Fast and Accurate Quantized Camera Scene Detection on Smartphones, Mobile AI 2021 Challenge: Report

Andrey Ignatov   Grigory Malivenko   Radu Timofte   Sheng Chen   Xin Xia
Zhaoyan Liu   Yuwei Zhang   Feng Zhu   Jiashi Li   Xuefeng Xiao   Yuan Tian
Xinglong Wu   Christos Kyrkou   Yixin Chen   Zexin Zhang   Yunbo Peng   Yue Lin
Saikat Dutta   Sourya Dipta Das   Nisarg A. Shah   Himanshu Kumar   Chao Ge
Pei-Lin Wu   Jin-Hua Du   Andrew Batutin   Juan Pablo Federico   Konrad Lyda
Levon Khojoyan   Abhishek Thanki   Sayak Paul   Shahid Siddiqui

Abstract

Camera scene detection is among the most popular computer vision problem on smartphones. While many custom solutions were developed for this task by phone vendors, none of the designed models were available publicly up until now. To address this problem, we introduce the first Mobile AI challenge, where the target is to develop quantized deep learning-based camera scene classification solutions that can demonstrate a real-time performance on smartphones and IoT platforms. For this, the participants were provided with a large-scale CamSDD dataset consisting of more than 11K images belonging to the 30 most important scene categories. The runtime of all models was evaluated on the popular Apple Bionic A11 platform that can be found in many iOS devices. The proposed solutions are fully compatible with all major mobile AI accelerators and can demonstrate more than 100-200 FPS on the majority of recent smartphone platforms while achieving a top-3 accuracy of more than 98%. A detailed description of all models developed in the challenge is provided in this paper.

1. Introduction

The problem of automatic camera scene prediction on smartphones appeared soon after the introduction of the first mobile cameras. While the initial scene classification approaches were using only manually designed features and some simple machine learning algorithms, the availability of much more powerful AI hardware such as NPUs, GPUs and DSPs made it possible to use considerably more accurate and efficient deep learning-based solutions. Neverthe-

less, this task has not been properly addressed in the literature until the introduction of the Camera Scene Detection Dataset (CamSDD) dataset in [36], where this problem was carefully defined and training data for 30 different camera scene categories was provided along with a fast baseline solution. In this challenge, we take one step further in solving this task by imposing additional efficiency-related constraints on the developed models.

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 [21, 18]. 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 [6, 14, 30, 31, 35], 16-bit / 8-bit [6, 28, 27, 47] and low-bit [5, 42, 25, 32] quantization, device- or NPU-specific adaptations, platform-aware neural architecture search [9, 38, 44, 43], *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 low-power devices. In this competition, the participating teams were provided with a large-scale CamSDD dataset containing more than 11K images
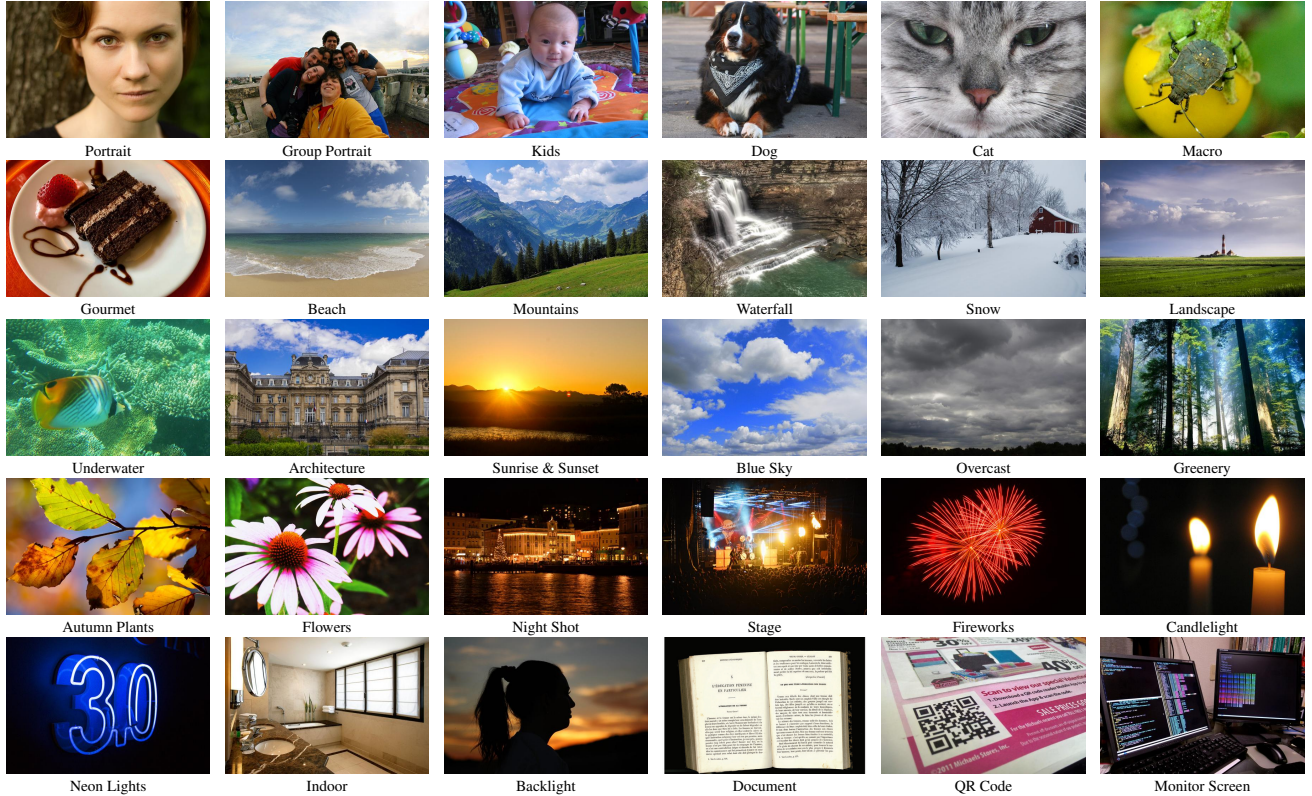

Figure 1. Visualization of the 30 Camera Scene Detection Dataset (CamSDD) categories.

belonging 30 camera scene categories. Since many mobile and IoT platforms can efficiently accelerate INT8 models only, all developed solutions had to be fully-quantized. Within the challenge, the participants were evaluating the runtime and tuning their models on the Apple Bionic A11 SoC used as the target platform for this task. The final score of each submitted solution was based on the runtime and top-1 / top-3 accuracy results, thus balancing between the precision and efficiency of the proposed model. Finally, all developed solutions are fully compatible with the TensorFlow Lite framework [40], thus can be deployed and accelerated on any mobile platform providing AI acceleration through the Android Neural Networks API (NNAPI) [3] or custom TFLite delegates [8].

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

- Learned Smartphone ISP on Mobile NPUs [11]
- Real Image Denoising on Mobile GPUs [10]
- Quantized Image Super-Resolution on Edge SoC NPUs [19]
- Real-Time Video Super-Resolution on Mobile GPUs [16]
- Single-Image Depth Estimation on Mobile Devices [13]
- Quantized Camera Scene Detection on Smartphones
- 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

In this challenge, we use the Camera Scene Detection Dataset (CamSDD) [36] that provides high-quality diverse data for the considered task. It consists of 30 different scene
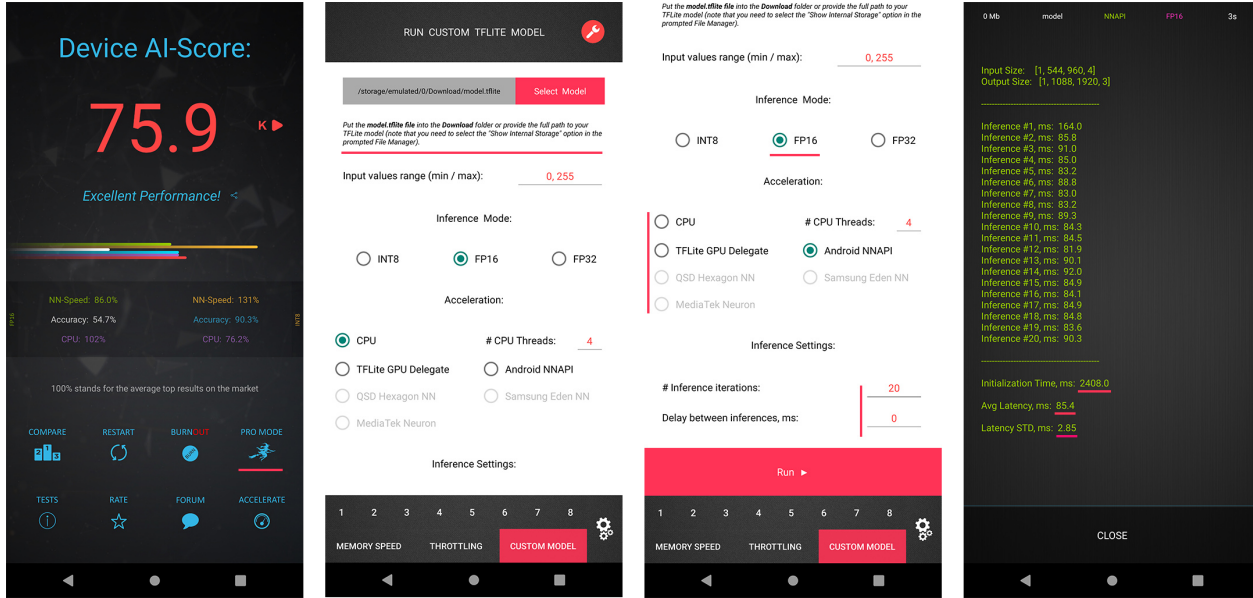
Figure 2. Loading and running custom TensorFlow Lite models with AI Benchmark application. The currently supported acceleration options include Android NNAPI, TFLite GPU, Hexagon NN, Samsung Eden and MediaTek Neuron delegates as well as CPU inference through TFLite or XNNPACK backends. The latest app version can be downloaded at https://ai-benchmark.com/download

categories shown in Fig. 1 and contains more than 11K images that were crawled from *Flickr*[1] using the same setup as in [12]. All photos were inspected manually to remove monochrome and heavily edited pictures, images with distorted colors and watermarks, photos that are impossible for smartphone cameras (*e.g.*, professional underwater or night shots), etc. The dataset was designed to contain diverse images, therefore each scene category contains photos taken in different places, from different viewpoints and angles: *e.g.*, the "cat" category does not only contain cat faces but also normal full-body pictures shot from different positions. This diversity is essential for training a model that is generalizable to different environments and shooting conditions. Each image from the CamSDD dataset belongs to only one scene category. The dataset was designed to be balanced, thus each category contains on average around 350 photos. After the images were collected, they were resized to 576×384 px resolution as using larger photos will not bring any information that is vital for the considered classification problem while will increase the processing time.

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 [18, 21] 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.

2.3. Runtime Evaluation on the Target Platform

In this challenge, we use the *Apple Bionic A11* chipset with the 1-st generation *Neural Engine* that can be found in the iPhone X and iPhone 8 / 8 Plus smartphones as our target runtime evaluation platform. The runtime of all solutions was tested using the TensorFlow Lite CoreML delegate [7]

---

[1]https://www.flickr.com/

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

Team	Author	Framework	Model Size, MB	Top-1, %	Top-3, %	Final Runtime, ms	Final Score
ByteScene	halfmoonchen	Keras / TensorFlow	8.2	95.00	99.50	4.44	163.08
EVAI	EVAI	Keras / TensorFlow	0.83	93.00	98.00	3.35	19.1
MobileNet-V2 [36]	*Baseline*	TensorFlow	18.6	94.17	98.67	16.38	13.99
ALONG	YxChen03	TensorFlow	12.7	94.67	99.50	64.45	8.94
Team Horizon	tensorcat	Keras / TensorFlow	2.27	92.33	98.67	7.7	8.31
Airia-Det	stvea	TensorFlow	1.36	93.00	99.00	17.51	7.31
DataArt Perceptrons	andrewBatutin	Keras / TensorFlow	2.73	91.50	97.67	54.13	0.33
PyImageSearch	thanki	TensorFlow	2.02	89.67	97.83	45.88	0.12
neptuneai	neptuneai	Keras / TensorFlow	0.045	83.67	94.67	4.17	0
Sidiki	Sidiki	Keras / TensorFlow	0.072	78.00	93.83	1.74	0

Table 1. Mobile AI 2021 Quantized Camera Scene Detection challenge results and final rankings. The runtime values were obtained on 576×384 px images on the Apple Bionic A11 platform. Top-1 / Top-3 accuracy was computed using the provided fully-quantized TFLite models. Team *ByteScene* is the challenge winner, *Baseline* corresponds to the MobileNet-V2 based solution presented in [36].

containing many important performance optimizations for this SoC, the latest iOS 14 operating system was installed on the smartphone. Within the challenge, the participants were able to upload their TFLite models to the runtime validation server connected to a real iPhone device 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 and predictions to the remote server to check the accuracy 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:

• Top-1 accuracy checking how accurately was recognized the main scene category,

• Top-3 accuracy assessing the quality of the top 3 predicted classes,

• The runtime on the target Apple Bionic A11 platform.

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

$$\text{Final Score} = \frac{2^{\,(\text{Top-1} + \text{Top-3})}}{C \cdot \text{runtime}},$$

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 accuracy 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 120 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 Top-1 / Top-3 accuracy and the 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 submitted solutions demonstrated a very high efficiency: the majority of models are able to detect the camera scene category at more than 50 FPS on the target Bionic A11 SoC while achieving a top-3 accuracy of over 97-98%. All teams were using either MobileNet [37, 9] or Efficient-Net [39] backbones pre-trained on the ImageNet dataset except for team *Sidiki* that proposed a very tiny classification model which size is only 73KB. Despite using only 60K parameters, it is still able to achieve a top-3 accuracy of more than 93% on the final test set, thus might be practically useful for constraint embedded systems with very tough requirements regarding the model size and memory consumption. Team *ByteScene* is the challenge winner — the proposed MobileNet-V3 based model was able to outperform all other solutions in terms of top-1 accuracy while showing more than 220 FPS on the Bionic SoC. To achieve these results, the authors used the Hierarchical Neural Ar-

| Mobile SoC | Snapdragon 888 | Snapdragon 855 | Dimensity 1000+ | Dimensity 800 | Exynos 2100 | Exynos 990 | Kirin 990 5G | Kirin 980 |
AI Accelerator	Adreno 660 GPU, fps	Hexagon 690, fps	APU 3.0 (6 cores), fps	APU 3.0 (4 cores), fps	NPU, fps	Mali-G77 GPU, fps	Mali-G76 GPU, fps	Mali-G76 GPU, fps
ByteScene	198	191	224	156	**262**	98	88	94
EVAI	290	347	**408**	275	292	144	138	166
ALONG	53	53	**71**	52	37	27	27	30
Team Horizon	**244**	224	171	122	174	101	111	110
Airia-Det	**207**	126	190	125	92	68	64	74
DataArt Perceptrons	91	68	61	41	30	28	**92**	42
PyImageSearch	**94**	68	73	51	39	32	51	43
neptuneai	**578**	274	252	144	204	181	195	245
Sidiki	**427**	400	198	161	373	228	164	154

Table 2. The speed of the proposed solutions on several popular mobile platforms. The runtime was measured with the AI Benchmark app using the fastest acceleration option for each device. The best FPS rate for each solution is denoted in bold.

chitecture Search to adapt the baseline architecture to the target accuracy / runtime constraints, and additional training data obtained by pseudo-labeling with a larger and more accurate network pre-trained on the CamSDD dataset. The same approach of data augmentation was also used by team *ALONG* that generated additional 50K training images using a similar strategy.

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 most efficient acceleration option. We should note that though the Snapdragon 888 and the Kirin 990 5G SoCs have NPUs / DSPs supporting quantized inference, they are not compatible with TensorFlow-2.x models right now, thus in these two cases their GPUs were used instead for accelerating the networks. As one can see, all models were able to achieve real-time performance with more than 27-30 classified images per second on all considered platforms. The winning solution from team *ByteScene* was able reach more than 150-200 FPS on the latest chipsets from Qualcomm, MediaTek and Samsung, and should be able to perform real-time classification on all recent high-end, mid-range and even low-end platforms. The solution from team *EVAI* can provide an additional 50-100% runtime boost at the expense of a slightly lower accuracy. Overall, the obtained results also demonstrate the efficiency of mobile AI accelerators for image classification tasks as they can achieve enormous processing rates while maintaining low power consumption.

4. Challenge Methods

This section describes solutions submitted by all teams participating in the final stage of the MAI 2021 Real-Time Camera Scene Detection challenge.

4.1. ByteScene

Team ByteScene used a transfer learning method (Fig. 3) inspired by the Big Transfer (BiT) [29] to train mobile and big classification models. The first one is using a MobileNet-V3 like architecture demonstrated in Table 3 that was searched using the Hierarchical Neural Architecture Search [45] with additional runtime constraints, and
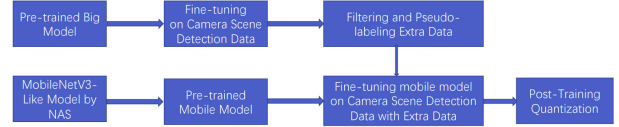

Figure 3. The training scheme used by ByteScene team.

achieves a top-1 accuracy of 67.82% on the ImageNet2012 validation set. Two fully-connected layers were appended on top of the model to the 86M FLOPs backbone. To reduce the amount of computations, the input image was resized from $576{\times}384$px to $128{\times}128$px, normalized to range $[-1, +1]$, and then passed to the network.

Big model training. The authors based their large model on the ResNet101x3 architecture. With 1003 held-out labeled images as a validation set, it was first fine-tuned on the CamSDD data. During fine-tuning, its backbone was frozen and only classifier parameters were optimized for 10 epochs using AdamW [33] with a batch size of 256, an initial learning rate of $1.5e-3$ and a piece-wise constant decay by a factor of 10 at steps of 200 and 300. Sparse categorical cross-entropy was used as an objective function. During the training process, the images were resized to $256{\times}256$px and then randomly cropped to $224{\times}224$px resolution. After performing pseudo-labeling of extra data to get a new validation set, the model was trained again using all available training images. The resulting big model achieved a top-1 accuracy of 97.83% on the official validation set.

Mobile model training. The pre-trained on the ImageNet2012 mobile model (Table 3) was first fine-tuned on the CamSDD data with a frozen backbone using AdamW optimizer with an initial learning rate of $1.5e - 3$, a weight decay of $4e - 5$, a batch size 256, and a schedule length of 400. The objective function was the same as for training of the big model. The input images were resized to $160{\times}160$px and then randomly cropped to $128{\times}128$px. Next, the backbone was unfreezed, and the model was fine-tuned for additional 800 steps with the same optimization parameters. Finally, the backbone was frozen again, and the model was fine-tuned for additional 400 steps with the SGDW optimizer. During the third fine-tuning, the train-

Input shape	Block	Exp Size	SE	Stride
$128^2 \times 3$	conv2d, 3×3	16	No	2
$64^2 \times 16$	bneck, 3×3	16	No	1
$64^2 \times 16$	bneck, 3×3	48	Yes	2
$32^2 \times 24$	bneck, 3×3	72	Yes	2
$16^2 \times 32$	bneck, 3×3	64	Yes	1
$16^2 \times 32$	bneck, 3×3	96	Yes	1
$16^2 \times 32$	bneck, 3×3	96	No	2
$8^2 \times 64$	bneck, 3×3	128	No	1
$8^2 \times 64$	bneck, 3×3	256	No	1
$8^2 \times 64$	bneck, 3×3	320	Yes	1
$8^2 \times 96$	bneck, 3×3	192	Yes	1
$8^2 \times 96$	bneck, 3×3	288	Yes	1
$8^2 \times 96$	bneck, 3×3	576	Yes	2
$4^2 \times 192$	bneck, 3×3	768	Yes	1
$4^2 \times 192$	bneck, 3×3	960	Yes	1
$4^2 \times 192$	bneck, 3×3	768	Yes	1
$4^2 \times 192$	bneck, 3×3	960	Yes	1
$4^2 \times 192$	bneck, 3×3	960	Yes	1
$4^2 \times 192$	bneck, 3×3	768	Yes	1
$4^2 \times 192$	bneck, 3×3	1152	Yes	1
$4^2 \times 192$	conv2d, 1×1	1024	No	1
$4^2 \times 1024$	global avgpool	1024	No	1
$1^2 \times 1024$	fc, Relu6	1280	No	-
1280	fc	30	No	-

Table 3. The model architecture proposed by ByteScene. The MobilenetV3 Bottleneck blocks *(bneck)* [9] are used to build the model. For a *bneck* block, *Exp Size* denotes the number of channels in the expansion layer. For the other blocks, *Exp Size* denotes the output channel number of a block. *SE* denotes whether there is a *Squeeze-And-Excite* op in the block.

ing images were directly resized to 128×128 pixels. In addition to the official 9897 CamSDD training images, 2577 extra training images were used (pseudo-labeled by the pre-trained big model) when training the mobile model.

The resulting quantized INT8 TFLite model was obtained using the standard TensorFlow's post-training quantization tools. To preserve the model accuracy after quantization, the authors used only ReLU6 and HardSigmoid [9] nonlinearities.

4.2. EVAI (Edge Vision and Visual AI)

Team EVAI developed a MobileNetV2-like [37] model with $alpha = 0.75$ and an input shape of 96×144px. This solution demonstrated the best accuracy-latency tradeoff according the experiments conducted by the authors (Table 4).

The architecture of the final model is shown in Fig. 4. A resize layer is introduced to reduce the resolution if the input images by 4 times. The last three blocks (*block15*, *block16* and the last convolutional block) of the original MobileNet-V2 model were removed to reduce the computations. The classification head contains a convolutional block with separable convolution layer, batch normalization layer and *ReLU* activations. The output of this block has 30
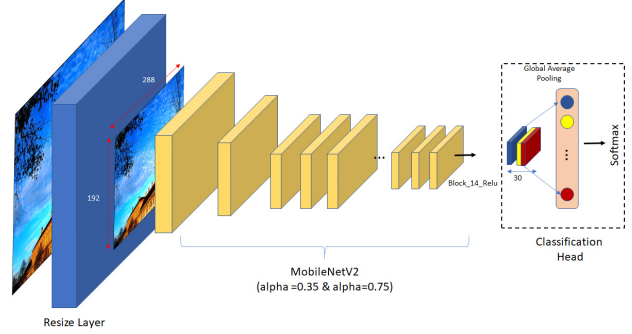

Figure 4. The model architecture proposed by team EVAI.

feature maps, its spacial resolution is then reduced with one global average pooling layer, and the final predictions are obtained after applying the *Softmax* activation. The proposed network does not have any dense layers.

The model was trained for 200 epochs to minimize categorical cross-entropy loss function with label smoothing $\epsilon = 0.1$. The model parameters were optimized using stochastic gradient descent with momentum 0.9 and exponential learning rate scheduler with the initial learning rate of $5e - 2$. The authors used a batch size is 128 and applied additional data augmentation during the training such as flips, grid distortions, shift-scale-rotations, perspective transforms, random resized crops, color jitter and blurring. The final INT8 model was obtained using TensorFlow's post-training quantization tools.

Model Details	Val. Accuracy FP32, %	Val. Accuracy INT8, %	Latency FP32, ms	Latency INT8, ms
MobileNetV2 a=0.35 96×144	90.3	90	8	3
MobileNetV2 a=0.35 192×288	95	94.5	14	8
MobileNetV2 a=0.75 96×144	93.6	92.6	9	4

Table 4. The performance of different MobileNet-V2 backbones obtained by team EVAI.
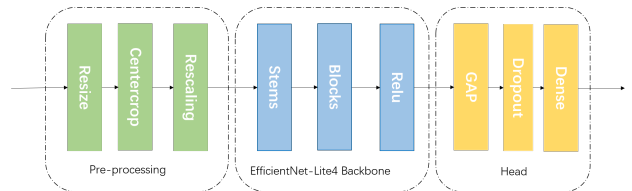
4.3. ALONG

Figure 5. The EfficientNet-based model proposed by ALONG.

Team ALONG based its solution on the EfficientNet-Lite4 [39] model, the architecture of the model is pre-

sented in Fig. 5. The authors used web crawlers to collect a large number of additional training images. Self-distillation method with 4-fold cross-validation was used to retain images with only high classification probability — 50K images were retained and integrated with the original dataset. This extra data improved the classification accuracy by around 2%. An additional 1% accuracy increase was achieved by re-weighting misclassified samples. Since using the original images for model training can result in overfitting, the authors used various data augmentation methods including random cropping, rotation, flipping, brightness and contrast adjustments, blur and mix-cut. In addition, dropout and weight decay were applied. The classification accuracy of the model was improved by around 1% through the above methods (Table 4.3).

The model parameters were optimized using SGD with an initial learning rate of 0.01 and momentum 0.9 for 200 epochs with a batch size of 320. The learning rate scheduler was as follows: 0.01 for the first 2 epochs, 0.001 for the next 118 epochs, and 0.0001 for the rest of the training. The objective function is the categorical cross-entropy with label smoothing ($\epsilon = 0.2$). The final INT8 model was obtained using TensorFlow's post-training quantization tools.

Method	QPT	QAT	Improvement	Acc	Latency
NasNetMobile	+	+	-	92.1%	40ms
MobileNetV3Large	+	-	-	93.3%	77ms
EfficientNet-B1	+	-	-	93.6%	89ms
NasNetLarge	-	-	-	96.6%	1500ms
Bit-m R50x1	-	-	+ImageNet21K Pretrained	97.8%	1200ms
EfficientNet-Lite0	+	+	-	92.0%	33ms
EfficientNet-Lite4	+	+	-	93.0%	74ms
EfficientNet-Lite4	+	+	+ Data Augmentation	94.0%	74ms
EfficientNet-Lite4	+	+	+ Image Collection	95.0%	74ms
EfficientNet-Lite4	+	+	+ Samples Reweighting	94.0%	74ms
EfficientNet-Lite4	+	+	+ All	97.0%	74ms

Table 5. The performance of different architectures and training strategies obtained by team ALONG on the validation set. *QPT* stands for quantization post training, and *QAT* means quantization-aware training.

4.4. Team Horizon

The authors used a MobileNetV2-based model, where the input image was resizes to $1/3$ of its original resolution to improve the runtime results at the expense of minor reduction in accuracy. The model was trained in 4 steps. First, the parameters of the classification head were optimized with Adam for 10 epochs with the default learning rate, then the classification head was trained for another 10 epochs with a reduced learning rate of $1e - 4$. Next, the whole model was trained for 10 epochs with a learning rate of $1e - 5$, and at the last step it was optimized again with a learning rate of $1e - 6$. The network was trained with a batch size of 256 to minimize the cross-entropy loss function in all steps.

4.5. Airia-Det

Team Airia-Det also based its solution on the MobileNetV2 [37] architecture with several modifications. In the classification head, a 5×5 depth separable convolution is used instead of two 3×3 convolutional layers to obtain a larger receptive field and to reduce the number of parameters. The input of the model is resized from 576×384px to 384×384px. The network was trained to minimize the categorical cross-entropy with label smoothing ($\epsilon = 0.1$) using Adam optimizer and a batch size of 32.

4.6. DataArt Perceptrons

TFLite Model	Type	Size (MB)	Top-1 (%)	iPhone 12 Pro CPU, ms	Score
MobileNet-V2	FP	8.6	92.5	83	0.0941
MobileNet-V2	INT8	2.7	91.3	58	0.0255
EfficientNet	FP	29.5	96	270	3.7037
EfficientNet	INT8	9.1	95.3	280	1.3533

Table 6. The performance on the validation dataset and the runtime of different models developed by DataArt Perceptrons team.

The authors used a standard MobileNet-V2 model and trained it on the provided images with various data augmentation methods such as rescaling, random rotations, flips, contrast adjustment and translation. The size of the input layer shape was set to (384, 576, 3) to fit the challenge requirements, the layers of the backbone model were freezed during the training. The final model was trained using Adam optimizer with a learning rate of $5e - 3$, a batch size 32 and the categorical cross-entropy loss function. Though the authors obtained better results with the EfficientNet model (Table 4.6), they faced the problems related to its quantization and decided to use the MobileNetV2-based solution instead.
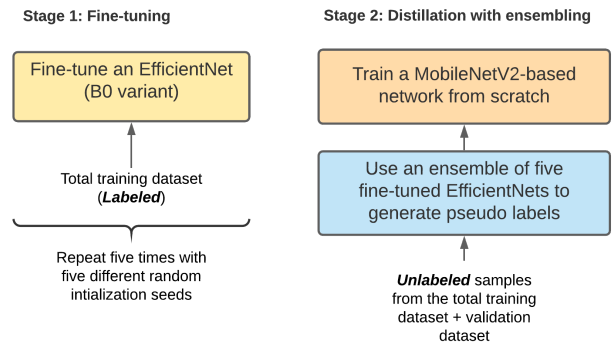
4.7. PyImageSearch

Figure 6. The solution proposed by PyImageSearch team.

Team PyImageSearch used the Noisy Student Training [46] approach (Fig. 6) to solve the considered classification problem. EfficientNet-B0 model [39] was initially

fine-tuned on the training dataset using sparse categorical cross-entropy five times with different random initialization seeds. To train the target MobileNetV2-based model, an ensemble of five obtained EfficientNet-B0 networks was used as a source for pseudo labels. The target model was trained using the KL-divergence as the distillation loss function, the distillation process is illustrated in Fig. 7. Adam optimizer was used for training both teacher and student models. To mitigate the problem of class imbalance, the authors used class weights (calculated from the proportion of samples per class) when calculating the cross-entropy loss for training teacher models. Stochastic weight averaging (SWA) was additionally applied to utilize the previously updated parameters from earlier epochs, which induces an ensembling effect [26]. The student model was trained for 110 epochs with early stopping.
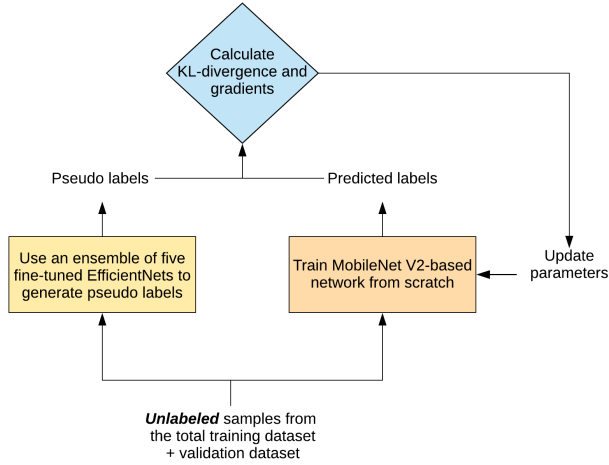

Figure 7. Knowledge distillation approach used by PyImage-Search.

4.8. Sidiki

This is the only solution that is not using any backbone networks. The proposed model can achieve an accuracy of over 80% on the validation set with only 60K parameters and the total model size of 73 KB. The model uses only 8 convolutional layers, 4 pooling layers and a fully connected layer. To speed up the inference, downsampling is applied in the first two convolutional layers by using strides 4 and 2 together with a small (6 and 12) number of channels.

The model was trained using Adam optimizer with a batch size of 32 as follows: for 10 epochs with a learning rate of $1e-3$, for 40 epochs with a learning rate of $1e-4$, then for 100 epochs with a learning rate of $7e-5$ and another 100 epochs with a learning rate of $9e-5$. After that, all layers of the model were prepared for Quantization Aware Training and then the model was trained for an-

other 100 epochs with a learning rate of $1e-4$, and for 100 epochs — with $1e-5$. Once the final accuracy of 80% was established on the validation set, the validation set was used for training of the quantized aware model for 50 epochs with a learning rate of $1e-4$. Then, the model was trained again on the original training set for 10 epochs with a learning rate of $1e-4$, and for 10 — with $1e-6$. Finally, the model was repeatedly trained around 6-8 times on the training and validation sets for 10 and 1 epoch, respectively.

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: [20, 24]

- Perceptual Image Enhancement: [23, 17]

- Image Super-Resolution: [23, 34, 4, 41]

- Bokeh Effect Rendering: [15, 22]

- Image Denoising: [1, 2]

Acknowledgements

A. Teams and Affiliations

Mobile AI 2021 Team

Title:
Mobile AI 2021 Challenge on Fast and Accurate Quantized Camera Scene Detection on Smartphones
Members:
Andrey Ignatov[1,2] *(andrey@vision.ee.ethz.ch)*, Grigory Malivenko *(grigory.malivenko@gmail.com)*, Radu Timofte[1,2] *(radu.timofte@vision.ee.ethz.ch)*
Affiliations:
[1] Computer Vision Lab, ETH Zurich, Switzerland
[2] AI Witchlabs, Switzerland

ByteScene

Title:
Transfer Knowledge from Both Big Pre-trained Models and Small Pre-trained Models
Members:
Sheng Chen (chensheng.lab@bytedance.com), Xin Xia, Zhaoyan Liu, Yuwei Zhang, Feng Zhu, Jiashi Li, Xuefeng Xiao, Yuan Tian, Xinglong Wu

Affiliations:
ByteDance Inc., China

EVAI (Edge Vision and Visual AI)

Title:
Scene Detection with truncated MobileNetV2 Backbone
Members:
Christos Kyrkou (ckyrko01@ucy.ac.cy)
Affiliations:
KIOS Research and Innovation Center of Excellence, University of Cyprus, Cyprus

ALONG

Title:
EfficientNet-Lite4 for Real-Time Scene Detection
Members:
Yixin Chen (chenyixin03@corp.netease.com), Zexin Zhang, Yunbo Peng, Yue Lin
Affiliations:
Netease Games AI Lab, China

Team Horizon

Title:
MobileNetV2 with Resized Input for Fast Scene Detection
Members:
Saikat Dutta[1] *(saikat.dutta779@gmail.com)*, Sourya Dipta Das[2], Nisarg A. Shah[3], Himanshu Kumar[3]
Affiliations:
[1] Indian Institute of Technology Madras, India
[2] Jadavpur University, India
[3] Indian Institute of Technology Jodhpur, India

Airia-Det

Title:
Bag of Tricks for Real-Time Scene Detection Based on MobileNets
Members:
Chao Ge (stvea@qq.com), Pei-Lin Wu Jin-Hua Du
Affiliations:
Nanjing Artificial Intelligence Chip Research, Institute of Automation, Chinese Academy of Sciences, China

DataArt Perceptrons

Title:
Using MobileNetv2 for scene detection
Members:
Andrew Batutin (andrey.batutin@dataart.com), Juan Pablo Federico, Konrad Lyda, Levon Khojoyan
Affiliations:
DataArt Inc, United States

PyImageSearch

Title:
A Bag of Tricks for Mobile-Friendly Image Classification
Members:
Sayak Paul (s.paul@pyimagesearch.com), Abhishek Thanki
Affiliations:
PyImageSearch, India

Sidiki

Title:
Tiny Mobile AI Model
Members:
Shahid Siddiqui (msiddi01@ucy.ac.cy)
Affiliations:
KIOS Center of Excellence, University of Cyprus, Cyprus

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. 8

[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. 8

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

[4] 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. 8

[5] 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

[6] 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

[7] TensorFlow Lite Core ML delegate. `https://www.tensorflow.org/lite/performance/coreml_delegate`. 3

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

[9] 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, 4, 6

[10] 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

[11] 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

[12] 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. 3

[13] Andrey Ignatov, Grigory Malivenko, David Plowman, Samarth Shukla, and Radu Timofte. 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*, pages 0–0, 2021. 2

[14] 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

[15] 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. 8

[16] 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

[17] 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. 8

[18] 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

[19] 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

[20] 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. 8

[21] 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

[22] 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. 8

[23] 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. 8

[24] 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. 8

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

[26] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 8

[27] 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

[28] 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

[29] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 6(2):8, 2019. 5

[30] Yawei Li, Shuhang Gu, Luc Van Gool, and Radu Timofte. Learning filter basis for convolutional neural network com-

pression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5623–5632, 2019. 1

[31] 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

[32] 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

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[34] 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. 8

[35] 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

[36] Angeline Pouget, Sidharth Ramesh, Maximilian Giang, Ramithan Chandrapalan, Toni Tanner, Moritz Prussing, Radu Timofte, and Andrey Ignatov. Fast and accurate camera scene detection on smartphones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2021. 1, 2, 4

[37] 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. 4, 6, 7

[38] 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

[39] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 4, 6, 7

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

[41] 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. 8

[42] 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

[43] 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

[44] 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

[45] Xin Xia and Wenrui Ding. Hnas: Hierarchical neural architecture search on mobile devices. *arXiv preprint arXiv:2005.07564*, 2020. 5

[46] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020. 7

[47] 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