An Analysis of the Performance of Lightweight CNNs in the Context of Object Detection on Mobile Phones

Jakub Łęcki
Faculty of Electronics, Telecommunications and Informatics
Gdańsk University of Technology, Poland *s175494@student.pg.edu.pl*

Marek Hering
Faculty of Electronics, Telecommunications and Informatics
Gdańsk University of Technology, Poland *s175729@student.pg.edu.pl*

Maciej Jabłoński
Faculty of Electronics, Telecommunications and Informatics
Gdańsk University of Technology, Poland *s175591@student.pg.edu.pl*

Aleksandra Karpus
Faculty of Electronics, Telecommunications and Informatics
Gdańsk University of Technology, Poland *alekarpu@pg.edu.pl*

Abstract

Convolutional Neural Networks (CNNs) are widely used in computer vision, which is now increasingly used in mobile phones. The problem is that smartphones do not have much processing power. Initially, CNNs focused solely on increasing accuracy. High-end computing devices are most often used in this type of research. The most popular application of lightweight CNN object detection is real-time image processing, which can be found in devices such as cameras and autonomous vehicles. Therefore, there is a need to optimize CNNs for use on mobile devices. This paper presents the comparision of latency and mAP of 22 lightweight CNN models from the *MobileNet* and *EfficientDet* families measured on 7 mobile phones.

Keywords: lightweight CNNs, object detection, mobile phones, smartphones, mobile devices.

1. Introduction

Computer Vision (CV) strives to make computers perceive visual information like humans. One key task is object detection, where a computer identifies and labels objects in images using Convolutional Neural Networks (CNNs)[3, 2]. CNNs excel in image analysis by executing complex matrix operations.Thus, CV problems have been solved by high-performance devices.

On the other side, the idea of the Internet of Things assumes the presence of network communication in most everyday objects [12]. These may be devices such as TVs or speakers, but also thermostats and refrigerators. To sustain device functionality, it is necessary to use network and computing modules of miniature size. Reduced device sizes limit computing power, necessitating simplified Neural Network (NN) structures tailored for mobile devices [4]. Lightweight CNNs (LCNNs) like *MobileNet*, *EfficientDet*, *Pelee*, and *ThunderNet* are designed for swift inference in applications like autonomous cars, drones, and smartphone cameras [4].

The use of computer vision on mobile devices is now widely researched [4, 10, 14, 1, 15, 9, 16]. The aim of this work was to investigate the performance of LCNN models on various mobile phones and to answer the following research questions.
(RQ1) Which of the LCNN models achieves the highest performance on mobile phones?
(RQ2) What impact does the used computing unit have on the performance of a given model?

(RQ3) What impact does quantization optimization have on the performance of LCNNs?
We address these questions in the following sections.

2. Related Work

The *MobileNet* series stands out among CNN architectures for mobile devices. Versions by Howard et al. [6, 11, 5] are noteworthy, evaluated using the *COCO 2017* dataset for device-independent *MAdd* operation comparisons. While this metric aids theoretical model evaluation, real performance may vary based on device hardware and software.

EfficientDet by Tan et al.[13] is another architecture that is worth attention. For comparison of different models, the authors use the *FLOPs* parameter, the *AP* metric as a measure of effectiveness, and the inference time on a desktop computer's graphics card as delay in *ms*.

Ignatov et al. [7] extensively reviewed Android devices for AI capabilities, testing over 10,000 devices across 9 image processing tasks. They highlighted the advantage of *TensorFlow Lite* for standardized inferences on various smartphones. Luo et al. [8] compared mobile technologies like *PyTorch*, *Caffe2*, and *TensorFlow Lite* based on efficiency and inference time of CNN models. They tests concluded that the models behave differently on different devices, making it difficult to compare the performance of individual models, but it was found that *TensorFlow Lite* allows for the fastest model initialization by using the optimal data format.

The aforesaid works showed the dominance of *MobileNet* and *EfficientDet*, prompting our comparison of these architectures. We opted for measuring real performance through inference time in *ms* rather than focusing on parameters or operations. In contrast to high-powered GPU setups in *EfficientDet*'s work, we conducted our experiments on mobile phones. Utilizing the *TensorFlow Lite* framework aligns with its proven advantages in the literature cited [7, 8].

3. Experiments

In order to evaluate the performance of LCNNs, we examined the influence of factors such as platform, CNN model and computation unit on the accuracy and inference time. We used 7 smartphones whose abbreviated specification is shared online[1].

We performed experiments on MS Coco(*Microsoft Common Objects in Context*) which is a dataset commonly used in computer vision problems. This collection consists of 328,000 digital images depicting common objects (e.g. a person or a car) in everyday situations. A set of fifty selected photos from the COCO 2017 test set was prepared for the study.

3.1. CNN Models

For the experiment, trained CNN models are crucial. Ensuring alignment with authors' results, the following assumptions were considered to minimize discrepancies: A. models are prepared for object detection; B. it is not possible to train models or fine-tune them; C. the tested models recognize the same set of classes and were trained on the same data set; D. models are available in **tflite** format or can be converted to this format; E. models can be run in *TensorFlow 2*. Consequently, we chose models distributed by Google in the TensorFlow Models Repository[2] and on the TensorFlow Hub platform[3]. We encountered some difficulties in using some of these models, including incompatibility between *TensorFlow 1* and *TensorFlow 2*, wrong model format or the need for train a model from scratch. Table 1[4] collects test models.

[1]`https://mostwiedzy.pl/pl/aleksandra-karpus,72346-1/lcnn-resources`

[2]`https://github.com/tensorflow/models`

[3]`https://tfhub.dev/s?deployment-format=lite&module-type=image-object-detection&tf-version=tf2`

[4]Explanation of features and model designations: *ed* - network from the *EfficientDet* family; *mn* - network from the *MobileNet* family; *_quant* - models that have been fully quantized to fixed-point representation; *_no_optimization* - models converted without

Table 1. Summary of all models used in the experiment

Model	Size	Resolution
ed-lite0_default	4.34 MB	320 × 320
ed-lite0_quant	4.34 MB	320 × 320
ed-lite1_default	5.79 MB	384 × 384
ed-lite2_default	7.20 MB	448 × 448
ed-lite3_default	11.39 MB	512 × 512
ed-lite3x_default	13.32 MB	640 × 640
ed-lite4_default	19.88 MB	640 × 640
mn-v1-300x300_default	26.02 MB	300 × 300
mn-v1-fpn-640x640_default	30.88 MB	640 × 640
mn-v1-fpn-640x640_no_optimization	118.85 MB	640 × 640
mn-v1-fpn-640x640_quant	31.0 MB	640 × 640
mn-v2-320x320_default	6.34 MB	320 × 320
mn-v2-320x320_no_optimization	23.15 MB	320 × 320
mn-v2-320x320_quant	6.42 MB	320 × 320
mn-v2-fpnlite-320x320_default	3.51 MB	320 × 320
mn-v2-fpnlite-320x320_no_optimization	11.22 MB	320 × 320
mn-v2-fpnlite-320x320_quant	3.69 MB	320 × 320
mn-v2-fpnlite-640x640_default	4.10 MB	640 × 640
mn-v2-fpnlite-640x640_no_optimization	11.81 MB	640 × 640
mn-v2-fpnlite-640x640_quant	4.28 MB	640 × 640
mn-v3-large_default	12.42 MB	320 × 320
mn-v3-small_default	6.86 MB	320 × 320

Diverse model sources lead to varied parameters, limiting comprehensive coefficient impact assessment on LCNN performance. Additionally, the low availability of fully quantized models reduces the generalization possibilities of the results of this type of networks.

3.2. Test Software

To assess LCNN model performance on mobile devices, we developed a Kotlin application[5] enabling testing on CPU and GPU. Images were initially scaled to match the model's input layer size. To prevent device overheating and result degradation, we conducted single inference time measurements on a prepared image set. Latency was evaluated through the TensorFlow Lite API's inference function across all test set images, repeated for each model. Test results, including latency and detected object details, were stored in device memory for accuracy calculation later. Using ADB, the application was installed on smartphones with device optimizations before testing commenced.

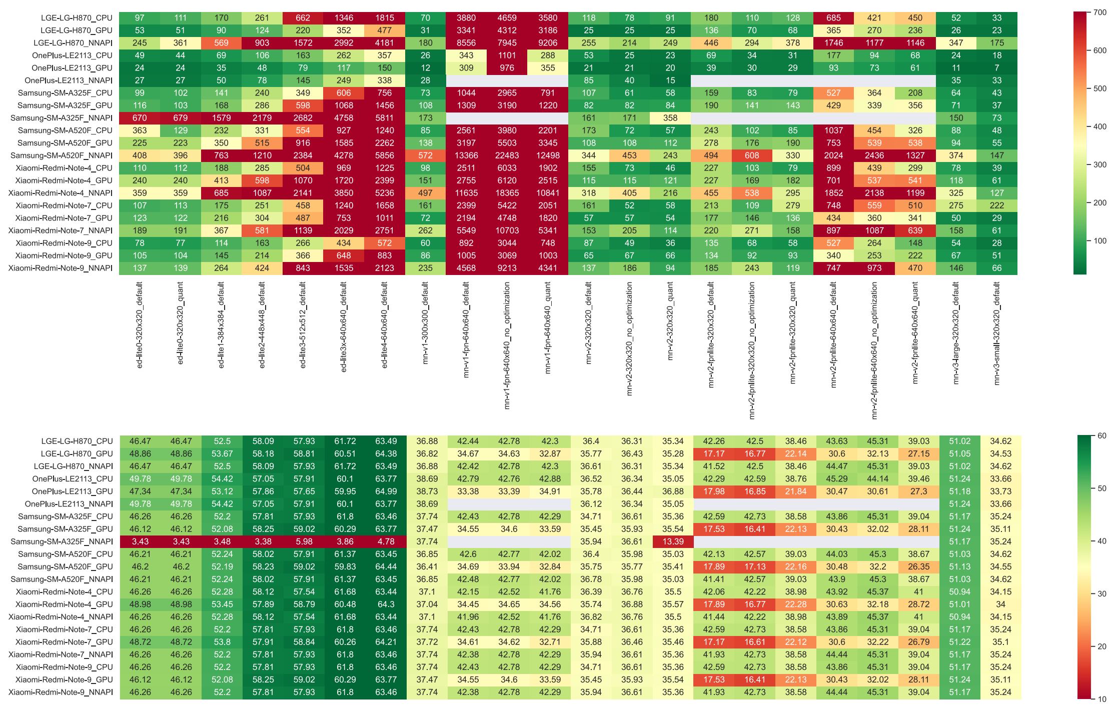
4. Results and Discussion

Final latency results for each LCNN model on tested smartphones were derived from the median of all inference times. Accuracy was measured using a script for computing mAP[6]. Findings are displayed in Figure 1. Latency results exhibit a significant variation. The average inference time value is very high compared to the performance expected from the *tflite* models, and the standard deviation indicates large differences in the delay introduced by different models or devices. Results primarily fall within the 0ms to 744ms range, with only a few instances showing considerable delays. Figure 1 highlights that the model has a greater impact on both latency and accuracy results compared to the device. Some models like *MobileNet* v1-fpn or *EfficientDet*

using optimization (saving weights in floating-point format); *_default* models imply the use of dynamic range quantization, also called default optimization; *fpn* and *fpnlite* means usage of a feature pyramid or a lightweight version of it; *efficientdet-lite0* is the model on which the rest of the *EfficientDet* models are based.

[5]Models and code are available at `https://gitlab.com/mobile_cnn_detection_analize_pg`

[6]`https://github.com/Cartucho/mAP`



	ed-lite0-320x320_default	ed-lite0-320x320_quant	ed-lite1-384x384_default	ed-lite2-448x448_default	ed-lite3-512x512_default	ed-lite3x-640x640_default	ed-lite4-640x640_default	mn-v1-300x300_default	mn-v1-fpn-640x640_default	mn-v1-fpn-640x640_no_optimization	mn-v1-fpn-640x640_quant
LGE-LG-H870_CPU	97	111	170	261	662	1346	1815	70	3880	4659	3580
LGE-LG-H870_GPU	53	51	90	124	220	352	477	31	3341	4312	3186
LGE-LG-H870_NNAPI	245	361	569	903	1572	2992	4181	180	8556	7945	9206
OnePlus-LE2113_CPU	49	44	69	106	163	262	357	26	343	1101	288
OnePlus-LE2113_GPU	24	24	35	48	79	117	150	12	309	976	355
OnePlus-LE2113_NNAPI	27	27	50	78	145	249	338	28			
Samsung-SM-A325F_CPU	99	102	141	240	349	606	756	73	1044	2965	791
Samsung-SM-A325F_GPU	116	103	168	286	598	1068	1456	108	1309	3190	1220
Samsung-SM-A325F_NNAPI	670	679	1579	2179	2682	4758	5811	173			
Samsung-SM-A520F_CPU	363	129	232	331	554	927	1240	85	2561	3980	2201
Samsung-SM-A520F_GPU	225	223	350	515	916	1585	2262	138	3197	5503	3345
Samsung-SM-A520F_NNAPI	408	396	763	1210	2384	4278	5856	572	13366	22483	12498
Xiaomi-Redmi-Note-4_CPU	110	112	188	285	504	969	1225	98	2511	6033	1902
Xiaomi-Redmi-Note-4_GPU	240	240	413	598	1070	1720	2399	151	2755	6120	2515
Xiaomi-Redmi-Note-4_NNAPI	359	359	685	1087	2141	3850	5236	497	11635	18365	10841
Xiaomi-Redmi-Note-7_CPU	107	113	175	251	458	1240	1658	161	2399	5422	2051
Xiaomi-Redmi-Note-7_GPU	123	122	216	304	487	753	1011	72	2194	4748	1820
Xiaomi-Redmi-Note-7_NNAPI	189	191	367	581	1139	2029	2751	262	5549	10703	5341
Xiaomi-Redmi-Note-9_CPU	78	77	114	163	266	434	572	60	892	3044	748
Xiaomi-Redmi-Note-9_GPU	105	104	145	214	366	648	883	86	1005	3069	1003
Xiaomi-Redmi-Note-9_NNAPI	137	139	264	424	843	1535	2123	235	4568	9213	4341

	mn-v2-320x320_default	mn-v2-320x320_no_optimization	mn-v2-320x320_quant	mn-v2-fpnlite-320x320_default	mn-v2-fpnlite-320x320_no_optimization	mn-v2-fpnlite-320x320_quant	mn-v2-fpnlite-640x640_default	mn-v2-fpnlite-640x640_no_optimization	mn-v2-fpnlite-640x640_quant	mn-v3-large-320x320_default	mn-v3-small-320x320_default
LGE-LG-H870_CPU	118	78	91	180	110	128	685	421	450	52	33
LGE-LG-H870_GPU	25	25	25	136	70	68	365	270	236	26	23
LGE-LG-H870_NNAPI	255	214	249	446	294	378	1746	1177	1146	347	175
OnePlus-LE2113_CPU	53	25	23	69	34	31	177	94	68	24	18
OnePlus-LE2113_GPU	21	21	20	39	30	29	93	73	61	11	7
OnePlus-LE2113_NNAPI	85	40	15							35	33
Samsung-SM-A325F_CPU	107	61	58	159	83	79	527	364	208	64	43
Samsung-SM-A325F_GPU	82	82	84	190	141	143	429	339	356	71	37
Samsung-SM-A325F_NNAPI	161	171	358							150	73
Samsung-SM-A520F_CPU	173	72	57	243	102	85	1037	454	326	88	48
Samsung-SM-A520F_GPU	108	108	112	278	176	190	753	539	538	94	55
Samsung-SM-A520F_NNAPI	344	453	243	494	608	330	2024	2436	1327	374	147
Xiaomi-Redmi-Note-4_CPU	155	73	46	227	103	79	899	439	299	78	39
Xiaomi-Redmi-Note-4_GPU	115	115	121	227	169	182	701	537	541	118	61
Xiaomi-Redmi-Note-4_NNAPI	318	405	216	455	538	295	1852	2138	1199	325	127
Xiaomi-Redmi-Note-7_CPU	161	52	58	213	109	279	748	559	510	275	222
Xiaomi-Redmi-Note-7_GPU	57	57	54	177	146	136	434	360	341	50	29
Xiaomi-Redmi-Note-7_NNAPI	153	205	114	220	271	158	897	1087	639	158	61
Xiaomi-Redmi-Note-9_CPU	87	49	36	135	68	58	527	264	148	54	28
Xiaomi-Redmi-Note-9_GPU	65	67	66	134	92	93	340	253	222	67	51
Xiaomi-Redmi-Note-9_NNAPI	137	186	94	185	243	119	747	973	470	146	66

	ed-lite0-320x320_default	ed-lite0-320x320_quant	ed-lite1-384x384_default	ed-lite2-448x448_default	ed-lite3-512x512_default	ed-lite3x-640x640_default	ed-lite4-640x640_default	mn-v1-300x300_default	mn-v1-fpn-640x640_default	mn-v1-fpn-640x640_no_optimization	mn-v1-fpn-640x640_quant
LGE-LG-H870_CPU	46.47	46.47	52.5	58.09	57.93	61.72	63.49	36.88	42.44	42.78	42.3
LGE-LG-H870_GPU	48.86	48.86	53.67	58.18	58.81	60.51	64.38	36.82	34.67	34.63	32.87
LGE-LG-H870_NNAPI	46.47	46.47	52.5	58.09	57.93	61.72	63.49	36.88	42.42	42.78	42.3
OnePlus-LE2113_CPU	49.78	49.78	54.42	57.05	57.91	60.1	63.77	38.69	42.79	42.76	42.88
OnePlus-LE2113_GPU	47.34	47.34	53.12	57.86	57.65	59.95	64.99	38.73	33.38	33.39	34.91
OnePlus-LE2113_NNAPI	49.78	49.78	54.42	57.05	57.91	60.1	63.77	38.69			
Samsung-SM-A325F_CPU	46.26	46.26	52.2	57.81	57.93	61.8	63.46	37.74	42.43	42.78	42.29
Samsung-SM-A325F_GPU	46.12	46.12	52.08	58.25	59.02	60.29	63.77	37.47	34.55	34.6	33.59
Samsung-SM-A325F_NNAPI	3.43	3.43	3.48	3.38	5.98	3.86	4.78	37.74			
Samsung-SM-A520F_CPU	46.21	46.21	52.24	58.02	57.91	61.37	63.45	36.85	42.6	42.77	42.02
Samsung-SM-A520F_GPU	46.2	46.2	52.19	58.23	59.02	59.83	64.44	36.41	34.69	33.94	32.84
Samsung-SM-A520F_NNAPI	46.21	46.21	52.24	58.02	57.91	61.37	63.45	36.85	42.48	42.77	42.02
Xiaomi-Redmi-Note-4_CPU	46.26	46.26	52.28	58.12	57.54	61.68	63.44	37.1	42.15	42.52	41.76
Xiaomi-Redmi-Note-4_GPU	48.98	48.98	53.45	57.89	58.79	60.48	64.3	37.04	34.45	34.65	34.56
Xiaomi-Redmi-Note-4_NNAPI	46.26	46.26	52.28	58.12	57.54	61.68	63.44	37.1	41.96	42.52	41.76
Xiaomi-Redmi-Note-7_CPU	46.26	46.26	52.2	57.81	57.93	61.8	63.46	37.74	42.43	42.78	42.29
Xiaomi-Redmi-Note-7_GPU	48.72	48.72	53.8	57.91	58.84	60.26	64.21	37.72	34.61	34.62	32.71
Xiaomi-Redmi-Note-7_NNAPI	46.26	46.26	52.2	57.81	57.93	61.8	63.46	37.74	42.38	42.78	42.29
Xiaomi-Redmi-Note-9_CPU	46.26	46.26	52.2	57.81	57.93	61.8	63.46	37.74	42.43	42.78	42.29
Xiaomi-Redmi-Note-9_GPU	46.12	46.12	52.08	58.25	59.02	60.29	63.77	37.47	34.55	34.6	33.59
Xiaomi-Redmi-Note-9_NNAPI	46.26	46.26	52.2	57.81	57.93	61.8	63.46	37.74	42.38	42.78	42.29

	mn-v2-320x320_default	mn-v2-320x320_no_optimization	mn-v2-320x320_quant	mn-v2-fpnlite-320x320_default	mn-v2-fpnlite-320x320_no_optimization	mn-v2-fpnlite-320x320_quant	mn-v2-fpnlite-640x640_default	mn-v2-fpnlite-640x640_no_optimization	mn-v2-fpnlite-640x640_quant	mn-v3-large-320x320_default	mn-v3-small-320x320_default
LGE-LG-H870_CPU	36.4	36.31	35.34	42.26	42.5	38.46	43.63	45.31	39.03	51.02	34.62
LGE-LG-H870_GPU	35.77	36.43	35.28	17.17	16.77	22.14	30.6	32.13	27.15	51.05	34.53
LGE-LG-H870_NNAPI	36.61	36.31	35.34	41.52	42.5	38.46	44.47	45.31	39.03	51.02	34.62
OnePlus-LE2113_CPU	36.52	36.34	35.05	42.29	42.59	38.76	45.29	44.14	39.46	51.24	33.66
OnePlus-LE2113_GPU	35.78	36.44	36.88	17.98	16.85	21.84	30.47	30.61	27.3	51.18	33.73
OnePlus-LE2113_NNAPI	36.12	36.34	35.05							51.24	33.66
Samsung-SM-A325F_CPU	34.71	36.61	35.36	42.59	42.73	38.58	43.86	45.31	39.04	51.17	35.24
Samsung-SM-A325F_GPU	35.45	35.93	35.54	17.53	16.41	22.13	30.43	32.02	28.11	51.24	35.11
Samsung-SM-A325F_NNAPI	35.94	36.61	13.39							51.17	35.24
Samsung-SM-A520F_CPU	36.4	35.98	35.03	42.13	42.57	39.03	44.03	45.3	38.67	51.03	34.62
Samsung-SM-A520F_GPU	35.75	35.77	35.41	17.89	17.13	22.16	30.48	32.2	26.35	51.13	34.55
Samsung-SM-A520F_NNAPI	36.78	35.98	35.03	41.41	42.57	39.03	43.9	45.3	38.67	51.03	34.62
Xiaomi-Redmi-Note-4_CPU	36.39	36.76	35.5	42.06	42.22	38.98	43.92	45.37	41	50.94	34.15
Xiaomi-Redmi-Note-4_GPU	35.74	36.88	35.57	17.89	16.77	22.28	30.63	32.18	28.72	51.01	34
Xiaomi-Redmi-Note-4_NNAPI	36.82	36.76	35.5	41.44	42.22	38.98	43.89	45.37	41	50.94	34.15
Xiaomi-Redmi-Note-7_CPU	34.71	36.61	35.36	42.59	42.73	38.58	43.86	45.31	39.04	51.17	35.24
Xiaomi-Redmi-Note-7_GPU	35.88	36.46	35.46	17.17	16.61	22.12	30.6	32.22	26.79	51.22	35.1
Xiaomi-Redmi-Note-7_NNAPI	35.94	36.61	35.36	41.93	42.73	38.58	44.44	45.31	39.04	51.17	35.24
Xiaomi-Redmi-Note-9_CPU	34.71	36.61	35.36	42.59	42.73	38.58	43.86	45.31	39.04	51.17	35.24
Xiaomi-Redmi-Note-9_GPU	35.45	35.93	35.54	17.53	16.41	22.13	30.43	32.02	28.11	51.24	35.11
Xiaomi-Redmi-Note-9_NNAPI	35.94	36.61	35.36	41.93	42.73	38.58	44.44	45.31	39.04	51.17	35.24

Fig. 1. Median latencies in *ms* (top) and mAP effectiveness (bottom) for all tests performed.

versions exhibit notably higher latencies. Models using FPN experience reduced accuracy on GPU execution, possibly due to discrepancies in rounding rules. The OnePlus 9 smartphone delivers superior results among tested devices due to its robust CPU and GPU capabilities.

Addressing **RQ1**, the efficiency comparison among tested architectures remains inconclusive. Different applications may benefit from distinct approaches: MobileNet emphasizes minimal inference time with acceptable mAP values, whereas EfficientDet prioritizes mAP with the lowest possible inference time. Replying to **RQ2**, quantized models outperform unquantized ones on CPUs, but GPU inference results vary unpredictably. Models with fixed-point weights showcased diverse outcomes when tested on GPUs. Answering **RQ3**, even devices without advanced graphics processors are able to obtain satisfactory results on some of the tested models thanks to optimization through quantization.

In summary, there are many ways to adapt lightweight CNNs to suit different needs. For applications in real-time systems where model accuracy is less important, *MobileNet* architecture networks are a much better choice. In turn, the *EfficientDet* architecture will work very well in systems where longer delays are acceptable, e.g. in static photo analysis.

5. Conclussions and Future Work

The study aimed to compare performance of 22 LCNN models from *EfficientDet* and *MobileNet* families on 7 mobile devices. Models ready for testing were prepared, on which various types of optimizations were made.

EfficientDet prioritized accuracy, while *MobileNet* focused on low inference time. Optimal model selection varies based on the application. Quantization optimization enables satisfactory results even on devices without advanced GPUs. Quantized models outperform unquantized ones on CPUs. Predicting quantized model inference on GPUs is challenging, with fixed-point weight models showing diverse GPU outcomes.

The work represents a small part of the LCNNs on mobile devices topic. Testing more devices can improve result applicability.

References

[1] Casanova, C., Franco, A., Lumini, A., and Maio, D.: SmartVisionApp: A framework for computer vision applications on mobile devices. In: *Expert Systems with Applications* 40.15 (2013), pp. 5884–5894.

[2] Cherapanamjeri, J. and Rao, B. N. K.: Neural Networks based Object Detection Techniques in Computer Vision. In: *2022 4th Int. Conf. on Inventive Research in Computing Applications (ICIRCA)*. 2022, pp. 1092–1099.

[3] Garcia-Rodriguez, J.: Advancements in Computer Vision and Image Processing. IGI Global, 2018, pp. 1–322.

[4] Glegoła, W., Karpus, A., and Przybyłek, A.: MobileNet family tailored for Raspberry Pi. In: *Procedia Computer Science* 192 (2021). Knowledge-Based and Intelligent Information & Engineering Systems: Proc. of the 25th Int. Conf. KES2021, pp. 2249–2258.

[5] Howard, A., Sandler, M., Chu, G., Chen, L., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., and Adam, H.: Searching for MobileNetV3. In: *CoRR* abs/1905.02244 (2019). arXiv: `1905.02244`.

[6] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. In: *CoRR* abs/1704.04861 (2017). arXiv: `1704.04861`.

[7] Ignatov, A., Timofte, R., Chou, W., Wang, K., Wu, M., Hartley, T., and Gool, L. V.: AI Benchmark: Running Deep Neural Networks on Android Smartphones. In: *CoRR* abs/1810.01109 (2018). arXiv: `1810.01109`.

[8] Luo, C., He, X., Zhan, J., Wang, L., Gao, W., and Dai, J.: Comparison and Benchmarking of AI Models and Frameworks on Mobile Devices. In: *CoRR* abs/2005.05085 (2020). arXiv: `2005.05085`.

[9] Munir, M., Avery, W., and Marculescu, R.: MobileViG: Graph-Based Sparse Attention for Mobile Vision Applications. In: *2023 IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023, pp. 2211–2219.

[10] Saleh, M. A., Ameen, Z. S., Altrjman, C., and Al-Turjman, F.: Computer-Vision-Based Statue Detection with Gaussian Smoothing Filter and EfficientDet. In: *Sustainability* 14.18 (2022). URL: `https://www.mdpi.com/2071-1050/14/18/11413`.

[11] Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L.: Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. In: *CoRR* abs/1801.04381 (2018). arXiv: `1801.04381`.

[12] Srinivasan, C., Rajesh, B., Saikalyan, P., Premsagar, K., and Yadav, E. S.: A review on the different types of internet of things (IoT). English. In: *Journal of Advanced Research in Dynamical and Control Systems* 11.1 (2019), pp. 154–158.

[13] Tan, M., Pang, R., and Le, Q. V.: EfficientDet: Scalable and Efficient Object Detection. In: *CoRR* abs/1911.09070 (2019). arXiv: `1911.09070`.

[14] Wan, Y., Liu, M., Li, G., and Dong, F.: CoCV: Heterogeneous Processors Collaboration Mechanism for End-to-End Execution of Intelligent Computer Vision Tasks on Mobile Devices. In: *2023 IEEE 29th Int. Conf. on Parallel and Distributed Systems (ICPADS)*. 2023, pp. 2507–2514.

[15] Wcisło, N., Szczepanik, M., and Jóźwiak, I.: Computer Diagnosis of Color Vision Deficiencies Using a Mobile Device. In: *Intelligent and Safe Computer Systems in Control and Diagnostics*. Ed. by Kowalczuk, Z. Springer Cham, 2023, pp. 63–70.

[16] Wu, N., Lin, F. X., Qian, F., and Han, B.: Hybrid mobile vision for emerging applications. In: *Proc. of the 23rd Annual Int. Workshop on Mobile Computing Systems and Applications*. HotMobile ’22. Tempe, Arizona: ACM, 2022, pp. 61–67.