5

Designing Lightweight CNN for Images: Architectural Components and Techniques

Lilian Hollard, Lucas Mohimont, and Luiz Angelo Steffenel

Université de Reims Champagne-Ardenne, France

Abstract

While neural networks have brought about impressive advancements in computer vision tasks, these achievements heavily depend on computationally demanding resources, restricting their deployment. The decentralized paradigm of Edge AI computing aims to bring decisional capabilities directly to the edge, facilitating real-time decision-making, streamlined data processing, and reduced dependence on network connectivity. In some cases, it is possible to rely on cloud computing to offload processing tasks, but this can introduce latency issues that affect system responsiveness, security, and efficiency. Instead, searching for optimized neural networks for edge device deployment may lead to a better balance between computational efficiency and accurate analysis, empowering sensors to execute their roles effectively with minimal reliance on external resources. This paper reviews the landscape of deep learning architecture optimization tailored for edge devices. Within this survey, we delve into the state-of-the-art advancements in computer vision techniques optimized for edge computing. The challenges deploying and optimizing computer vision models on edge devices emphasize the importance of efficient computation and resource management while navigating the trade-offs between model performance and hardware constraints.

Keywords: neural network architecture, Edge AI, deep learning, neural architecture search, transformers, Edge vision, computer vision.

5.1 Introduction and Background

Neural networks enabled significant advancements in computer vision. However, these achievements often rely on computationally expensive resources, limiting deployment on less powerful devices. Despite the rapid adoption of cloud-based processing and cloud AI over the last decade, such offloading brings several inconveniences such as latency, bandwidth limitations, and security concerns. These challenges led to the development of Edge AI, which stands to the AI landscape. Edge Computing offers notable advantages, such as data ownership, heightened security, reduced latency, and decreased power consumption attributed to minimized back-and-forth communication with the cloud.

The term “Edge” encompasses a wide spectrum of devices and applications, including peripheral data centres (cloudlets [1], fog [2]) and IoT endpoints. Hence, it is not uncommon to partition the Edge according to the capabilities of the devices or the distance to end users. For example, the EdgeAI European project classifies edge solutions in three levels: Meta, Deep and Micro-Edge. Micro Edge represents mostly the final sensors and devices, Deep Edge lies in the vicinity (gateways, network routers) and Meta-Edge interfaces the Edge with external technologies such as the cloud. As a result, the size and capability of devices are determinant factors that differentiate various “Edge” application areas within Edge AI.

Bringing computer vision tasks to Micro-Edge devices such as microcontrollers is often complex due to resource limitations and computational constraints. These devices may struggle with the intensive processing demands of computer vision algorithms, making it difficult to perform analysis and decision-making directly at the edge. Offloading computer-vision tasks to “upper” layers is also a problem, as the data volume to be transmitted is far over more traditional IoT sensor data. Instead, Edge-AI computer vision requires optimized solutions adapted to the resource constraints of edge devices.

In this chapter, we review the most recent advances in computer vision methodologies for edge computing, with a specific emphasis on model architecture. While various established techniques such as quantization, pruning, and hardware optimization have been extensively investigated, our primary focus is the substantial enhancements that deep learning model architecture has witnessed over the last few decades. These enhancements have notably contributed to the improvement of Edge-AI. We explore the challenges faced in deploying and optimizing computer vision models on edge devices, the need for efficient computation and resource management, and the trade-offs between model performance and hardware constraints.

Furthermore, we run our own benchmarks to obtain uniform comparison results. Indeed, the deployment of deep learning models must emphasize model efficiency and comparison across various parameters. Metrics such as inference time, latency, training and inference costs, and other established indicators are crucial for researchers to demonstrate the contribution of new deep-learning techniques. However, researchers often assume a correlation among these metrics and report only a few of them, leading to partial conclusions and incomplete evaluations of different models [3].

Considering the types of models, different computational aspects may yield varying results. One example of bias in deep learning model optimization is relying solely on parameter-matched comparisons as a single metric, which may result in a flawed understanding of overall model performance. Shift-based convolution for instance, improves overall accuracy by offering a parameter-free alternative to traditional convolution but increases processing times. Memory access costs on different platforms or overall, unsatisfactorily optimized multi-branch model architecture for parallel computing might as well influence speeds metrics [3][7][8]. Therefore, models should evaluate multiple metrics on the targeted platform, as memory access and model parallelization are architecture dependent.

For the sake of reproducibility, most benchmarks presented in this chapter originate from the PyTorch Torchvision module benchmark. Our intention is to enable readers to replicate the results, although some models may exhibit slight variations from their original paper.

The remainder of the chapter is structured as follows: Section 1.2 explores the latest advancements in the computer vision research community using convolutional neural networks. Section 1.3 examines how Transformers revolutionized computer vision and how these techniques can be employed to reduce the overall computational cost. Section 1.4 investigate ConvNeXt convolution and its potential to elevate CNN models, enabling them to rival Transformers, while also exploring their utility in Edge computation. Section 1.5 covers the neural architecture search in an efficient computation scenario. We conclude this paper in Section 1.6, presenting some final remarks and research directions.

5.2 CNNs

Convolutional Neural Networks (CNN) are a class of deep learning models that excel when processing and analysing visual data. CNN enhanced the ability to learn intricate patterns and features from massive datasets, empowering deep learning to achieve remarkable breakthroughs in diverse areas, including computer vision but also natural language processing, speech recognition, recommendation systems and many more. The scalability, adaptability, and robustness of CNNs make it a dominant force in the breakout of AI technologies.

5.2.1 The pioneers

Optimising CNNs for low-power applications often involves weight pruning, quantisation, and model compression techniques. In theory, these methods position CNNs as ideal solutions for edge devices operating in energy-constrained environments; however, although essential for edge devices, these techniques often reduce accuracy. We cannot deny that the computer vision community made substantial progress since the remarkable performance of AlexNet's [9] first publication. Through architectural changes and optimisations, significant performance improvements extended several times state-of-the-art computer vision models while reducing computational demands.

ResNet [10] revolutionised the landscape of neural networks by introducing the concept of residual connections, a breakthrough that facilitated the construction of exceptionally deep models. This innovation proved instrumental in optimising the training of deeper layers. As a result, these extended CNN architectures attained unparalleled performance across diverse benchmark tasks.

ResNet's pioneering influence shaped the field of Deep Learning, serving as a cornerstone that inspired the architectural design of countless contemporary models. Residual connections provide an alternative pathway for the gradient to flow during backpropagation to address the vanishing gradient problem, as illustrated in Figure 5.1.

However, while many high-performing CNN models characterised by substantial numbers of parameters and FLOPS (Floating point operations per second) achieved impressive performance, the realm of Lightweight CNNs emerged as a potent contender. These efficient architectures, including EfficientNet [11][13], MobileNets [14][15], ShuffleNet [16][17], SqueezeNet [18], and ESPNet [19][20], devoted remarkable efforts to optimise CNNs by

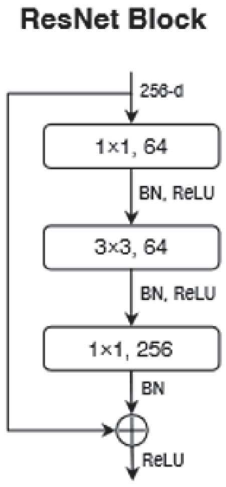

Figure 5.1 ResNet architecture [11]

renouncing the need for excessively deep and densely interconnected structures, aligning with the philosophy that ResNet and VGG [21] established. The success of these Lightweight CNNs highlights their ability to achieve competitive performance while maintaining a judicious balance between model complexity, parameters, and FLOPS.

CNNs are widely utilised across various domains, such as classification, object detection, segmentation, and other tasks. Currently, the object detection and segmentation research communities closely collaborate to enhance classification CNNs and the other way around, recognising their pivotal role as the backbone for object detection and segmentation models. Hence, the forthcoming sections will not specifically address these distinctions, as they all contribute to improving EdgeAI capabilities.

5.2.2 YOLO, first step towards fast object detectors

YOLO (You Only Look Once) [22], a deep learning model introduced in 2016, revolutionised object detection by providing real-time detection and accurate results. Before YOLO, most object detection algorithms followed a two-step approach, which was time-consuming and limited in case of detection speed. On the other hand, YOLO formulated object detection as a regression problem, dividing the input image into a grid and predicting bounding boxes and class probabilities directly from the grid.

Newer versions of YOLO [23][24][25][26][27][28][29][30] incorporated concepts like anchor boxes, feature pyramid networks, and advanced network architectures (e.g., Darknet, ResNet) to improve detection accuracy and handle objects of various sizes. They also introduced multi-scale predictions, enabling detection at different resolutions within the network. However, these advanced network architectures, such as YOLOv5, have high memory requirements and need relatively powerful edge devices like the Jetson Nano with Nvidia GPU.

To circumvent such requirements, the object detection research community has three major perspectives. First, the improvement of YOLO-based models, both in terms of complex computation (e.g., YOLOv7 [25]) and adapting YOLO for edge computation (e.g., TinyssimiYOLO [31], YOLOv3 Tiny[26], YOLOv5 Nano [23], YoloNAS [32]). Second, the enhancement of backbones for object detection, such as combining MobileNetV3 [33] with SSD320 [34] detection heads. One last approach is the exploration of Transformers-based object detectors.

The miniaturisation of YOLO models is still ongoing research. Recent efforts have focused on reducing the architecture to create highly flexible, memory-efficient, and ultra-lightweight object detection networks with less than 0.5MB of memory. However, these optimised models are most suitable for detecting a few classes. For example, TinyissimiYOLO [31] performs well for up to three classes, and challenges remain when trying to improve the edge-oriented benchmark on datasets like MS COCO [35], which consists of 80 classes. As demonstrated in TinyissimoYOLO, a plain-architecture model still exhibits great potential for efficient inference on microcontrollers or edge devices.

In addition to YOLO object detectors, there is a significant effort to enhance classifier CNNs, which extend to backbone CNNs for object detector models in a broader context. These CNN architectures undergo rigorous benchmarking on datasets like ImageNet but also on datasets such as MSCOCO and Pascal VOC to address object detection and segmentation tasks. Indeed, the emergence of SSD detector heads and Mask R-CNN segmentation heads catalysed a distinct research avenue, prompting a concentrated exploration of classification models or backbone designs specifically tailored for advancing object detection capabilities.

Research to enhance classifiers changed the recent panel of YOLO models. Since YOLOv4 [30], the composition of YOLO models depends on CSPNet [36] block modifications, an architecture that already enabled known architectures such as ResNet, ResNeXt, and DenseNet to reduce

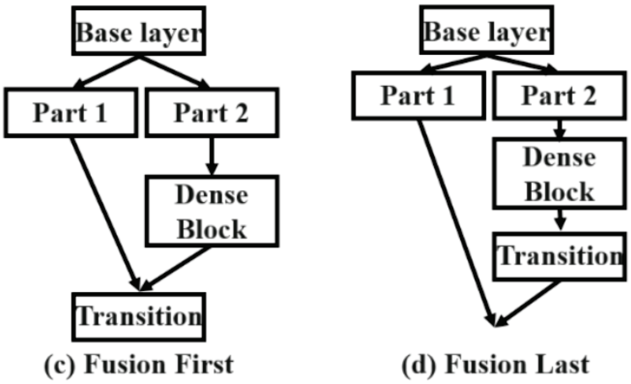

Figure 5.2 CSPNet (Identity Block - DenseNet)

computational cost while preserving accuracy. It effectively reduces computational bottlenecks (YOLOv3's computational bottlenecks can be reduced by 80%) and memory costs. ResNeXt already proved that cardinality can be more efficient than width and height. CSPNet divides feature maps into two main parts: one used to create an identity block (DenseNet, ResNet, MobileNet, etc.) and the other that is combined at the output after or before a transition layer, as shown in Figure 5.2.

5.2.3 Convolutional Neural Network architecture improvements

MobileNetv1 was one of the first CNN architectures specifically created to bring efficiency to mobile and embedded vision applications. Its main improvement was the efficient use of depthwise separable convolutions to build lightweight neural networks. MobileNet was nearly as accurate as VGG16, with 32 times less size and 27 times less computation. MobileNetv1 performance on the ImageNet dataset achieved a top-1 of 68.4%. MobileNetv2, on the other hand, improved MobileNetv1 drastically while preserving the same mobile-first philosophy, using inverted residual block and linear bottlenecks. MobileNetv2's linear bottleneck does not incorporate linear activation within its narrow input and output layer. Instead, it incorporates non-linearity after each expanded layer of the bottleneck.

The hypothesis of MobileNetv2 stated that ReLU can preserve complete information only if the i-th feature input lies in a low-dimensional subspace of the input space. Researchers showed through experimental evidence that using non-linear layers in the input/output of bottlenecks impacts the

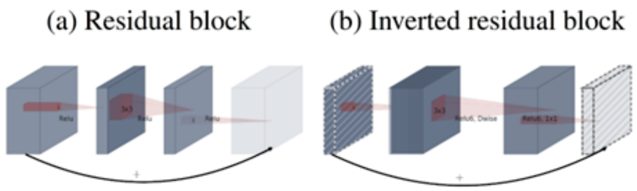

Figure 5.3 MobileNetv2 block

model's performance by several percent. Figure 5.3 illustrates the difference between non-linearity in the residual and inverted residual blocks. Residual block architecture skip connections with fewer feature maps between connections. In contrast, an inverted residual block broke this relation by using an expansion of feature maps. As a result, MobileNetV2 TOP-1 ImageNet performance reaches 71.978%, with only 2.6M parameters and 0.3G Flops. Table 5.1 details the performance of different MobileNet models when conducting the ImageNet classification.

Since MobileNet, a vast majority of modern networks adopted depth-wise separable convolutions. ShuffleNetv1 and v2 introduced practical guidelines for efficient network design, resolving some MobileNet issues.

ShuffleNet v2 stated that the expensive use of depth-wise separable convolutions and grouped convolution increase memory access cost. Also, element-wise operations have a high MAC (Memory Access Cost) and FLOP cost, even with a small parameter count. The ShuffleNet architecture thus introduces an architecture using balanced convolutions of equal channel width, reducing the degree of fragmentation and reduced element-wise operations, surpassing MobileNetv2 with an ImageNet top-1 performance of

Table 5.1 CNNs based model optimization since AlexNet

Model	ImageNet Top 1(%)	Parameters (M)	FLOPs (G)
ShuffleNetv2x0.5	60.5	1.3	0.04
ShuffleNetv2x1.0	69.36	2.27	0.14
MobileNetv3 Small	67.4	2.5	0.06
MobileNetv2	71.97	2.6	0.3
ShuffleNetv2x1.5	72.99	3.5	0.30
ShuffletNetv2x2.0	76.23	7.3	0.58
AlexNet	56.52	61	0.71
VGG16	76.3	132.8	7.61

72.99%, with comparable FLOPs of 0.3G and 3.5M parameters. Table 5.1 benchmarks ShuffletNet v2 across various scales.

5.2.4 Tackling memory consumption

Memory consumption is a significant concern when optimizing CNNs for mobile computing applications. State-of-the-art models for mobile and edge often employ grouped and depth-wise convolutions to reduce overall model parameters [37][14][15][39][16][17]. However, these models require more computation time and memory per layer, which may pose challenges for edge AI models focused on video stream processing.

Therefore, in addition to the architecture optimization research of CNNs, researchers made significant efforts to achieve extreme memory consumption optimization for microcontroller use cases. A remarkable example of this is MCUNet [40], which demonstrated impressive potential for memory optimization by simply enhancing the memory workflow of CNNs following the MobileNet architecture philosophy previously mentioned.

MCUNet significantly reduces memory usage for MobileNetv2, fitting it within a mere 320kB of RAM. This impressive feat is accomplished through two key strategies: first, by identifying the optimal input resolution size and adapting the model width to achieve the most efficient neural architecture size. Second, it leverages the characteristic of depth-wise convolutions, which do not perform filtering across channels, allowing each channel to be computed in a temporary buffer. This approach substantially reduces overall memory consumption, computing the input and the output feature map as one shared memory, with one additional buffer to compute and transfer the data. MCUNetv2 [41] goes further by optimizing memory usage through patch-based computation. Instead of processing the entire feature width and height, it strategically employs small input portions to generate activation maps, leading to more efficient memory utilization. Table 5.2 describes MobileNets memory consumption improvement with MCUNets.

5.2.5 Structural re-parameterization

Within this survey, we showcase research aimed at enhancing the architecture of models commonly referred to as “mobile”. However, these mobile-first models rely heavily on grouped and depth-wise convolutions, which induce many other computational challenges.

Table 5.2 MCUNet memory optimization compared to MobileNet and MobileNetv2

Model	**ImageNet Top 1 (%)**	**SRAM**
MobileNetv1	68.4	NS
MobileNetv2	69.8	1.8 MB
MCUNetv1-int8	60.3	238 kB
MCUNetv2-int8	64.90	**196 kB**
MCUNetv1-int8	68.5	452 kB
MCUNetv2-int8	**71.8**	465 kB

Grouped and depth-wise convolutions utilize 1x1 convolutions not well-optimized for certain architectures. In contrast, 3x3 convolution architectures are more efficient on generic GPUs than 1x1 convolutions. Multi-branch design models like ResNet [7] or branch-concatenation in Inception [42] encounter similar issues, making them less efficient for parallel architectures like GPUs due to additional overhead, such as kernel launching and synchronization. Residual connections also face challenges in retaining convoluted feature maps in memory during the computation of multi-branch deep learning architectures.

To address these challenges, recent research suggests structural re-parameterization to revert to early deep learning plain models like VGG [21], Darknet [22], and AlexNet [9], which are theoretically efficient for edge computation. However, these models no longer compete in terms of accuracy and overall performance with the current state-of-the-art models.

Residual connections and multi-branch architecture [43][44][42][36][39] [45][46], to cite only a few, are indeed essential components in deep learning architectures. Their introduction addresses the vanishing gradient problem, which occurs when gradients diminish as they propagate through deep networks during training.

It is challenging for a plain model to achieve comparable performance to a multi-branch architecture. The complex structure of multi-branch architectures often slows down inference, as the combination of small operators is not favourable for devices with strong parallel computing capabilities like GPUs. Taking a more edge-centric perspective, utilising multi-branch structures necessitates significant cache memory, as these structures demand the model to retain the feature maps of each branch in memory before processing to the subsequent layer. However, the benefits of multi-branch architecture mainly apply while training [47][7][46].

Structural re-parameterization involves transferring the knowledge gained from multi-branch architecture during training into a single plain convolution block for inference.

MobileOne TOP-1 ImageNet performance on multiple scaling is 71.4% and 75.9% for MobileOne-S0 and MobileOne-S1 respectively, both under 1G Flops. Table 5.2 lists the complete results. Other re-parametrization models, like RepVGG, accompany the comprehensive results in Table 5.3.

While the training cost may be significant, improving performance at the cost of additional training resources is acceptable if the deployed model fits the size and computing power required for edge devices. Hence, one

Table 5.3 CNNs based model optimization since AlexNet

Model	**ImageNet Top 1 (%)**	**Parameters (M)**	**FLOPs (G)**
AlexNet	56.52	61	0.71
EfficientNet B0	77.69	5.2	0.32
EfficientNet B1	78.64	7.79	0.69
EfficientNet B2	80.6	9.1	1.09
EfficientNet B3	82.2	12.2	1.83
EfficientNet B4	83.4	19.34	4.38
EfficientNet B7	84.122	66.34	37.75
EfficientNetv2 Large	85.808	118.5	56.08
EfficientNetv2 Medium	85.112	54.1	24.58
EfficientNetv2 Small	84.228	21.4	8.37
ESPNetv2	72.1	3.49	0.28
MobileNetv1	68.4	2.6	NS
MobileNetv2	71.97	2.6	0.3
MobileNetv3 Small	67.4	2.5	0.06
MobileOne-S0	71.4	2.1	0.275
MobileOne-S1	75.9	4.8	0.825
MobileOne-S2	77.4	7.8	1.29
MobileOne-S3	78.1	10.1	1.89
MobileOne-S4	79.4	14.8	2.97
RepVGG-A0	72.4	8.3	1.4
RepVGG-A1	74.5	12.8	2.4
RepVGG-B0	75.1	14.3	3.1
ResNet101	77.374	44.5	7.80
ResNet152	78.312	60.19	11.51
ResNet18	69.75	11.6	1.81
ResNet34	73.314	21.7	3.66
ResNet50	76.13	25.5	4.09
ShuffleNetv2x0.5	60.5	1.3	0.04
ShuffleNetv2x1.0	69.36	2.27	0.14
ShuffleNetv2x1.5	72.99	3.5	0.30
ShuffletNetv2x2.0	76.23	7.3	0.58
SqueezeNet	57.5	1.2	0.35
VGG16	76.3	132.8	7.61

can use a high-end GPU server for training before deployment on an edge device.

5.3 Transformers for EdgeAI

Initially proposed for natural language processing tasks, transformers have also found exploration in computer vision [48][49][50][51][52][53]. The Vision Transformer (ViT) [52], introduced in 2020, adapted the Transformer architecture specifically for computer vision by replacing CNN-based backbones with Transformer encoders. ViT achieved competitive performance on image classification benchmarks, indicating the effectiveness of Transformers in vision tasks. However, Transformers require substantial amounts of data to work well, and ViT struggles to perform on ImageNet-1K 0, which already contains over one million images. Convolution still plays an important role in Transformers.

5.3.1 Hybrid transformers

Research towards the miniaturization of Transformers focuses on utilizing CNNs as feature extractors with low computational cost. Models like MobileViT and its latest versions [37][50][49] achieved impressive performance with significantly fewer parameters and floating-point operations than previously published computer vision Transformers. The concept involves incorporating multiple CNN blocks within transformers to enhance the extraction of features, as illustrated in Figure 5.4. While transformers like ViT demonstrated remarkable capabilities in natural language processing tasks, they often lack the reliance on powerful feature extractors, such as CNNs, which may explain the challenges faced by such models in achieving efficient training with limited data, as seen in the struggle encountered

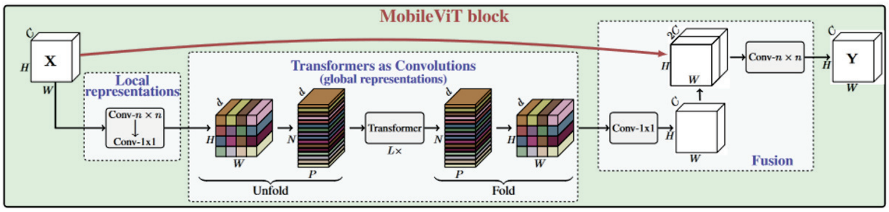

Figure 5.4 MobileViT block [37].

when training ViT based network on datasets like ImageNet1K. These contributions may lead to new backbones for object detection, segmentation, and classification in the computer vision community, utilizing these hybrid models.

As mentioned earlier, the number of parameters is not the sole metric for comparing results; a compelling example is the comparison between MobileNetv2 and its supposed enhancement, MobileViT. MobileViT fails to match the computing speed of MobileNetv2 on the iPhone 12 neural engine, as the latter achieves an inference time of 0.9ms, while MobileViT requires 7.28ms per inference despite having a similar number of parameters.

Transformers inherently tend to be slower than CNNs for several reasons. First, Vision Transformers (ViTs) are designed to leverage dedicated CUDA kernels on GPUs, enabling improved scalability and efficiency. In contrast, CNNs benefit from device-level optimizations like batch normalization fusion with convolutional layers, 3x3 convolution optimization, and other techniques. This observation suggests there is still room for improvement in optimizing transformers at a lower computational level. Despite their potential, transformers must continue to evolve to achieve faster and more efficient performance comparable to that of CNNs in specific contexts.

MobileViTv2's [50] enhancement primarily focused on optimizing the self-attention operation within transformers. As previously mentioned, researchers have a significant opportunity to improve attention layers like Multi-Head Attention (MHA). The computational complexity of the MHA layer is typically $O(k^2)$ whereas MobileViTv2's version of MHA has reduced it to $O(k)$ through the implementation of element-wise operations. The concept involves using element-wise operations such as summation, multiplication, and softmax instead of more computationally intensive operations like batch-wise matrix multiplication, which is quadratically expensive.

Previous efforts to optimize self-attention, such as the Linformer [37] approach, decompose the self-attention operation into smaller segments via linear projections, effectively reducing the complexity from $O(k^2)$ to $O(k)$. However, Linformer still employs resource-intensive operations to learn global representations within MHA, which could pose challenges for deploying such models on devices with limited resources. Other methods have managed to reduce complexity to $O(k)$ but often suffer substantial performance degradation.

In contrast, MobileViTv2 outperformed MobileViTv1 by approximately 1% and exhibited a significant speed boost, running 3.2 times faster on comparable devices. This advancement underscores the potential of optimizing self-attention operations within transformers while maintaining robust performance, especially in constrained computing environments.

In recent developments, there has been progress in enhancing hybrid architectures combining CNN and Transformers for mobile devices. MobileViTv3 [49] emerged as an improved iteration of the initial MobileViT architecture. This advancement involves substituting resource-intensive 3x3 convolutional layers with more efficient depthwise and 1x1 convolutions. Additionally, the integration of residual connections contributed to an overall performance boost for the MobileViT v1 design. Furthermore, this enhancement opened avenues for scaling the width of the MobileViTv3 model. The removal of the costly 3x3 convolutions led to a reduction in parameters and FLOPs, resulting in improved scalability while maintaining or enhancing performance.

Table 5.4 gives comprehensive results around each MobileViT version compared to Swin-T as an accuracy gap to achieve for the transformer-based model. While Swin Transformer [51] and DETR [48] significantly improved tackling the ImageNet classification and MS COCO challenges, they remain less suitable for edge computing due to their computational demands. Nevertheless, the technological advancements achieved through the rise of transformers could be instrumental in enhancing convolutions.

Table 5.4 Optimized transformers and hybrid transformers performance by scale

Models	**ImageNet Top 1 (%)**	**Parameters (M)**	**FLOPs (G)**
MobileViT-XXS	69.4	1.3	**0.4**
MobileViTv2-0.5	70.2	1.4	0.5
MobileViTv3-XXS	70.98	1.2	0.28
MobileViTv3-0.5	72.33	1.4	0.48
MobileViTv2-1.0	78.1	4.9	1.8
MobileViTv3-XS	76.7	2.5	0.92
MobileViTv3-1.0	**79.64**	5.1	1.87
MobileViTv3-S	79.3	5.8	1.84
MobileViTv2-0.5	70.2	1.4	**0.5**
MobiletViTv2-2.0	81.2	18.5	7.5
Swin-T	**81.3**	28.3	4.5
MobileNetv2	**71.978**	2.6	0.3

Consequently, using the benefits of the previously mentioned miniaturization techniques with these technological advancements in an edge-computing context becomes a promising avenue for further progress.

5.4 ConvNeXts

While Transformers are a significant breakthrough in computer vision, recent advancements, such as the ConvNeXt [11] convolution, quietly surpassed their performance in computer vision. The foundation of ConvNeXt lies in the adaptation of ResNet, which serves as a starting point, leveraging techniques inspired by transformers to fill the gap between ResNet's and Swin Transformer's performance (Figure 5.5).

To achieve this, ConvNeXt introduces a new CNN design inspired by Transformers. Firstly, it employs the patchify technique, which involves flattening the input into a vector of smaller patches, an idea initially introduced in Vision Transformers (ViT) to exploit text-based transformer techniques for processing 2D images effectively. Secondly, the training recipe aligns closely with the strategies employed in Swin Transformers and DeiT.

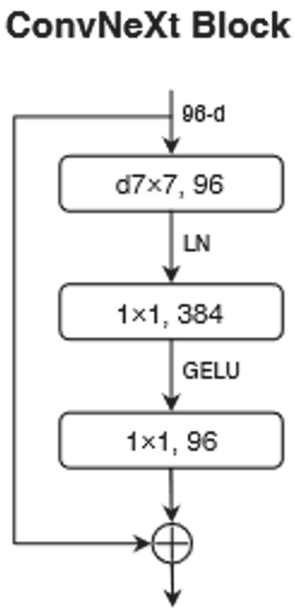

Figure 5.5 ConvNeXt Block [11]

Table 5.5 ConvNeXt compared to hybrid transformers performance by scale.

Models	ImageNet Top 1 (%)	Parameters (M)	FLOPs (G)
AlexNet	56.52	61	0.71
VGG16	76.3	132.8	7.61
ResNet152	78.312	60.19	11.51
EfficientNetv2 Large	85.808	118.5	56.08
MobiletViTv2-2.0	81.2	18.5	7.5
Swin-T	81.3	28.3	4.5
ConvNeXt-T	82.9	28.6	4.5
ConvNeXt-S	83.616	50.2	8.68
ConvNeXt-B	84.06	88.6	15.36
ConvNeXt-L	84.414	197	34.36

As a result, ConvNeXt exhibits less sensitivity to image shift-invariance. The training process is extensively enhanced through data augmentation, longer training epochs, and the AdamW optimizer, leading to improved results.

Furthermore, ConvNeXt introduces a compelling microdesign aspect, replacing the conventional ReLU activation function with GeLU (Gaussian Error Linear Unit), a different non-linear activation function. Additionally, the model incorporates fewer activations and norms than those found in transformer architecture (Recognizing the initial hypothesis involving non-linearity in MobileNetv2). These adaptations contribute to the model's overall efficiency and performance, effectively leveraging the power of transformer-inspired concepts within a CNN framework.

ConvNeXt's new philosophy for CNNs might open the path for new mobile architecture. The benchmark in Table 5.5 enlightens ConvNeXT performance compared to models with similar computing performance.

5.5 Neural Architecture Search

Amidst the collection of hand-crafted neural networks, the question arises: Can we venture into automatic network architecture design, to reduce the dependency on deep learning expert insights? The Neural Architecture Search (NAS) literature categorizes two primary domains: Evolutionary Algorithms (EA) and Reinforcement Learning (RL). Evolutionary algorithms utilize a pool of candidate architectures, each with its respective accuracy. Only a limited number of top-performing candidates evolve further. Should these evolved candidates exhibit enhanced accuracy, the candidate pool is accordingly updated. On the other hand, Reinforcement Learning (RL) employs an LSTM Agent to generate a string, serving as a dictionary of convolution operations to execute on hardware to train and test. The accuracy serves as the reward signal for this operation, and the LSTM Agent subsequently refines and produces another dictionary block.

The initial NASnet [54] model lacks consideration for runtime or computational efficiency. The search space for potential architectures is inherently resource intensive. While the LSTM Agent discovered architecture proves superior to manually crafted ones, it inherits the complexity identified in this survey as challenging for edge devices. This complexity arises from the substantial memory requirements due to the neural architecture search algorithm's reliance on a configuration of five cells per layer, each with three potential residual depth connections (from the previous cell's output, the cell before the previous one, and the previous block's output within the current cell).

Most NAS methods [54][55] explore architectural spaces to construct intricate cells, subsequently employing these cells with identical configurations throughout the network. Unfortunately, this approach lacks the potential for layer wise diversity. MnasNet [56], a breakthrough in Aware Neural Architecture Search for Mobile, introduces an edge computation model for inference by selecting a considerably smaller number of convolution blocks. This time, the LSTM agent selects hand-crafted architectures in alignment with the MobileNet philosophy. Employing such block-based designs reduces the search space from 10^{39} options to 10^{13}.

The search algorithm within MnasNet also introduces a multi-objective reward system that combines validation accuracy and a metric for real-world latency on mobile devices. This dual-objective approach optimization creates architectures that excel in both accuracy and efficiency for real-world performance.

5.5.1 NAS scale study

While NAS is primarily concerned with discovering novel convolutions based on accuracy-to-speed trade-offs, EfficientNet [12][13] takes a different approach, aiming to optimise a manually designed model by identifying the optimal balance among width, depth, and resolution (Figure 5.6). Although these components might seem independent, EfficientNet's case study highlights that achieving superior accuracy requires simultaneous optimization of all three components rather than considering them separately.

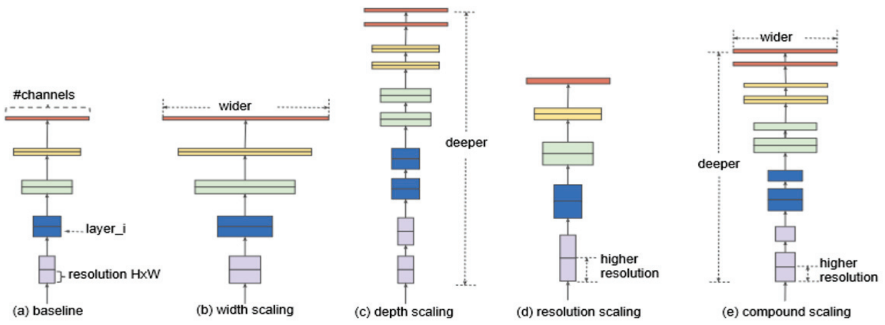

Figure 5.6 EfficientNet Scaling [12]

Table 5.6 Efficient neural network architectures with neural network search

Models	**ImageNet Top 1(%)**	**Parameters (M)**	**FLOPs (G)**
ResNet50	76.13	25.5	4.09
NASNet-A (4 @ 1056)	74	5.3	0.56
NASNet-B (4 @ 1536)	72.8	5.3	0.48
NASNet-C (3 @ 960)	72.5	4.9	0.558
MobileNetv1_efficientNetv1 (d=1.4,w=1.2,r=1.3)	75.6		2.3
MobileNetv2_efficientNetv1 (d=1.4, w=1.2, r=1.3)	77.4		1.3
PNASNet-5 (N = 3, F = 54)	74.2	5.1	0.58
PNASNet-5 (N = 4, F = 216)	82.9	86.1	25
EfficientNet_B0	77.692	5.2	0.32
EfficientNet_B1	78.642	7.79	0.69
EfficientNetv2_Small	84.228	21.4	8.37
EfficientNetv2_Medium	85.112	54.1	24.58
EfficientNetv2_Large	85.808	118.5	56.08
MobileNetv1	68.4	2.6	ND
MobileNetv2	71.978	2.6	0.3

To enhance accuracy and efficiency, the interplay between depth, width, and resolution is computed and then trained on the CIFAR dataset, allowing rapid assessment of the newly formulated model. Subsequently, this model is scaled up for evaluation in an ImageNet context. Building upon this foundation, EfficientNetv2 introduces a further innovation: the incorporation of fused MBConv (MobileNet-like convolution). This mechanism combines the 1x1 expansion convolution with the subsequent 3x3 depthwise convolution into a single 3x3 operation, streamlining both processes and enhancing overall efficiency. Table 5.6 compares the performance of

NAS-based models against a list of comparable and well-known hand-crafted architectures.

5.6 Conclusion

Efficient neural network architectures are a subject of ongoing intensive research within the deep learning community. This research aims to harness the scalability potential of Convolutional Neural Networks (CNNs) for emerging edge computing paradigms. Since the publication of AlexNet, these architectures not only improved accuracy but also advanced the state-of-the-art through optimised proposals. However, developing efficient neural networks for mobile and edge devices highlights the challenge of crafting such models manually.

Throughout this work, we presented a comprehensive array of optimised neural network architectures tailored for edge devices, encompassing more than just microcontrollers. While these architectures may exhibit discrepancies and contradictions, they collectively highlight the deep learning community's commitment to refining model architectures. This emphasises the absence of a one-size-fits-all architecture for edge devices and the necessity for benchmarks when searching for neural network architectures to fit our needs. This chapter provides researchers with a global perspective on significant advancements and their pros and cons, fostering a deeper understanding.

Acknowledgements

This research was conducted as part of the EdgeAI "Edge AI Technologies for Optimised Performance Embedded Processing" project, which has received funding from KDT JU under grant agreement No 101097300. The KDT JU receives support from the European Union's Horizon Europe research and innovation program and Austria, Belgium, France, Greece, Italy, Latvia, Luxembourg, Netherlands, and Norway.

References

[1] Buyya R., Yeo C. S., Venugopal, S., Broberg J., Brandic I. 'Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility', Future Generation Computer Systems, Volume 25, Issue 6, 2009, doi: 10.1016/j.future.2008.12.001.

[2] Steffenel L. A., "Improving the Performance of Fog Computing Through the Use of Data Locality," 2018 30th Int. Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), Lyon, France, 2018, pp. 217-224, doi: 10.1109/CAHPC.2018.8645879.

[3] Mostafa Dehghani et al. The Efficiency Misnomer. The Tenth International Conference on Learning Representations (ICLR), April. 2022. doi: 10.48550/arXiv.2110.12894.

[4] Bichen Wu et al. "Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions". In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, UT: IEEE, June 2018, pp. 9127–9135. doi: 10.1109/CVPR.2018.00951.

[5] Andrew Brown, Pascal Mettes, and Marcel Worring. "4-Connected Shift Residual Networks". In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). Seoul, Korea (South): IEEE, Oct. 2019, pp. 1990–1997. doi: 10.1109/ICCVW.2019.00248.

[6] He Y., Liu X., Zhong H., Ma Y., 'AddressNet: Shift-Based Primitives for Efficient Convolutional Neural Networks,' 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 2019, pp. 1213-1222, doi: 10.1109/WACV.2019.00134.

[7] Ding X., Zhang X., Ma N., Han J., Ding G., Sun J."RepVGG: Making VGG-style ConvNets Great Again". In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021. p. 13733-13742.

[8] Ma N., Zhang X., Zheng HT, Sun J. "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design", Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 116-131.

[9] Krizhevsky A., Sutskever I., Hinton G.E. "ImageNet classification with deep convolutional neural networks". Communications of the ACM 60, 6 (June 2017), 84–90. doi: 10.1145/3065386.

[10] He K., Zhang X., Ren S., Sun J., "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[11] Liu Z., Mao H., Wu C.-Y., Feichtenhofer C., Darrell T., Xie S., "A ConvNet for the 2020s," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 11966-11976, doi: 10.1109/CVPR52688.2022.01167.

[12] Tan, M., Le, Q. "EfficientNET: Rethinking model scaling for convolutional neural networks". In: International conference on machine learning (ICML). 2019. p. 6105-6114. url: http://arxiv.org/abs/1905.11946

[13] Tan, M., Le, Q. "EfficientNETv2: Smaller models and faster training". In: International conference on machine learning (ICML). 2021. p. 10096-10106. doi: 10.48550/arXiv.2104.00298.

[14] Howard A. G., Zhu M., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". arXiv preprint arXiv:1704.04861. 2017.

[15] Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.C. "MobileNnetV2: Inverted residuals and linear bottlenecks". In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018. p. 4510-4520.

[16] Zhang X., Zhou X., Lin M., Sun J., "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 6848-6856, doi: 10.1109/CVPR.2018.00716.

[17] Ma N., Zhang X., Zheng HT., Sun J. "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design". In: Computer Vision – ECCV 2018. LNCS, vol 11218. doi: 10.1007/978-3-030-01264-9_8.

[18] Iandola F.N., Han S., Moskewicz MW., Ashraf K., Dally WJ., Keutzer K. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size". arXiv preprint, Nov. 2016. doi: 10.48550/arXiv.1602.07360.

[19] Mehta S., Rastegari M., Caspi A., Shapiro L., Hajishirzi H. "ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation". In: Computer Vision – ECCV 2018. LNCS vol 11214. doi: 10.1007/978-3-030-01249-6_34.

[20] Mehta S., Rastegari M., Shapiro L., Hajishirzi H., "ESPNetv2: A Light-Weight, Power Efficient, and General Purpose Convolutional Neural Network," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 9182-9192, doi: 10.1109/CVPR.2019.00941.

[21] Simonyan K., Zisserman A. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, May 7-9, 2015. doi: 10.48550/arXiv.1409.1556.

[22] Redmon J., Divvala S., Girshick R., Farhadi A., "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[23] Glenn, J., " YOLOv5 by Ultralytics". doi: org/10.5281/zenodo.3908559.

[24] Chuyi Li et al. "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications". Sept. 2022. doi: 10.48550/arXiv.2209.02976.

[25] Wang CY., Bochkovskiy A., Liao HY. M., "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.. doi: 10.48550/arXiv.2207.02696.

[26] Redmon J., Farhadi A. "YOLOv3: An Incremental Improvement". Apr. 2018. doi: 10.48550/arXiv.1804.02767.

[27] Yan W., Liu T., Fu Y., "YOLO-Tight: An Efficient Dynamic Compression Method for YOLO Object Detection Networks". In: 2021 13th International Conference on Machine Learning and Computing. ICMLC 2021. event-place: Shenzhen, China. New York, NY, USA: ACM, 2021, pp. 378–384. doi: 10.1145/3457682.3457740.

[28] Redmon J., Farhadi A., "YOLO9000: Better, Faster, Stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.

[29] Fang F., Wang L., Ren P. "Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments". In: IEEE Access 8 (2020), pp. 1935–1944. doi: 10.1109/ACCESS.2019.2961959.

[30] Bochkovskiy A., Wang C.Y., Liao H-Y., M. "YOLOv4: Optimal Speed and Accuracy of Object Detection". Apr. 2020. doi: 10.48550/arXiv.2004.10934.

[31] Moosmann J., Giordan M., Vogt, C., Magno, M. "TinyissimoYOLO: A Quantized, Low-Memory Footprint, TinyML Object Detection Network for Low Power Microcontrollers". (2023). arXiv preprint, url:http://arxi v.org/abs/2306.00001

[32] Aharon S., Louis-Dupont, Ofri Masad, Yurkova K., Lotem F., Lkdci, Khvedchenya E., Rubin R., Bagrov N., Tymchenko B., Keren T., Zhilko A., Eran-Deci. "Super-Gradients". 2021. doi: 10.5281/ZENODO.7789328.

[33] Howard A., Sandler M., Chu G., et al. "Searching for MobileNetV3". Proceedings of the IEEE International Conference on Computer

Vision, Seoul, 27 October-2 November 2019, 1314-1324. doi: 10.1109/ICCV.2019.00140

[34] Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C-Y., Berg A.C. "SSD: Single Shot MultiBox Detector". In: Computer Vision – ECCV 2016. LNCS vol 9905. doi: 10.1007/978-3-319-46448-0_2

[35] Lin, TY. et al. "Microsoft COCO: Common Objects in Context". In: ECCV 2014. LNCS vol 8693. doi: 10.1007/978-3-319-10602-1_48

[36] Wang C.-Y., Mark Liao H.-Y., Wu Y.-H., Chen P.-Y., Hsieh J.-W., Yeh I.-H., "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 2020, pp. 1571-1580, doi: 10.1109/CVPRW50498.2020.00203.

[37] Sinong Wang et al. "Linformer: Self-attention with linear complexity". In: arXiv preprint arXiv:2006.04768 (2020).

[38] Mehta S., Rastegari M., "MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer". International Conference in Learning Representation. 2022. Available at:http://arxiv.org/abs/2110.02178

[39] Chollet F. "Xception: Deep Learning with Depthwise Separable Convolutions," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 1800-1807, doi: 10.1109/CVPR.2017.195.

[40] Lin J. et al. "MCUNet: Tiny Deep Learning on IoT Devices". Annual Conference on Neural Information Processing Systems (NeurIPS 2020) Nov. 2020. doi: 10.48550/arXiv.2007.10319.

[41] Lin J. et al., "MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning". Oct. 2021. arXiv preprint. doi: 10.48550/arXiv.2110.15352.

[42] C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.

[43] Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z., "Rethinking the Inception Architecture for Computer Vision," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 2818-2826, doi: 10.1109/CVPR.2016.308.

[44] Ioffe S., Szegedy C. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". Proceedings of the 32nd International Conference on Machine Learning (ICML) 37:448-456, 2015.Mar. 2015. doi: 10.48550/arXiv.1502.03167.

[45] Huang G., Liu Z., Van Der Maaten L. Weinberger K. Q., "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

[46] Xiaohan Ding et al. "Diverse Branch Block: Building a Convolution as an Inception-like Unit". In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Nashville, TN, USA: IEEE, June 2021, pp. 10881–10890. doi: 10.1109/CVPR46437.2021.01074.

[47] Vasu P, Gabriel J., Zhu J., Tuzel O., Ranjan A., "MobileOne: An Improved One millisecond Mobile Backbone". Mar. 2023. arXiv preprint, url:http://arxiv.org/abs/2206.04040

[48] Carion N., Massa F., Synnaeve G., Usunier N., Kirillov A., Zagoruyko S., "End-to-End Object Detection with Transformers". In: ECCV 2020. LNCS vol 12346. doi: 10.1007/978-3-030-58452-8_13.

[49] Adekar S.N., Chaurasia A. "MobileViTv3: Mobile-Friendly Vision Transformer with Simple and Effective Fusion of Local, Global and Input Features". Oct. 2022. arXiv preprint. doi: 10.48550/arXiv.2209.15159.

[50] Mehta S, Rastegari M. "Separable Self-attention for Mobile Vision Transformers". June 202. arXiv preprint. doi: 10.48550/arXiv.2206.02680.

[51] Liu Z. et al., "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 9992-10002, doi: 10.1109/ICCV48922.2021.00986.

[52] Dosovitskiy A. et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", 9th International Conference on Learning Representations (ICLR 2021). May 2021. doi: 10.48550/arXiv.2010.11929.

[53] Xie S., Girshick R., Dollár P., Tu Z., He K., "Aggregated Residual Transformations for Deep Neural Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 5987-5995. Available at:https://doi.org/10.1109/CVPR.2017.634. Deng J., Dong W., Socher R., et al., "ImageNet: A Large-Scale Hierarchical Image Database". 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, 20-25 June 2009, 248-255. doi: 10.1109/CVPR.2009.5206848.

[54] Barret Zoph et al. “Learning Transferable Architectures for Scalable Image Recognition”. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, UT: IEEE, June 2018, pp. 8697–8710. doi: 10.1109/CVPR.2018.00907.

[55] Liu C. et al. “Progressive Neural Architecture Search”. In: ECCV 2018. LNCS vol 11205. doi: 10.1007/978-3-030-01246-5_2

[56] Mingxing T. et al. “MnasNet: Platform-Aware Neural Architecture Search for Mobile”. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, CA, USA: IEEE, June 2019, pp. 2815–2823. doi: 10.1109/CVPR.2019.00293.