distillation [19] is further used to improve the quality of the reconstructed depth maps: a bigger ViT-Large [13] was first trained on the same dataset and then its features obtained before the last activation function were used to guide the smaller network. This process allowed to decrease the si-RMSE score from 0.3304 to 0.3141. The proposed model was therefore trained to minimize a combination of the distillation loss (computed as L_2 norm between its features from the last convolutional layer and the above mentioned features from the larger model), and the depth estimation loss proposed in [44]. The network parameters were optimized for 500 epochs using Adam [41] with a learning rate of $8e-3$ and a polynomial decay with a power of 0.9. The model was implemented and trained with PyTorch and then converted to TensorFlow Lite using ONNX as an intermediate representation. A more detailed description of the proposed solution is provided in [74].

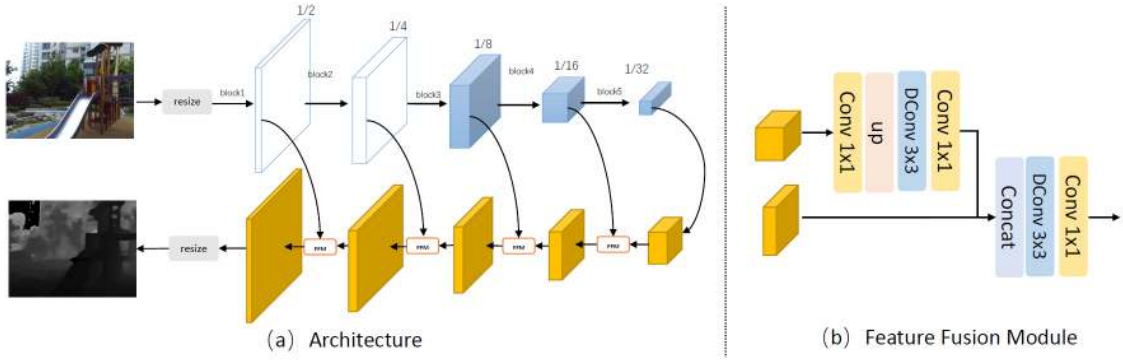

Figure 3. The model architecture and the structure of the Feature Fusion Module (FFM) proposed by team Tencent GY-Lab.

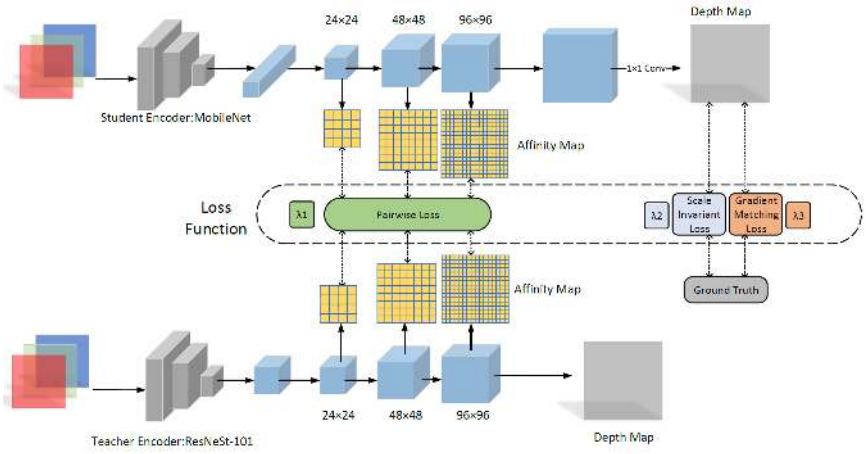

Figure 4. An overview of the knowledge distillation strategy used by team SMART.

4.2. SMART

Same as the previous solution, team SMART used a MobileNet-based encoder module for feature extraction and applied knowledge distillation to train the network. The architecture of the proposed solution is demonstrated in Fig. 5: the standard FastDepth [69] architecture with a MobileNet-V1 backbone is used for the main (student) model. The larger teacher network consists of a ResNeSt-101 [72] based encoder and a decoder block [71] with the adaptive output layer on top of it. The representation ability of a pre-trained teacher model is transferred to the student network via knowledge distillation: a pairwise distillation loss is adopted to force the student network to output feature maps that are similar to the outputs of the corresponding layers of the teacher network. The distillation loss is computed in two steps (Fig. 4): let $F_t \in \mathbb{R}^{h \times w \times c_1}$ and $F_s \in \mathbb{R}^{h \times w \times c_2}$ be the feature maps with the same spatial resolution from the teacher and the student models, respectively, then the affinity maps are first computed as:

$$a_{ij} = \frac{f_i^T f_j}{(\|f_i\|_2 \times \|f_j\|_2)},$$

where f denotes one row of the feature map (F_t or F_s). Next, the mean square error is computed between the affinity maps obtained for student and teacher models:

$$\mathcal{L}_{pa}(S,T) = \frac{1}{w \times h} \sum_i \sum_j (a_{ij}^s - a_{ij}^t)^2.$$

Besides that above knowledge distillation loss, two other metrics are used to train the student model. The scale invariant loss [14] is used to measure the discrepancy between the output of the student network and the ground truth depth map:

$$\mathcal{L}_s(d, d^*) = \frac{1}{n} \sum_i g_i^2 - \frac{1}{n^2} (\sum_i g_i)^2,$$

where d and d^* are the predicted and the ground truth depth maps, and $g_i = \log d_i - \log d_i^*$ is the corresponding error in log space. Finally, the scale-invariant gradient matching loss [57] is defined as:

$$\mathcal{L}_{reg}(d, d^*) = \frac{1}{M} \sum_{k=1}^{K} \sum_{i=1}^{M} (|\nabla_x R_i^k| + |\nabla_y R_i^k|),$$

where $R_i = d - d^*$, and R^k denotes the difference between the disparity maps at scale $k = 1, 2, 3, 4$ (the resolution of the feature maps is halved at each level). The final loss function is then defined as:

$$\mathcal{L} = 10 \cdot \mathcal{L}_s(d, d^*) + 0.1 \cdot \mathcal{L}_{reg}(d, d^*) + 1000 \cdot \mathcal{L}_{pa}(S, T).$$

The model was trained using Adam for 100 epochs with an initial learning rate of $1e{-}3$ and a polynomial decay with a power of 0.9. A more detailed description of the model, design choices and training procedure is provided in [68].

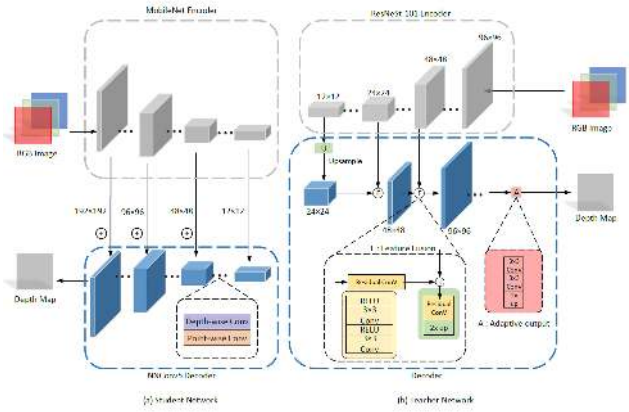

Figure 5. The architecture of the student and teacher models developed by team SMART.

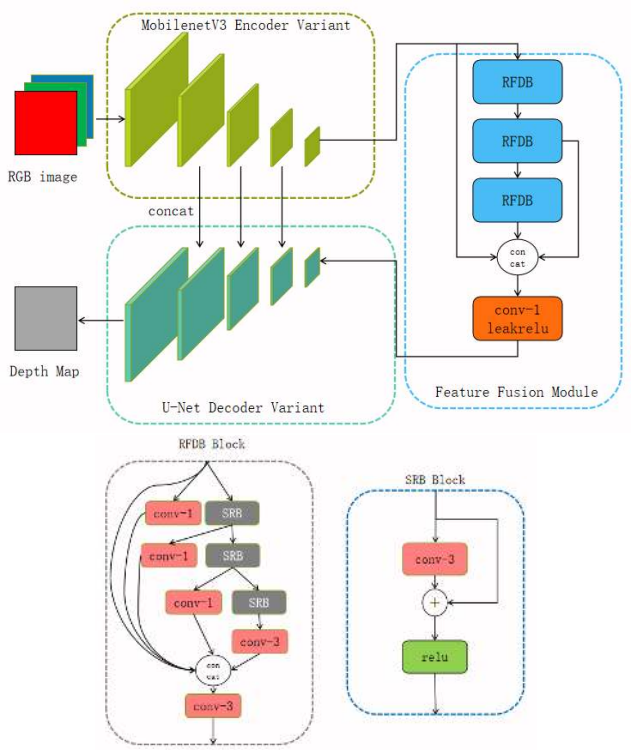

Figure 6. The architecture proposed by Airia-Team1 (top) and the structure of the RFDB block (bottom). *Conv-1* and *Conv-3* stands for 1×1 and 3×3 convolution, respectively.

4.3. Airia-Team1

Figure 6 demonstrates the architecture developed by Airia-Team1. The authors proposed an encoder-decoder model, where MobileNet-V3 [20] network is used for feature extraction, same as in the previous two solutions. The resulting features are fed to three residual feature distillation blocks (RFDB), each one composed of three residual blocks (SRB) and several convolutional and concatenation layers. The refined features obtained after these blocks are then passed to a 5-layer decoder producing the final predictions, several skip connections are additionally used to speed-up the training. The pixel-wise depth loss [7] was used as the target loss function. The model parameters were optimized using Adam with a learning rate of $1e-4$ multiplied by 0.6 each 100 epochs. A batch size of 8 was used during the training, random flips were additionally applied for data augmentation.

4.4. YTL

The authors proposed a U-net based architecture where the ResNet-18 [18] model is used for feature extraction. The input RGB image was resized to 320×240 resolution and then concatenated with an X/Y meshgrid (containing centered pixel coordinates) to form a 5-channel tensor passed to the model. The output of the model was also upsampled from 320×240 to the target 640×480 resolution using one *bilinear resize* layer on top of it. The network was trained to minimize a combination of the Mean Absolute Error (MAE) and gradient losses using Adam optimizer.

4.5. CFL2

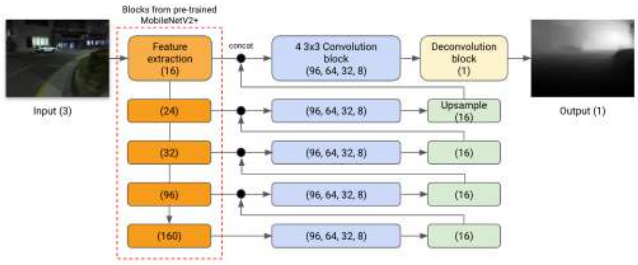

Figure 7. The PyDNet [3] architecture adopted by CFL2 team.

Team CFL2 based its solution on the PyDNet [56, 3] model. The input image was downscaled to 256×256 pixels and then passed to the MobileNetV2 [59] encoder. While the original PyDNet model produces several outputs at multiple scales, the authors used only the highest one that corresponds to the target resolution to reduce the computational complexity of the model. Since the PyDNet is originally producing 128×128px images, they were additionally upscaled to the target resolution using one *bilinear resize* layer. The scale invariant data loss [14] and the scale-invariant gradient matching loss [57] were used to train the model for 2M interations using Adam with a learning rate of $1e-4$.

4.6. HIT-AIIA

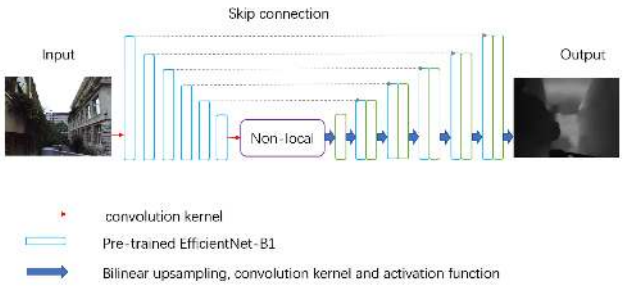

Figure 8. EfficientNet-based model proposed by team HIT-AIIA.

The model proposed by HIT-AIIA is using the EfficientNet-B1 network [61] as an encoder to extract features from the input images (Fig. 8). The outputs from its last layer are passed to the Non-Local block [67] that effectively improves the accuracy of the model. The authors used a combination of the bilinear upsampling, convolutional and *Leaky ReLU* layers in the decoder module predicting the final depth map. Additional skip connections were added to

speed-up the training process and improve the fidelity results. The model was trained to minimize RMSE loss function using Adam with a learning rate of $1e-4$ and a batch size of 6. Image mirroring and flipping as well as color alteration were used for data augmentation.

4.7. weichi

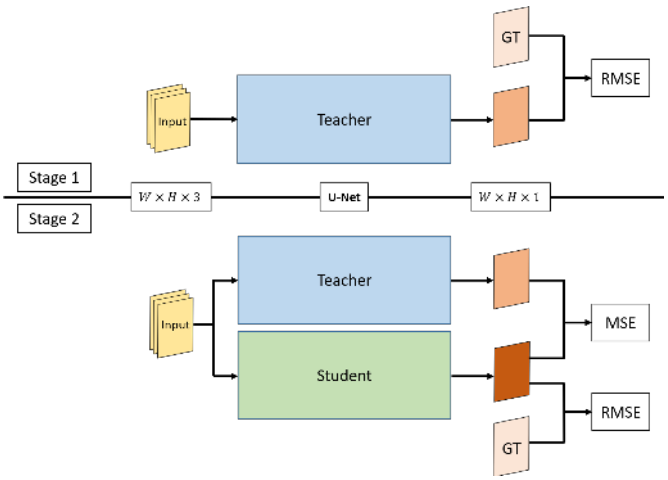

Figure 9. An overview of the knowledge distillation strategy used by team weichi.

Team weichi used the standard U-Net [58] architecture with a reduced by a factor of 8 number of feature maps in each layer. Same as in [52], the authors added batch normalization after each convolution in the encoder block. To improve the accuracy of the model, knowledge distillation [19] was additionally applied during the training process: a larger U-Net model (with an increased number of channels) was first trained on the same dataset using the RMSE loss function. Next, the main student network was minimizing a combination of the RMSE loss between its outputs and the target depth maps, and the MSE loss between its outputs and the outputs of the larger network (Fig. 9). Both models were trained using Adam optimizer with a learning rate of $5e-5$ and a batch size of 16.

4.8. MonoVision Palace

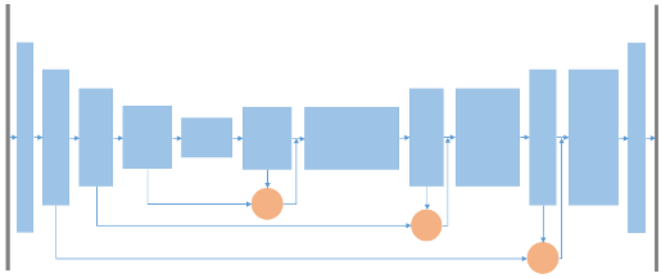

Figure 10. DA-UNet architecture proposed by MonoVision Palace.

Team MVP proposed a Depth Attention UNet (DA-UNet) architecture demonstrated in Fig. 10. The input image was first passed to the EfficientNet-Edge-TPU-S [17] model with removed *hard-swish* activations and *squeeze-and-excitation* blocks to reduce the latency. Its outputs were then processed by the decoder block composed of convolution, upsampling, *Leaky ReLU* and Gated Attention Blocks (GA) [54] where *ReLU* and *sigmoid* activations were replaced with *Leaky ReLUs* and *hard-sigmoid* ops, respectively. The model was trained using the same metrics as in [4]: the point-wise L_1 loss, the gradient L_1 loss, and the SSIM loss function. Adam was used to optimize the model parameters for 30 epochs with an initial learning rate of $1e-4$ reduced by a magnitude after the 20th and the 25th epoch.

4.9. 3dv oppo

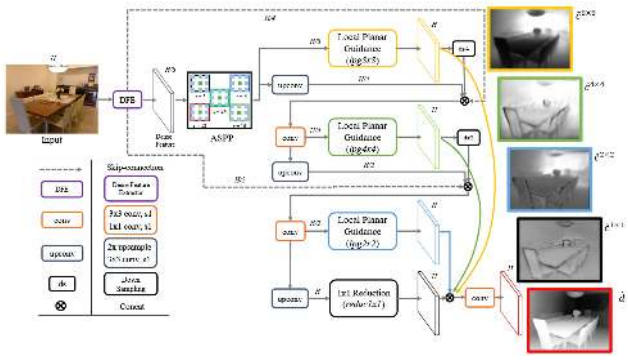

Figure 11. BTS network architecture adopted by 3dv oppo team.

The authors directly used the BTS model [44] demonstrated in Fig 11. This network is composed of the dense feature extractor (the ResNet model), the contextual information extractor (ASPP), the local planar guidance layers and their dense connection for final depth estimation. The same training setup and the target loss functions as in [44] was used except for the learning rate that was set to $5e-5$.

4.10. MegaUe

Team MegaUe trained a standard U-Net like architecture (Fig. 12) with one additional 2x image downsampling and upsampling layers at the begging and at the top of the model, respectively. The model was first pre-trained on the MegaDepth dataset [46] using the same metrics as in the original paper: the ordinal, data and gradient matching losses. Then, the model was fine-tuned on the challenge data using the last two loss functions.

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:

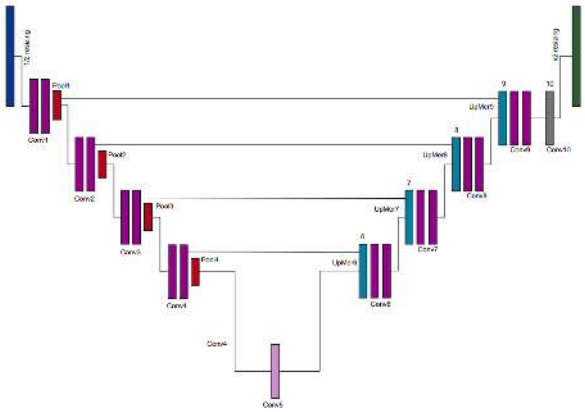

Figure 12. U-Net model proposed by MegaUe team.

- Learned End-to-End ISP: [32, 36]
- Perceptual Image Enhancement: [35, 29]
- Image Super-Resolution: [35, 51, 8, 64]
- Bokeh Effect Rendering: [27, 34]
- Image Denoising: [1, 2]

Acknowledgements

We thank Raspberry Pi (Trading) Ltd, 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 Challenge on Single-Image Depth Estimation on Mobile Devices
Members:
Andrey Ignatov[1,3] *(andrey@vision.ee.ethz.ch)*, Grigory Malivenko *(grigory.malivenko@gmail.com)*, David Plowman[2] *(david.plowman@raspberrypi.com)*, Samarth Shukla[1] *(samarth.shukla@vision.ee.ethz.ch)*, Radu Timofte[1,3] *(radu.timofte@vision.ee.ethz.ch)*
Affiliations:
[1] Computer Vision Lab, ETH Zurich, Switzerland
[2] Raspberry Pi (Trading) Ltd
[3] AI Witchlabs, Switzerland

Tencent GY-Lab

Title:
A Simple Baseline for Fast and Accurate Depth Estimation on Mobile Devices [74]
Members:
Ziyu Zhang (parkzyzhang@tencent.com), Yicheng Wang, Zilong Huang, Guozhong Luo, Gang Yu, Bin Fu
Affiliations:
Tencent GY-Lab, China

SMART

Title:
Knowledge Distillation for Fast and Accurate Monocular Depth Estimation on Mobile Devices [68]
Members:
Yiran Wang (wangyiran@hust.edu.cn), Xingyi Li, Min Shi, Ke Xian, Zhiguo Cao
Affiliations:
Key Laboratory of Image Processing and Intelligent Control, Ministry of Education, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, China

Airia-Team1

Title:
Monocular Depth Estimation based on MobileNetV3Small
Members:
Jin-Hua Du (2982192572@qq.com), Pei-Lin Wu, Chao Ge
Affiliations:
Nanjing Artificial Intelligence Chip Research, Institute of Automation, Chinese Academy of Sciences, China

YTL

Title:
U-Net with Pixel Position Encoding for Monocular Depth Estimation
Members:
Jiaoyang Yao (jiaoyangyao@gmail.com), Fangwen Tu, Bo Li
Affiliations:
Black Sesame Technologies Inc., Singapore

CFL2

Title:
Lightfast Depth Estimation
Members:
Jung Eun Yoo (jey920@kaist.ac.kr), Kwanggyoon Seo
Affiliations:
Visual Media Lab, KAIST, South Korea

HIT-AIIA

Title:
EfficientNet Encoder with Non-Local Module for Monocular Depth Estimation
Members:
Jialei Xu (20S003044@stu.hit.edu.cn), Zhenyu Li, Xianming Liu, Junjun Jiang
Affiliations:
Harbin Institute of Technology, China
Peng Cheng Laboratory, China

weichi

Title:
Distillation on UNet
Members:
Wei-Chi Chen (ne6094041@gs.ncku.edu.tw)
Affiliations:
Multimedia and Computer Vision Laboratory, National Cheng Kung University, Taiwan
http://mmcv.csie.ncku.edu.tw/

MVP - MonoVision Palace

Title:
DA-UNet: Depth Attention UNet for Monocular Depth Estimation
Members:
Shayan Joya (joya.shayan@gmail.com)
Affiliations:
Samsung Research UK, United Kingdom

3dv oppo

Title:
Accurate Monocular Depth Estimation Using BTS
Members:
Huanhuan Fan (fanhuanhuan@oppo.com), Zhaobing Kang, Ang Li, Tianpeng Feng, Yang Liu, Chuannan Sheng, Jian Yin
Affiliations:
OPPO Research Institute, China

MegaUe

Title:
Mega-Udepth for Monocular Depth Estimation
Members:
Fausto T. Benavides (fausto.tapiabenavides@gmail.com)
Affiliations:
ETH Zurich, Switzerland

References

[1] Abdelrahman Abdelhamed, Mahmoud Afifi, Radu Timofte, and Michael S Brown. Ntire 2020 challenge on real image denoising: Dataset, methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 496–497, 2020. 9

[2] Abdelrahman Abdelhamed, Radu Timofte, and Michael S Brown. Ntire 2019 challenge on real image denoising: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 9

[3] Filippo Aleotti, Giulio Zaccaroni, Luca Bartolomei, Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Real-time single image depth perception in the wild with handheld devices. *Sensors*, 21(1):15, 2021. 7

[4] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning. *arXiv preprint arXiv:1812.11941*, 2018. 8

[5] Android Neural Networks API. https://developer.android.com/ndk/guides/neuralnetworks. 2

[6] AI Benchmark Archive. http://web.archive.org/web/20210425131428/https://ai-benchmark.com/ranking_processors.html. 3

[7] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. *arXiv preprint arXiv:2011.14141*, 2020. 7

[8] Jianrui Cai, Shuhang Gu, Radu Timofte, and Lei Zhang. 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*, pages 0–0, 2019. 9

[9] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178, 2020. 1

[10] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *arXiv preprint arXiv:1604.03901*, 2016. 1

[11] Cheng-Ming Chiang, Yu Tseng, Yu-Syuan Xu, Hsien-Kai Kuo, Yi-Min Tsai, Guan-Yu Chen, Koan-Sin Tan, Wei-Ting Wang, Yu-Chieh Lin, Shou-Yao Roy Tseng, 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*, pages 502–503, 2020. 1

[12] TensorFlow Lite delegates. https://www.tensorflow.org/lite/performance/delegates. 2

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 5

[14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014. 1, 6, 7

[15] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European conference on computer vision*, pages 740–756. Springer, 2016. 1

[16] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019. 1

[17] Suyog Gupta and Mingxing Tan. Efficientnet-edgetpu: Creating accelerator-optimized neural networks with automl, 2019. 8

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 5, 8

[20] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019. 1, 5, 7

[21] Andrey Ignatov, Kim Byeoung-su, and Radu Timofte. 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*, pages 0–0, 2021. 2

[22] Andrey Ignatov, Jimmy Chiang, Hsien-Kai Kuo, Anastasia Sycheva, and Radu Timofte. 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*, pages 0–0, 2021. 2

[23] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. Dslr-quality photos on mobile devices with deep convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3277–3285, 2017. 1

[24] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. Wespe: weakly supervised photo enhancer for digital cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 691–700, 2018. 1

[25] Andrey Ignatov, Grigory Malivenko, and Radu Timofte. 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*, pages 0–0, 2021. 2

[26] Andrey Ignatov, Jagruti Patel, and Radu Timofte. Rendering natural camera bokeh effect with deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 418–419, 2020. 1

[27] Andrey Ignatov, Jagruti Patel, Radu Timofte, Bolun Zheng, Xin Ye, Li Huang, Xiang Tian, Saikat Dutta, Kuldeep Purohit, Praveen Kandula, et al. Aim 2019 challenge on bokeh effect synthesis: Methods and results. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3591–3598. IEEE, 2019. 9

[28] Andrey Ignatov, Andres Romero, Heewon Kim, and Radu Timofte. 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*, pages 0–0, 2021. 2

[29] Andrey Ignatov and Radu Timofte. Ntire 2019 challenge on image enhancement: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 9

[30] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. Ai benchmark: Running deep neural networks on android smartphones. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 1, 3

[31] Andrey Ignatov, Radu Timofte, Maurizio Denna, and Abdel Younes. 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*, pages 0–0, 2021. 2

[32] Andrey Ignatov, Radu Timofte, Sung-Jea Ko, Seung-Wook Kim, Kwang-Hyun Uhm, Seo-Won Ji, Sung-Jin Cho, Jun-Pyo Hong, Kangfu Mei, Juncheng Li, et al. Aim 2019 challenge on raw to rgb mapping: Methods and results. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3584–3590. IEEE, 2019. 9

[33] Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. Ai benchmark: All about deep learning on smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3617–3635. IEEE, 2019. 1, 3

[34] Andrey Ignatov, Radu Timofte, Ming Qian, Congyu Qiao, Jiamin Lin, Zhenyu Guo, Chenghua Li, Cong Leng, Jian Cheng, Juewen Peng, et al. Aim 2020 challenge on rendering realistic bokeh. In *European Conference on Computer Vision*, pages 213–228. Springer, 2020. 9

[35] Andrey Ignatov, Radu Timofte, Thang Van Vu, Tung Minh Luu, Trung X Pham, Cao Van Nguyen, Yongwoo Kim, Jae-Seok Choi, Munchurl Kim, Jie Huang, et al. Pirm challenge on perceptual image enhancement on smartphones: Report. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 9

[36] Andrey Ignatov, Radu Timofte, Zhilu Zhang, Ming Liu, Haolin Wang, Wangmeng Zuo, Jiawei Zhang, Ruimao Zhang, Zhanglin Peng, Sijie Ren, et al. Aim 2020 challenge on learned image signal processing pipeline. *arXiv preprint arXiv:2011.04994*, 2020. 9

[37] Andrey Ignatov, Luc Van Gool, and Radu Timofte. Replacing mobile camera isp with a single deep learning model. In *Proceedings of the IEEE/CVF Conference on Computer*

Vision and Pattern Recognition Workshops, pages 536–537, 2020. 1

[38] Dmitry Ignatov and Andrey Ignatov. Controlling information capacity of binary neural network. *Pattern Recognition Letters*, 138:276–281, 2020. 1

[39] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018. 1

[40] Sambhav R Jain, Albert Gural, Michael Wu, and Chris H Dick. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. *arXiv preprint arXiv:1903.08066*, 2019. 1

[41] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[42] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016. 1

[43] Juhyun Lee, Nikolay Chirkov, Ekaterina Ignasheva, Yury Pisarchyk, Mogan Shieh, Fabio Riccardi, Raman Sarokin, Andrei Kulik, and Matthias Grundmann. On-device neural net inference with mobile gpus. *arXiv preprint arXiv:1907.01989*, 2019. 4, 5

[44] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019. 5, 8

[45] Yawei Li, Shuhang Gu, Luc Van Gool, and Radu Timofte. Learning filter basis for convolutional neural network compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5623–5632, 2019. 1

[46] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2041–2050, 2018. 1, 8

[47] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5162–5170, 2015. 1

[48] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2015. 1, 4

[49] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3296–3305, 2019. 1

[50] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. 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)*, pages 722–737, 2018. 1

[51] Andreas Lugmayr, Martin Danelljan, and Radu Timofte. 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*, pages 494–495, 2020. 9

[52] Karttikeya Mangalam and Mathieu Salzamann. On compressing u-net using knowledge distillation. *arXiv preprint arXiv:1812.00249*, 2018. 8

[53] Anton Obukhov, Maxim Rakhuba, Stamatios Georgoulis, Menelaos Kanakis, Dengxin Dai, and Luc Van Gool. T-basis: a compact representation for neural networks. In *International Conference on Machine Learning*, pages 7392–7404. PMLR, 2020. 1

[54] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018. 8

[55] Luis Enrique Ortiz, Elizabeth V Cabrera, and Luiz M Gonçalves. Depth data error modeling of the zed 3d vision sensor from stereolabs. *ELCVIA: electronic letters on computer vision and image analysis*, 17(1):0001–15, 2018. 2

[56] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards real-time unsupervised monocular depth estimation on cpu. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5848–5854. IEEE, 2018. 7

[57] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019. 6, 7

[58] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 8

[59] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 7

[60] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 1

[61] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 7

[62] TensorFlow-Lite. https://www.tensorflow.org/lite. 2

[63] TensorFlow-Lite. https://www.tensorflow.org/lite/guide/python. 3

[64] Radu Timofte, Shuhang Gu, Jiqing Wu, and Luc Van Gool. Ntire 2018 challenge on single image super-resolution:

Methods and results. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 852–863, 2018. 9

[65] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Mixed precision dnns: All you need is a good parametrization. *arXiv preprint arXiv:1905.11452*, 2019. 1

[66] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, 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*, pages 12965–12974, 2020. 1

[67] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 7

[68] Yiran Wang, Xingyi Li, Min Shi, Ke Xian, and Zhiguo Cao. Knowledge distillation for fast and accurate monocular depth estimation on mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2021. 6, 9

[69] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth estimation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108. IEEE, 2019. 4, 6

[70] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019. 1

[71] Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li, and Zhenbo Luo. Monocular relative depth perception with web stereo data supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 311–320, 2018. 6

[72] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 6

[73] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7308–7316, 2019. 1

[74] Ziyu Zhang, Yicheng Wang, Zilong Huang, Guozhong Luo, Gang Yu, and Bin Fu. A simple baseline for fast and accurate depth estimation on mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2021. 5, 9