

Lightweight Neural Network Architectures for Efficient Artificial Intelligence Processing on Edge Devices

Aoife Brennan

Research Fellow, Atlantic Institute for Embedded Intelligence, Ireland

Abstract

Artificial intelligence applications are increasingly expected to operate directly on smartphones, embedded controllers, sensor platforms, robots, and other edge devices. Conventional deep neural networks can provide strong predictive performance but often require substantial memory, computation, and energy. These requirements can make cloud-independent inference difficult on hardware with restricted processing capacity. Lightweight neural-network design therefore focuses on reducing parameters and arithmetic operations while retaining sufficient predictive accuracy for practical on-device artificial intelligence. This hypothetical methodological study compares alternative lightweight convolutional architectures for image classification under an edge-computing scenario. A fictional dataset containing 12,000 images across twelve recognition categories is divided into training, validation, and testing subsets. Four conceptual architectures are examined: a conventional convolutional network, a SqueezeNet-style compact model, a MobileNet-style depthwise-separable network, and a ShuffleNet-style grouped-convolution architecture. Models are evaluated using classification accuracy, parameter count, model size, multiply-accumulate operations, and hypothetical inference latency.

The illustrative analysis indicates that the conventional network achieves competitive classification accuracy but requires substantially greater storage and computation. The MobileNet-style model provides the strongest hypothetical accuracy–latency balance, whereas the ShuffleNet-style architecture records the lowest latency and computational requirement. A second analysis examines 8-bit quantization and structured pruning of the MobileNet-style model and demonstrates further reductions in model size, latency, and estimated inference energy with a limited illustrative reduction in classification accuracy. The study concludes that efficient edge artificial intelligence requires joint consideration of predictive accuracy, computational operations, memory consumption, latency, and energy rather than optimization of accuracy alone. Architecture-level efficiency and post-training compression can provide complementary benefits. However, actual deployment performance depends on hardware architecture, runtime implementation, memory access, operator support, thermal constraints, and application-specific latency requirements.

Keywords: Edge Artificial Intelligence, Lightweight Neural Networks, MobileNet, SqueezeNet, ShuffleNet, Model Compression, Quantization, Embedded AI, Efficient Deep Learning, Edge Computing.

1. Introduction

Deep neural networks have achieved substantial performance improvements across image recognition, speech processing, sensing, and intelligent automation. Many successful architectures achieve this performance by increasing network depth, channel width, and representational capacity. Such growth also increases parameter storage, arithmetic operations, memory movement, and energy consumption. These costs are comparatively manageable in data-center environments equipped with specialized accelerators but can become restrictive when artificial intelligence must operate on resource-constrained edge devices.

Edge artificial intelligence refers broadly to performing machine-learning inference close to the location where data are generated rather than transferring every observation to a remote computational service. Local inference can reduce communication dependence and support applications requiring low response latency or operation under intermittent connectivity. However, edge devices typically provide substantially less memory and compute capacity than high-performance servers. Neural-network architecture must therefore be designed according to resource constraints rather than selected exclusively according to benchmark accuracy.

Earlier convolutional architectures demonstrated that increasing model capacity could substantially improve image classification. AlexNet accelerated the adoption of deep convolutional models, while VGG networks demonstrated the effectiveness of deeper stacks of small convolutional filters. GoogLeNet introduced more computationally efficient multi-branch modules, and ResNet enabled significantly deeper architectures through residual connections. These developments established strong visual representations but also highlighted the growing computational requirements associated with modern deep learning.

Model-compression research addressed this problem by reducing redundancy after or during network training. Han et al. developed pruning and compression approaches that eliminated unnecessary connections and reduced numerical precision. Iandola et al. proposed SqueezeNet, which achieved strong classification performance with dramatically fewer parameters than larger architectures. Rastegari et al. explored binary weights and activations, demonstrating the potential of extremely low-precision computation for reducing memory and arithmetic requirements.

Architecture-level efficiency subsequently became a major research direction. MobileNet used depthwise-separable convolutions to replace computationally expensive standard convolutions with factorized operations. ShuffleNet combined pointwise group convolution with channel shuffling, while MobileNetV2 introduced inverted residuals and linear bottlenecks. The present hypothetical study compares representative lightweight design strategies and examines whether additional quantization and pruning can further improve suitability for constrained edge inference.

2. Objectives of the Study

The principal objective of this study is to examine lightweight neural-network architectures for efficient artificial intelligence inference on resource-constrained edge devices. The investigation focuses on architecture-level computation reduction, accuracy–efficiency trade-offs, model compression, inference latency, and methodological considerations affecting practical edge deployment.

2.1 Examination of Lightweight Architectural Strategies

The first objective is to examine how different architectural mechanisms reduce neural-network complexity. SqueezeNet-style models reduce parameter requirements through compact squeeze-and-

expand modules. MobileNet-style networks replace standard convolution with depthwise and pointwise operations. ShuffleNet-style networks additionally reduce pointwise-convolution cost through grouped computation and channel rearrangement. These strategies address computational efficiency through structural network design rather than merely compressing a conventional model after training.

The study treats parameter count and operation count as related but distinct characteristics. A model can contain relatively few parameters while still requiring substantial computation when large feature maps are processed repeatedly. Conversely, a network with moderate parameter storage may execute efficiently when operations map effectively to the underlying hardware. Architectural efficiency should therefore be evaluated using several resource indicators rather than relying solely on model file size.

2.2 Evaluation of Accuracy and Edge Inference Efficiency

The second objective is to compare the classification performance and computational characteristics of the selected architectures using a common hypothetical dataset and edge-device platform. Overall accuracy is examined together with parameter count, storage requirement, multiply-accumulate operations, and inference latency. This multidimensional evaluation is necessary because an architecture producing the highest accuracy may not provide the most appropriate deployment solution when latency or energy constraints are strict.

The study also considers actual latency conceptually rather than assuming that theoretical operation counts completely determine runtime. Memory access, operator implementation, cache behavior, parallelism, and accelerator support can influence measured execution time. Two architectures with similar mathematical complexity can therefore produce different latency on a particular processor. Edge-AI evaluation should consequently combine analytical complexity with device-level benchmarking.

2.3 Assessment of Compression and Quantization

The third objective is to examine whether post-training optimization can further improve an already lightweight architecture. The conceptual MobileNet-style model is evaluated under full-precision operation, 8-bit quantization, and combined structured pruning with 8-bit quantization. Quantization decreases numerical precision, while structured pruning removes complete computational units such as channels or filters to reduce execution requirements in a hardware-friendly manner.

The objective is to determine how these transformations alter model storage, latency, estimated energy consumption, and predictive accuracy simultaneously. Compression is not considered successful merely because a model file becomes smaller. For edge deployment, the optimized network should also produce practical runtime improvements while limiting degradation in application accuracy. The relationship between compression and physical-device performance therefore represents a central evaluation requirement.

3. Review of Literature

3.1 Compact Neural Networks and Model Compression

Deep neural networks contain substantial redundancy, motivating research into compression techniques that preserve predictive performance while reducing storage. Han et al. demonstrated that pruning can eliminate unnecessary network connections and subsequently developed Deep Compression, combining

pruning, trained quantization, and entropy coding. Their results established that conventional high-capacity networks can often be compressed substantially without equivalent reductions in predictive accuracy.

Iandola et al. approached efficiency through architecture design rather than only post-training modification. SqueezeNet used Fire modules containing squeeze and expand operations to reduce parameters substantially. Other studies explored low-rank approximations and structured decomposition of convolutional operations. These approaches collectively demonstrated that efficient deep learning can be achieved through both architectural design and transformation of previously trained networks.

3.2 Low-Precision and Efficient Computation

Numerical precision represents another major source of computational cost. Courbariaux et al. investigated networks using binary weights and activations, while Rastegari et al. developed XNOR-Net for binary convolutional computation. Such methods can substantially reduce memory and replace expensive arithmetic with simpler operations, although classification accuracy may decrease depending on the architecture and task.

Jacob et al. later developed integer-arithmetic-only inference using 8-bit quantization and demonstrated that reduced precision can provide practical efficiency on mobile processors. Quantization is particularly relevant to edge computing because many mobile and embedded processors include optimized integer arithmetic. However, actual benefits depend on runtime support, quantization calibration, layer sensitivity, and whether intermediate activations can also be represented efficiently.

3.3 Mobile-Oriented Neural Architectures

Howard et al. introduced MobileNet as a family of efficient networks based on depthwise-separable convolution. Standard convolution simultaneously performs spatial filtering and cross-channel combination, whereas depthwise-separable convolution divides these operations into lightweight depthwise filtering followed by pointwise channel mixing. Width and resolution multipliers further enable adjustment of model complexity according to available resources.

ShuffleNet subsequently introduced pointwise group convolution and channel shuffle for extremely restricted computational budgets. MobileNetV2 developed inverted residual blocks with linear bottlenecks to improve representational efficiency and reduce intermediate-memory requirements. CondenseNet explored learned group convolutions, while EfficientNet later investigated systematic scaling of depth, width, and resolution. Together, these studies demonstrate that efficiency must be considered during architectural construction rather than only after a large network has been trained.

4. Research Methodology

4.1 Research Design and Neural Architectures

The present study adopts a hypothetical comparative image-classification design involving four neural-network architectures. The baseline network contains conventional convolutional blocks followed by global pooling and a classification layer. The SqueezeNet-style network uses squeeze-and-expand modules, the MobileNet-style network uses depthwise-separable convolution, and the ShuffleNet-style architecture combines grouped pointwise operations with channel shuffling. All models are conceptually trained using identical image partitions.

The experimental comparison holds input resolution, class labels, augmentation, optimizer family, and evaluation procedure constant. Hyperparameters are assumed to be selected using validation observations only. Model parameter count and multiply-accumulate operations are computed conceptually from each architecture. Inference latency is assigned according to a hypothetical ARM-based edge platform using batch size one, representing a latency-sensitive local inference scenario rather than high-throughput server processing.

4.2 Hypothetical Image Dataset and Edge Platform

The fictional dataset contains 12,000 RGB images representing twelve general visual-recognition categories. Eight thousand four hundred images are assigned to training, 1,800 to validation, and 1,800 to independent testing. Images are normalized to 224×224 pixels before inference. Standard augmentation includes random cropping, horizontal reflection, moderate brightness variation, and limited geometric transformation. No image corresponds to an identified real person or proprietary dataset.

The conceptual edge platform contains a four-core ARM-based processor, 4 GB system memory, and an embedded neural-computation runtime capable of floating-point and 8-bit integer operations. The platform contains no discrete desktop-class GPU. Latency represents single-image inference after model loading and warm-up. Energy values used later are hypothetical estimates for comparative illustration rather than measurements from physical instrumentation.

4.3 Performance and Efficiency Metrics

Predictive performance is evaluated using top-1 classification accuracy. Computational complexity is represented through model parameter count and multiply-accumulate operations per image. Storage footprint represents the approximate model size under the stated numerical representation. Inference latency represents hypothetical device-level execution time per image. Lower parameter count, model size, operation count, and latency indicate greater resource efficiency when predictive accuracy remains sufficient.

The second analytical stage applies 8-bit quantization and structured channel pruning to the MobileNet-style architecture. Quantization converts 32-bit floating-point weights and eligible activations to 8-bit integer representations. Structured pruning conceptually removes low-contribution channels followed by fine-tuning. Accuracy, model size, inference latency, and estimated energy per inference are then compared across optimization levels.

5. Analysis

5.1 Comparative Architecture Efficiency

The first hypothetical analysis demonstrates substantial differences in computational requirements among the four architectures. The conventional CNN achieves an assigned accuracy of 89.6% but contains 12.4 million parameters and requires approximately 820 million multiply-accumulate operations per image. The SqueezeNet-style architecture reduces storage substantially but retains a comparatively moderate computation requirement. MobileNet-style and ShuffleNet-style architectures provide greater reductions in arithmetic operations.

The MobileNet-style configuration produces the strongest hypothetical accuracy at 90.4% while requiring 280 million operations and 31 milliseconds per image. The ShuffleNet-style model achieves

slightly lower accuracy at 89.9% but records the lowest computation requirement and latency. These results demonstrate that architecture selection depends on deployment priorities. Where maximum speed is critical, the lowest-latency model may be preferable even when another model provides a small accuracy advantage.

Table 1: Hypothetical Accuracy and Efficiency of Lightweight Neural Architectures

Architecture	Parameters (Millions)	Model Size (MB)	MACs per Image (Millions)	Inference Latency (ms)	Accuracy (%)
Conventional CNN	12.4	49.6	820	92	89.6
SqueezeNet-Style Network	1.3	5.2	360	54	87.8
MobileNet-Style Network	3.4	13.6	280	31	90.4
ShuffleNet-Style Network	1.8	7.2	145	24	89.9

Note: Architecture measurements and classification results are illustrative and were not obtained from physical hardware experiments.

5.2 Effects of Quantization and Structured Pruning

The second hypothetical analysis applies model optimization to the MobileNet-style configuration. Converting the network from 32-bit floating-point representation to 8-bit integer operation reduces the illustrative model footprint from 13.6 MB to 3.5 MB and decreases inference latency from 31 to 20 milliseconds. Accuracy declines slightly from 90.4% to 89.8%. The estimated energy per inference decreases from 410 to 270 millijoules.

Combining structured pruning with 8-bit quantization produces a further reduction to 2.8 MB and hypothetical latency of eighteen milliseconds. Accuracy decreases to 89.2%, representing a 1.2-percentage-point reduction relative to the original full-precision configuration. The example illustrates that increasingly aggressive optimization can provide diminishing efficiency gains alongside larger predictive degradation.

Table 2: Hypothetical Optimization of the MobileNet-Style Edge Model

Optimization Configuration	Model Size (MB)	Inference Latency (ms)	Estimated Energy/Inference (mJ)	Accuracy (%)
FP32 Lightweight Baseline	13.6	31	410	90.4

INT8 Quantization	3.5	20	270	89.8
Structured Pruning + INT8 Quantization	2.8	18	240	89.2

Note: Energy, latency, size, and accuracy values are fictional and do not describe a tested commercial edge device.

6. Discussion

6.1 Architectural Efficiency and Accuracy Trade-Off

The hypothetical findings illustrate that lightweight architecture design can substantially reduce computational requirements without necessarily causing proportionally large decreases in classification accuracy. SqueezeNet demonstrates parameter reduction through compact modules, while MobileNet and ShuffleNet directly reduce convolutional computation. These strategies address different bottlenecks and therefore produce different relationships among storage, operation count, and runtime latency.

The assigned MobileNet-style result demonstrates a favorable balance between classification accuracy and device latency, while ShuffleNet-style processing provides the most aggressive computational reduction. Such results should not be interpreted as universal rankings. Actual performance depends on framework implementation, processor architecture, available vector instructions, memory bandwidth, and whether individual network operators have optimized hardware kernels.

6.2 Depthwise, Grouped, and Bottleneck Operations

Depthwise-separable convolution reduces the computational burden of standard convolution by separating spatial filtering from channel mixing. This design forms the basis of MobileNet and later MobileNetV2. The latter additionally uses inverted residual structures and narrow linear bottlenecks to preserve useful information while controlling intermediate memory. These structural changes are particularly relevant for edge processors where memory movement can represent a substantial component of inference cost.

ShuffleNet uses a different strategy by dividing expensive pointwise convolution into groups and shuffling channels to permit information exchange across groups. This design demonstrates that theoretical efficiency can be increased through sparse connectivity patterns. However, fragmented operators can sometimes execute inefficiently on hardware despite reduced operation counts. Architecture design should therefore consider both mathematical complexity and actual runtime characteristics.

6.3 Compression as a Complement to Lightweight Design

Deep Compression demonstrated that pruning and reduced-precision representation can remove substantial redundancy from conventional networks. Similar techniques can also be applied to architectures that were already designed for efficiency. The hypothetical second analysis illustrates this complementary relationship: architecture design first reduces computation, and quantization or pruning subsequently provides additional storage and latency reduction.

Compression does not always translate directly into hardware acceleration. Unstructured pruning can generate irregular sparsity that conventional processors cannot exploit efficiently. Structured pruning is therefore attractive because removing complete channels or filters can reduce dense tensor dimensions without requiring specialized sparse-computation support. Quantization similarly provides the greatest benefit when the target runtime includes optimized low-precision kernels.

7. Challenges

7.1 Hardware-Dependent Inference Performance

One of the most important challenges in edge-AI research is that theoretical operation counts do not perfectly predict actual runtime. Cache behavior, memory bandwidth, processor instruction sets, thread scheduling, and kernel implementation can substantially alter latency. A network containing fewer multiply-accumulate operations can occasionally execute more slowly if its operators are fragmented or poorly optimized for the target processor.

Research should therefore benchmark models directly on intended hardware rather than report only parameter count or FLOPs. Warm-up behavior, batch size, processor frequency, thread count, runtime library, and thermal conditions should be documented. Without these details, latency comparisons may be difficult to reproduce and may not transfer to another edge platform.

7.2 Accuracy Loss During Compression

Quantization and pruning can reduce representation capacity and therefore decrease predictive accuracy. Some layers are more sensitive to numerical precision or channel removal than others. Aggressive optimization can particularly affect difficult or infrequent categories even when aggregate accuracy remains relatively stable. Model compression should consequently be evaluated using class-level metrics as well as overall performance.

Quantization-aware training can mitigate some accuracy loss by exposing the model to low-precision effects during optimization. Structured pruning can similarly be followed by fine-tuning so that remaining parameters adapt to the reduced architecture. Nevertheless, the optimal compression level remains task dependent. A safety-critical inference application may tolerate substantially less predictive degradation than a low-risk consumer application.

7.3 Energy, Memory, and Thermal Constraints

Edge inference consumes energy through arithmetic operations, memory access, and data movement. Parameter reduction can decrease memory traffic, while lower precision can reduce both storage and computational energy. However, power consumption is influenced by the entire hardware and software stack. Energy measurements should therefore be obtained through appropriate device-level instrumentation rather than estimated only from theoretical operations.

Sustained inference can also create thermal throttling. A model that meets latency requirements during a short benchmark may slow after continuous operation because the device reduces processor frequency to control temperature. Edge-AI evaluation should consequently consider sustained workloads, available battery capacity, memory use, and thermal behavior in addition to one-time inference speed.

8. Conclusion

Lightweight neural-network architectures provide an important pathway for deploying artificial intelligence on devices with constrained computation, memory, and energy. The present hypothetical study compared a conventional CNN with SqueezeNet-style, MobileNet-style, and ShuffleNet-style architectures using a common fictional classification task and conceptual ARM-based edge platform.

The illustrative analysis assigned the strongest accuracy–latency balance to the MobileNet-style network and the lowest computational requirement to the ShuffleNet-style configuration. Additional 8-bit quantization and structured pruning further reduced the hypothetical MobileNet-style model footprint, latency, and energy consumption with a moderate reduction in classification accuracy. Every numerical result is fictional and does not constitute empirical benchmarking evidence.

The study concludes that edge-AI model selection should be based on a multidimensional efficiency profile rather than accuracy alone. Parameter count, operation count, memory footprint, measured latency, energy consumption, hardware support, and sustained operating conditions are all relevant. Future empirical research should therefore benchmark optimized architectures across multiple physical edge platforms and evaluate whether theoretical efficiency improvements translate consistently into real deployment benefits.

References

1. Benmeziane, H., El Maghraoui, K., Ouarnoughi, H., Niar, S., Wistuba, M., & Wang, N. (2021). Hardware-aware neural architecture search: Survey and taxonomy. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence* (pp. 4322–4329). <https://doi.org/10.24963/ijcai.2021/592>
2. Cai, H., Zhu, L., & Han, S. (2020). ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations (ICLR)*. <https://arxiv.org/abs/1812.00332>
3. Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2020). Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*. <https://arxiv.org/abs/1908.09791>
4. David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Wang, T., Warden, P., & Rhodes, R. (2021). TensorFlow Lite Micro: Embedded machine learning for TinyML systems. In *Proceedings of Machine Learning and Systems*, 3, 800–811. https://proceedings.mlsys.org/paper_files/paper/2021/hash/9a05a6f9d9a5f3c9b5f6e8c7a8e5d2a8-Abstract.html
5. Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations (ICLR)*. <https://arxiv.org/abs/1510.00149>
6. Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. In *Proceedings of the 43rd Annual International Symposium on Computer Architecture* (pp. 243–254). Association for Computing Machinery. <https://doi.org/10.1145/3007787.3001139>
7. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv*. <https://arxiv.org/abs/1704.04861>

8. Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., & Adam, H. (2019). Searching for MobileNetV3. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1314–1324). <https://doi.org/10.1109/ICCV.2019.00140>
9. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and less than 0.5 MB model size. arXiv. <https://arxiv.org/abs/1602.07360>
10. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2704–2713). <https://doi.org/10.1109/CVPR.2018.00286>
11. Lin, J., Chen, W.-M., Lin, Y., Cohn, J., Gan, C., & Han, S. (2020). MCUNet: Tiny deep learning on IoT devices. In Advances in Neural Information Processing Systems, 33, 11711–11722. <https://arxiv.org/abs/2007.10319>
12. Ma, N., Zhang, X., Zheng, H.-T., & Sun, J. (2018). ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In Proceedings of the European Conference on Computer Vision (pp. 116–131). https://doi.org/10.1007/978-3-030-01264-9_8
13. Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). XNOR-Net: ImageNet classification using binary convolutional neural networks. In Computer Vision – ECCV 2016 (pp. 525–542). Springer. https://doi.org/10.1007/978-3-319-46493-0_32
14. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4510–4520). <https://doi.org/10.1109/CVPR.2018.00474>
15. Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Proceedings of the 36th International Conference on Machine Learning (pp. 6105–6114). PMLR. <https://proceedings.mlr.press/v97/tan19a.html>
16. Warden, P. (2019). TinyML: Machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers. O'Reilly Media.
17. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., & Keutzer, K. (2019). FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10734–10742). <https://doi.org/10.1109/CVPR.2019.01099>
18. Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6848–6856). <https://doi.org/10.1109/CVPR.2018.00716>
19. Zhang, Y., Qin, Y., Zhang, Y., Zhou, X., Jian, S., Tan, Y., & Li, K. (2024). OnceNAS: Discovering efficient on-device inference neural networks for edge devices. Information Sciences, 669, Article 120567. <https://doi.org/10.1016/j.ins.2024.120567>
20. Efficient neural networks for edge devices. (2021). Computers & Electrical Engineering, 92, Article 107121. <https://doi.org/10.1016/j.compeleceng.2021.107121>