Lightweight Deep Learning for Visual Perception: A Survey of Models, Compression Strategies, and Edge Deployment Challenges

Syed Muhammad Raza, [a], Syed Murtaza Hussain Abidi, [a], Md Masuduzzaman, [b] and Soo Young Shin, [a,*]

[a] Department of IT Convergence Engineering, Kumoh National Institute of Technology, 61 Daehak-ro, Geoui-dong, Gumi-si, 39177, Gyeongsangbuk-do, South Korea

[b] Department - Degroote School of Business, McMaster University, 1280 Main St W, Hamilton, ON L8S 4L8, Canada, Hamilton, 1280, Ontario, Canada

ARTICLE INFO

Keywords:
Convolution Neural Networks
Deep Neural Networks
Lightweight Deep Learning
Model Compression
Resource-constrained devices.

ABSTRACT

The increasing demand for the deployment of deep neural networks (DNNs) in edge devices has led to the development of lightweight deep learning (LDL) models designed to operate efficiently under resource constraints. Although DNNs have achieved remarkable success in various applications, their high computational requirements often limit their deployment on devices with restricted memory and processing power. This challenge has motivated researchers to develop optimized LDL models that balance accuracy, speed, and efficiency while maintaining competitive performance. Despite existing surveys covering specific aspects of LDL models, a comprehensive review encompassing image classification, object detection, and segmentation remains limited. This proposed survey systematically explores recent advancements in LDL models, highlighting their architectures, optimization techniques, and real-world applications. This survey conducts an empirical evaluation by testing latest state-of-the-art LDL models on the Jetson Orin edge device using benchmark datasets: ImageNet for classification, VisDrone for object detection, and COCO for segmentation. The experimental analysis focuses on key performance metrics, including inference speed, model accuracy, and computational efficiency, while comparing LDL models with their conventional counterparts. This study provides a holistic understanding of the role of LDL models in edge computing, providing insight into emerging trends, challenges, and future research directions in the field.

1. Introduction

The rapid advancement of deep learning (DL) has significantly accelerated the development of artificial intelligence (AI) applications, facilitating their deployment across a broad spectrum of real-world use cases. Despite their remarkable success, the deployment of DL models, especially CNNs, on embedded systems and resource-constrained edge devices presents substantial challenges. Conventional deep architectures often require significant computational resources, extensive memory capacity, and high power consumption, rendering them impractical for real-time applications on edge devices such as mobile phones, Internet of Things (IoT) nodes, and embedded systems. This challenge has catalyzed the evolution of lightweight deep learning (LDL) models: streamlined CNN-based architectures that emphasize computational efficiency, reduced model size, and improved inference speed without severely compromising accuracy.

While CNN have long dominated deep learning (DL) applications in computer vision, recent advancements have brought transformer-based architectures into prominence. Vision Transformers (ViTs), adapted from natural language processing, have demonstrated competitive performance in vision tasks. However, their reliance on self-attention mechanisms introduces quadratic computational complexity with respect to input size, making them less suitable for edge and embedded environments where resources are constrained. In contrast, CNNs maintain architectural simplicity and computational efficiency, rendering them the preferred backbone for real-time AI on edge devices.

The evolution of LDL models has largely stemmed from a drive to adapt CNN architectures for deployment under stringent hardware constraints. Landmark models such as AlexNet [33], ZFNet [89], VGGNet [4], and Inception-V1 [67] laid the foundation for deep CNN development, while ResNet [63] introduced architectural innovations to mitigate vanishing gradients in deeper networks. However, as CNNs became deeper and more complex, they presented challenges in computational overhead, latency, and memory demand, factors that are incompatible with edge computing requirements.

To address these limitations, LDL techniques have emerged to reduce model size, optimize computation, and enable AI capabilities on resource-constrained platforms.

Lightweight CNN models have been designed to prioritize execution efficiency without severely degrading model performance. Techniques such as pruning (removing redundant weights), quantization (reducing precision), knowledge distillation (training smaller models to mimic larger ones), and Neural Architecture Search (NAS) have been employed to optimize models for edge environments. These techniques have become vital for enabling AI in low-power applications, including autonomous vehicles, intelligent surveillance, and AIoT (Artificial Intelligence of Things) systems. Although

*Corresponding Author: Soo Young Shin (wdragon@kumoh.ac.kr)

iamraza1998@gmail.com (S.M.R.); murtazaabidi@kumoh.ac.kr (S.M.H.A.); masud.prince@mcmaster.ca (M.M.); wdragon@kumoh.ac.kr (S.Y.S.)

ORCID(s):

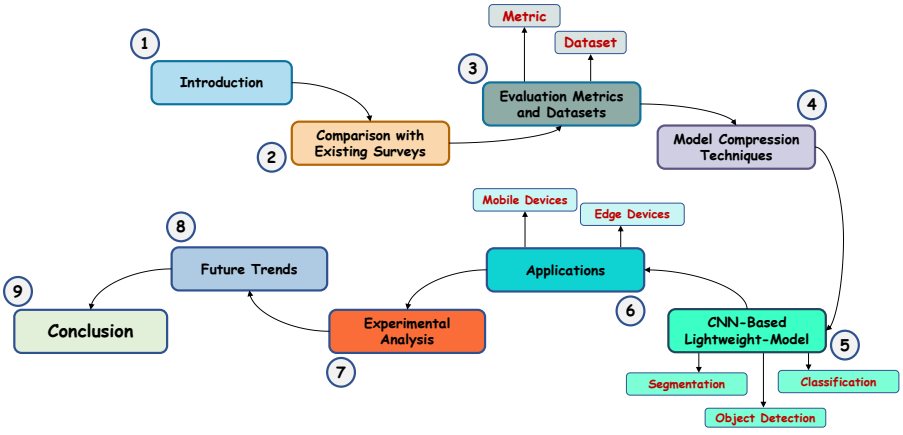


Figure 1: Structure of the paper.

transformer-based vision models have shown impressive results in recent years, their heavy reliance on matrix multiplications and self-attention modules has limited their adoption on edge platforms compared to CNNs. Therefore, CNNs continue to dominate edge AI deployments due to their superior trade-off between performance and efficiency. This survey presents a comprehensive and up-to-date review of LDL models, emphasizing their architectures, design paradigms, and optimized strategies tailored and limited for edge devices and embedded applications fig. 1. Building on previous studies [72, 87, 45, 8], this study integrates recent advances in LDL to provide a holistic understanding of the field. The key contributions of this paper are as follows:

1. This review provides a comprehensive analysis and limitations of the major LDL models, including lightweight image classification, object detection, and segmentation models.
2. This research examines changes in network architectures and their impact on model efficiency.
3. Finally, we performed experimental work to evaluate the latest LDL model performance on the Jetson Orin edge device.

2. Comparison with existing surveys

Previous studies have conducted surveys and reviews on the deployment and usability techniques of LDL models, including model compression. In the early years of LDL models, the research was focused on efficient model compression techniques, hardware accelerators for DNNs models, and custom hardware architectures [72, 38, 15, 88, 78]. Other software-based LDL model techniques were discussed in [16]. Software-based model compression methods encompass a range of algorithm optimization techniques, including quantization, pruning, compression, and approximation. These techniques aim to streamline computational processes while maintaining performance accuracy. However, all such reviews focused on hardware or software-based model compression and acceleration methods for deploying vast DNN models to edge systems, as discussed in [43]. Moreover, deploying DL models on mobile applications, as discussed in [81, 16], includes local optimization on mobile devices and distributed optimization based on the computational position of machine learning tasks. In addition, recent studies were conducted on object detection methods and LDL-based object detection models [87, 97]. These comprehensive reviews were based on the CNN-based object detection backbone architecture using lightweight models and their optimization techniques.

Most existing surveys in the domain of LDL have focused primarily on specific aspects such as model compression techniques, hardware acceleration, or deployment strategies for mobile and edge devices. Although these studies provide valuable information, they often address individual components in isolation, focusing either on image classification, object detection, or segmentation, rarely in an integrated manner. This fragmented perspective creates a challenge for researchers and practitioners who must synthesize knowledge across multiple sources to understand trade-offs between models, compression strategies, and deployment feasibility in different computer vision tasks.

The absence of a unified survey that concurrently examines LDL models across all three core vision tasks image classification, object detection, and segmentation can lead to redundancy in experimentation, suboptimal model selection, and inconsistencies in evaluating deployment performance across application domains. These issues become more pronounced as edge AI applications increasingly require multi-task capabilities on constrained hardware platforms.

This survey addresses this gap by providing a consolidated, task-agnostic overview of LDL models, offering comparative analysis and deployment insights across classification, detection, and segmentation as shown in table 2

Table 1
Detailed comparison of existing survey and review on Lightweight DL models.

Reference	Year	Topic	Details of survey
[73]	2017	Efficient processing of deep neural networks: A tutorial and survey	This paper discusses various hardware platforms and architectures that support DNNs, and highlights key trends in reducing the computation cost of DNNs.
[12]	2018	Model compression and acceleration for DNN	This paper reviews works on compressing and accelerating DNNs, which attracted considerable attention from the DL community.
[16]	2019	DL on mobile devices: A review	This paper covers algorithm optimization techniques, such as quantization, pruning, compression, and approximations that simplify computation while retaining performance accuracy.
[15]	2020	Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey	This paper discusses and reviews the mainstream compression approaches such as compact models, tensor decomposition, data quantization, and network sparsification.
[88]	2021	Custom Hardware Architectures for Deep-Learning on Portable Devices: A Review	This paper discusses several DL accelerators, as well as technologies with emerging devices, to highlight their architectural features in application-specific integrated circuit (IC) and field-programmable gate array (FPGA) platforms.
[43]	2021	Bringing AI to edge: From DL perspective	This paper discusses DL techniques for edge intelligence systems.
[81]	2022	A survey on deploying mobile DL applications: A systemic and technical perspective	This paper discusses local optimization on mobile devices and distributed optimization based on the computational position of machine learning tasks.
[87]	2022	A survey of modern DL-based object detection models	This paper discusses object detection and lightweight object detection models.
[97]	2023	Object detection in 20 years: A survey	This paper discusses object detection techniques of the last 20 years including LDL models.
[8]	2024	Review of lightweight deep CNNs	This paper discusses LDL CNN-based models.
[47]	2024	Image recognition based on lightweight convolutional neural network	This paper discusses model compression techniques and LDL image classification models.
[45]	2024	Lightweight deep learning for resource-constrained environments	This paper discusses traditional LDL CNN models with model compression techniques.
The proposed work	**2025**	**A Comprehensive Survey on Lightweight DL Models and their Applications**	**This paper presents a comprehensive survey on LDL models and their applications, including model compression techniques, lightweight image classification, object detection, and segmentation models.**

summarizes the distinctions between this work and prior surveys. By presenting a comprehensive synthesis of the field, this paper aims to serve as a foundational resource to guide model development, selection of compression strategies, and optimization at the system level for edge-based AI systems.

3. Metrics and Datasets for LDL Models

This section provides an overview of the commonly used evaluation metrics and benchmark datasets that have been widely adopted in LDL models. These metrics and datasets reflect the broader evaluation ecosystem used in the literature and are not limited to the specific empirical benchmarks presented later in this paper.

3.1. Evaluation Metrics

LDL models are typically assessed using both task-specific accuracy measures and resource-oriented efficiency metrics. For image classification, accuracy, top-1 accuracy, and top-5 accuracy are the most commonly reported performance metrics. These indicate the proportion of correctly predicted class labels from all predictions, with top-k variants considering the top-k model outputs.

In object detection, the primary metric is the mean Average Precision (mAP), calculated as the mean of average precision (AP) across classes. AP itself is derived from precision recall curves, typically at different intersection over union (IoU) thresholds. The Pascal VOC protocol uses a fixed IoU threshold of 0.5, whereas MS COCO employs mAP across multiple IoU thresholds (from 0.5 to 0.95), making it more rigorous. In segmentation tasks, metrics like IoU (Jaccard Index), pixel accuracy, and mean pixel accuracy are commonly used.

In addition to task-specific metrics, **efficiency-oriented measures** are critical in evaluating LDL models. These include:

- **Inference time or frames per second (FPS):** Quantifies how fast a model performs predictions.
- **Model size (in MB) and number of parameters:** Reflect the storage and memory footprint.
- **Floating point operations per second (FLOPs):** Measures computational complexity.

These metrics together offer a multi-dimensional understanding of an LDL models suitability for deployment on constrained hardware.

Table 2
Dataset Details

Dataset Details					
Dataset Name	**No of Images**	**No of Classes**	**Category**		
			Object Detection	**Segmentation**	**Image Classification**
ImageNet	14M+	1000	✓	✗	✓
CIFAR-10	60K	10	✗	✗	✓
CIFAR-100	60K	100	✗	✗	✓
MNIST	70K	10	✗	✗	✓
Fashion-MNIST	70K	10	✗	✗	✓
SVHN	600K+	10	✗	✗	✓
Caltech-101	9K+	101	✗	✗	✓
Caltech-256	30K+	256	✗	✗	✓
Pascal VOC 07/12	11K	20	✓	✓	✗
ILSVRC Detection	456K	200	✓	✗	✗
MS-COCO	2.5M	91	✓	✓	✗
Open Images	9.2M+	600	✓	✓	✗
Cityscapes	25K	30	✗	✓	✗
CamVid	701	32	✗	✓	✗

3.2. Benchmark Datasets

This section presents an overview of the most popular datasets used in LDL models, including image classifications, object detection, and segmentation models. This study categorizes the datasets with each specific category and presents a dataset table to better understand the readers as shown in table 2.

4. Model Compression Techniques

The increasing deployment of DNNs on edge and embedded systems necessitates reducing their computational and memory demands without significantly compromising performance. To address this, a broad spectrum of model compression techniques has been proposed, with the most prominent being pruning, quantization, knowledge distillation, and neural architecture search (NAS) [78]. While these methods have shown promising results individually, each has unique design principles, operational trade-offs, and practical limitations. This section analyzes these compression paradigms in terms of their structural goals, commonalities, distinctions, and suitability across application scenarios. Furthermore, we highlight open research questions and future directions that can guide the development of more effective and deployable LDL models.

4.1. Pruning

Pruning is one of the most established and widely adopted techniques for reducing the computational complexity of DNNs, particularly in resource-constrained environments. The central premise of pruning is that not all parameters or neurons contribute equally to a models predictive capacity. Many weights, filters, or channels are redundant or minimally influential and can be removed with negligible impact on accuracy. By eliminating these components, pruning yields thinner, more efficient networks that are better suited for deployment on edge devices and embedded systems [26].

In CNNs, pruning can be applied at various granularities, including individual weights, neurons, filters, or entire layers. For instance, filter pruning reduces the number of convolutional filters, thereby decreasing the number of multiply-and-accumulate (MAC) operations required during inference [11]. This not only lowers latency and memory usage but also facilitates deployment on hardware accelerators with limited computational throughput. Fig. 2 illustrates a schematic representation of how pruning removes non-essential weights and neurons from a network.

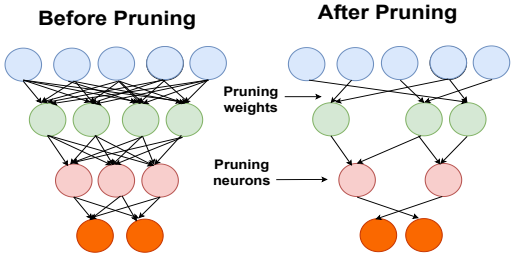


Figure 2: Schematic illustrating neuron and weight pruning. Circles represent neurons; arrows represent weights. Redundant elements are removed to reduce complexity. Courtesy of S. Hen et al. [15].

Two primary categories of pruning techniques are widely used in the literature:

Weight pruning focuses on eliminating individual connections within the network by identifying low-magnitude weights. This technique was pioneered by LeCun et al. and later advanced by Han et al., who demonstrated that sparsity can be effectively induced via L1 or L2 regularization [1, 29]. L1 regularization tends to zero out weights below a certain threshold, while L2 regularization offers smoother penalization, often yielding better generalization. By exploiting weight-level redundancy, high compression

ratios can be achieved with relatively minor performance degradation.

Filter pruning, in contrast, operates at a higher structural level by evaluating the importance of entire convolutional filters. Common approaches involve computing the L1- or L2-norm of filters or assessing their influence on the model loss function [13, 41]. Advanced techniques further introduce ranking criteria based on the activation feature maps or low-rank approximations. For example, HRank [41] proposes pruning based on the rank of feature maps, thereby ensuring the removal of least informative filters. This form of structured pruning is more hardware-friendly than unstructured pruning and enables efficient acceleration on parallel processors and edge inference engines.

4.2. Quantization

Quantization is a widely adopted model compression technique that aims to reduce the numerical precision of parameters and activations in DL models, thereby lowering computational complexity, memory usage, and energy consumption. To facilitate efficient neural network implementation on hardware [23]. The standard 32-bit floating-point format used for storing weights and activations imposes significant hardware and power demands, motivating the use of lower bit-width formats such as 16-bit, 8-bit, or even binary representations.

Recent advancements [48, 94] in quantization research have focused on balancing the trade-off between computational efficiency and model accuracy. These studies highlights methods to minimize quantization error by preserving the statistical distribution of full-precision representations during training and inference. One commonly used approach is low-bit precision quantization, which compresses model weights and activations into reduced numeric formats while maintaining acceptable accuracy levels. For example, 16-bit floating-point (FP16) quantization has become popular due to its compatibility with modern GPUs and specialized AI accelerators, typically yielding negligible performance degradation [32]. Even more aggressive, 8-bit integer quantization has demonstrated strong performance across vision tasks with minimal accuracy loss, as shown in the work of He et al. [68]. More advanced strategies such as ternary quantization use discrete levels (e.g., -1, 0, +1) to further reduce computation, with the advantage of eliminating multiplications involving zero-valued weights [35].

Binarization represents an extreme form of quantization, reducing all weights and activations to binary values (0 or 1). Several studies have explored this technique for edge deployment, demonstrating its effectiveness in extremely low-power environments [9, 40]. However, the aggressive reduction in precision often results in significant accuracy degradation, especially on complex datasets. Fig. 3 illustrates a schematic of the binarization process, where both neurons and weights operate in binary form.

4.3. Quantization Sensitivity

Quantization sensitivity extents to which a model performance degrades when reducing numerical precision from full-precision floating point (e.g., FP32) to low-bit formats such as INT8 or lower.

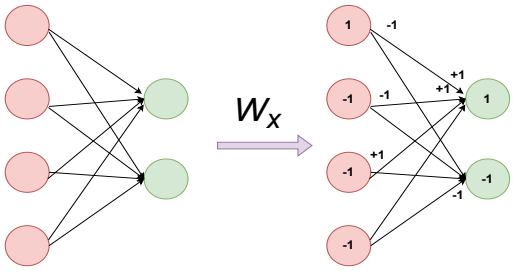


Figure 3: Visual representation of binary neural network where neurons and weights are encoded in binary format, enabling efficient computation. Adapted from Deng et al. [15].

Table 3
Quantization sensitivity of CNN vs. Transformer-based lightweight models on ImageNet.

Model	FP32 Acc. (%)	INT8 Acc. (%)	Drop (%)
MobileNetV3-Large [28]	75.2	74.9	0.3
EfficientNet-B0 [75]	77.1	76.6	0.5
MobileViT-S [51]	78.4	76.0	2.4
LeViT-128S [22]	76.5	74.0	2.5

CNN-based lightweight architectures, such as MobileNetV3 and EfficientNet-B0, typically exhibit strong resilience to post-training quantization (PTQ), with accuracy drops often remaining below 0.5% on benchmark datasets [28, 75]. This robustness is largely attributed to the localized nature of convolutional operations and the relatively smaller dynamic range of feature activations, which enables more uniform quantization without significant information loss.

In contrast, transformer-based lightweight models, including MobileViT and LeViT, demonstrate markedly higher quantization sensitivity. The self-attention mechanism relies on high-precision dot products and normalization operations (e.g., LayerNorm), both of which are prone to amplified quantization noise. The variance across token embeddings and the wide dynamic range of attention scores exacerbate numerical instability, resulting in disproportionately large performance drops. Such sensitivity is especially problematic for edge deployments, where INT8 inference is often mandatory to meet memory and latency budgets. Furthermore, the lack of standardized low-bit optimization kernels for transformer operations on many embedded platforms limits their practical adoption despite strong FP32 performance.

Recent studies [51, 75] have reported that INT8 PTQ can cause performance degradations of 2–3% for MobileViT on ImageNet, compared to less than 0.5% for MobileNetV3, underscoring an architectural gap in robustness to aggressive quantization. Table 3 illustrates the comparative quantization sensitivity of selected CNN and transformer-based lightweight models under INT8 PTQ.

To mitigate quantization-induced performance loss in transformer-based LDL models, several strategies have been explored. **Quantization-aware training** (QAT) simulates

quantization effects during training, enabling models to adapt weights and activations to lower precision. **Mixed-precision quantization** selectively maintains higher precision (e.g., FP16) for sensitive layers such as attention mechanisms, while quantizing other components (e.g., MLP layers) to INT8. Furthermore, **per-channel scaling** improves representational fidelity by tailoring quantization parameters to individual channels, reducing dynamic range compression. Despite these advances, the additional training cost, hardware compatibility issues, and complexity of integrating mixed-precision pipelines remain open challenges. As a result, improving transformer robustness to low-bit quantization remains a critical direction for enabling their broader use in constrained edge environments.

4.4. Knowledge Distillation

Knowledge distillation has become a widely used strategy for compressing deep learning models, particularly when high accuracy must be preserved while reducing model size and computational cost. Unlike direct parameter reduction techniques, knowledge distillation employs a dual-model framework: a large, well performing teacher model and a smaller, more efficient student model. The student is trained not only on hard labels from the dataset but also on the richer output distributions or internal representations learned by the teacher. This method was originally popularized by Hinton et al. [27], and has since evolved into a diverse set of techniques tailored to different tasks and deployment constraints.

The standard distillation process typically transfers soft logits probability distributions generated by the teacher into the student model under a softened temperature scale, enabling the student to capture inter class relationships that are often absent in one hot labels. Romero et al. [21] demonstrated that aligning internal layer outputs between teacher and student networks can significantly enhance learning in compact architectures. This concept is particularly useful when compressing deep networks for image classification, object detection, or segmentation tasks.

Fig. 4 provides a visual overview of the knowledge distillation framework, highlighting the dual supervision pathway from both logits and intermediate layers.

Several variants of knowledge distillation have been explored:

Logit-based distillation focuses on transferring output distributions from the teacher model. Using softened logits, the student learns to approximate the teacher's prediction behavior rather than relying solely on ground-truth labels. Malinin et al. [50] extended this approach by distilling information from ensemble models to capture uncertainty and diversity in predictions. Moreover, Ku et al. [34] proposed a hierarchical scheme involving teacher assistants of intermediate capacity to bridge the gap between large teachers and small students, thereby improving knowledge transfer efficiency.

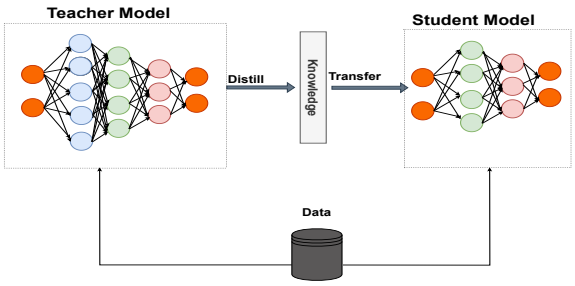


Figure 4: Illustration of knowledge distillation framework, where the student model learns from both soft logits and internal feature representations of the teacher. Adapted from Gou et al. [21].

Feature-based distillation transfers knowledge from intermediate layers or attention maps. This method is especially effective when the goal is to preserve spatial features or semantic structures in vision tasks. Aguilar et al. [2] explored distillation for object detection by aligning high-level response maps between teacher and student, bypassing soft output logits. Aligning internal features allows student models to mimic the hierarchical representation learning of deeper architectures, leading to enhanced generalization despite fewer parameters.

Distillation schemes are typically categorized as offline or online. In *offline distillation*, the teacher is pre-trained and fixed; the student then learns from stored or generated outputs. This setup is straightforward to implement but computationally demanding due to the need to first train a high-capacity teacher. In contrast, *online distillation* trains both teacher and student simultaneously, often using mutual learning strategies among multiple peer networks. Zhang et al. [91] and Huang et al. [30] demonstrated that such collaborative learning frameworks can yield student models that outperform the original teacher. Online approaches offer significant training efficiency and are especially attractive for distributed and privacy-sensitive training pipelines.

4.5. Neural Architecture Search

Neural Architecture Search (NAS) is a model compression and automation technique that aims to design neural network architectures optimally suited for a given task or hardware constraint. Unlike manual architecture design, NAS explores a predefined search space to automatically discover efficient network configurations that balance accuracy, speed, and resource usage. This is particularly advantageous for deploying deep learning models in low-resource environments where handcrafted architectures may be suboptimal.

One of the core challenges in NAS lies in the intractable size of the search space. For a network with N layers and kernel size K, there can be up to N^K possible architectural combinations, making exhaustive search computationally infeasible. To address this, a wide range of strategies based

on reinforcement learning (RL), evolutionary algorithms, and gradient based optimization have been proposed to traverse this space effectively.

Early contributions by Baker et al. and Zoph et al. [5, 95] leveraged RL to generate network architectures for image classification. Later, Real et al. [59] demonstrated that evolution-based approaches not only achieve competitive accuracy but also tend to produce more compact models. These methods have since been extended to tasks beyond classification, including semantic segmentation [44, 53]. As summarized by Elsken et al. [17], NAS methods can be systematically categorized into three components: search space, search strategy, and evaluation strategy as shown in fig. 5.

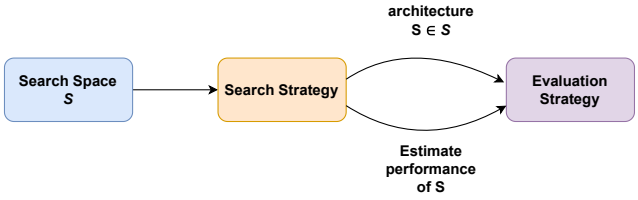


Figure 5: Core components of a NAS pipeline. A model is selected from a defined search space, guided by a search strategy, and validated through an evaluation mechanism. Adapted from Liang et al. [40].

Search Space: The search space defines the possible building blocks and configurations from which architectures are assembled. It may include parameters such as the number and type of layers, kernel sizes, skip connections, and activation functions. Most NAS frameworks adopt a cell-based design, where modular units are repeatedly stacked to form deep networks. Zoph et al. [96], Real et al. [59], and Chen et al. [10] explored fixed cell structures, while later works such as Saikia et al. [64] and Cui et al. [14] proposed dynamic and heterogeneous cell formations. Fang et al. [19] emphasized exploration of the global search space to relax structural constraints and enhance design diversity.

Search Strategy: This component determines how the search space is explored to identify high-performing architectures. RL-based approaches [95], while effective, are computationally expensive requiring thousands of GPU days in early implementations. To mitigate this, differentiable methods like DARTS [44] reformulated architecture search as a continuous optimization problem solvable via gradient descent. Other strategies include single-path sampling [7], meta-heuristics [10, 31], and surrogate models that estimate performance to accelerate search.

Evaluation Strategy: The evaluation strategy governs how candidate architectures are assessed during the search. Full training of each candidate is computationally prohibitive, prompting the development of proxy metrics and weight-sharing methods. Pham et al. [56] introduced one-shot NAS with shared parameters to accelerate evaluation, while Baker et al. [5] and Zheng et al. [93] proposed learning curve and performance prediction techniques.

5. Lightweight Classification Models

This section provides a comprehensive overview of state-of-the-art models for LDL classification. Most LDL approaches leverage CNNs as backbone architectures, as summarized in table 4.

5.1. SqueezeNet

SqueezeNet [83] is a DL framework specifically developed to enable efficient neural network inference on devices with limited resources, such as mobile phones or embedded systems. The architecture includes a combination of 1×1 and 3×3 convolutions. The primary use of 1×1 convolutions is to increase the number of channels, enhancing the potential to capture intricate patterns through learning. The use of 3×3 convolutions facilitates the acquisition of spatial information and contributes to the acquisition of hierarchical characteristics. Some 3×3 convolutional operations are replaced with low-complexity 1×1 convolution operations. However, there are limitations in SqueezeNet which overcome by ResNet [77], introduced a modified version of SequeezeNet incorporated residual connections. This adaptation resulted in improved accuracy compared with the original vanilla model. Gschwend et al. [3] presented a modified iteration of SqueezeNet and deployed it on an FPGA in their research. The SqueezeNet model exhibits considerable promise for enhancing the hardware efficiency of neural network architectures , as shown in fig.6

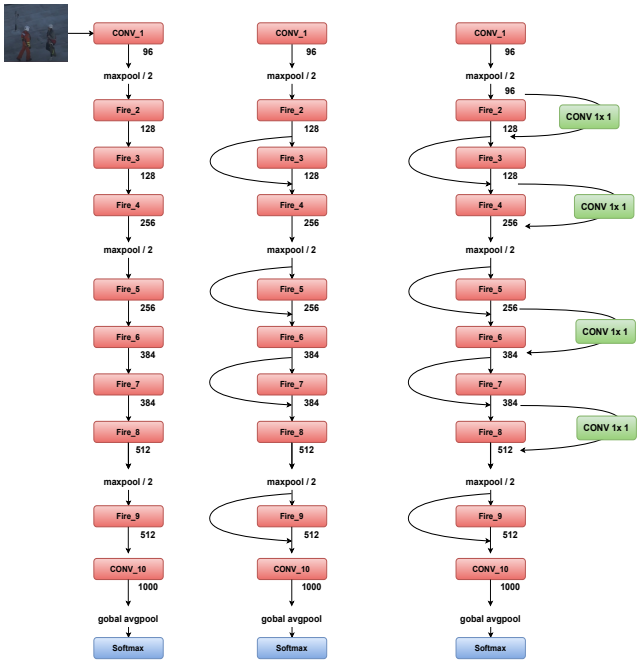


Figure 6: Structure diagram of SqueezeNet. Left: SqueezeNet; Middle: SqueezeNet with simple bypass; Right: SqueezeNet with complex bypass [57].

Table 4
Details of LDL image classification models and their datasets.

Model	Parameters (M)	Accuracy (%)	Year	Dataset
SqueezeNet [57]	1.24	60.4	2016	ImageNet
MobileNet [24]	4.2	70.6	2017	
MobileNetV2 [65]	3.4	72.0	2018	
MobileNetV2-Large [65]	6.9	74.7	2018	
ShuffleNet [86]	3.4	71.5	2018	
ShuffleNetV2 [49]	2.3	69.4	2018	
MnasNet [74]	3.9	75.2	2018	
MobileNetV3-Small [28]	2.5	67.4	2019	
MobileNetV3-Large [28]	5.4	75.2	2019	
EfficientNet-B0 [74]	5.3	77.1	2019	
GhostNet [25]	5.2	73.9	2020	
MobileViT [51]	5.6	78.4	2021	
EfficientFormer [37]	12.3	79.2	2022	
EdgeFormer [20]	4.6	77.5	2022	
TinyFormer [84]	6.1	80.1	2023	
LT-IMFEN [92]	3.8	76.7	2024	
LYT-NET [6]	2.9	–	2025	LOL, SID, ExDark

5.2. MobileNets

In 2017, mobilenet [24] was explicitly designed to deploy mobile and embedded devices with limited computational resources efficiently. It moved away from an approach that diverged from traditional techniques such as model reduction. The network utilizes a depthwise separable convolution, which involves the decomposition of a traditional convolution into two distinct operations: depthwise and pointwise convolution using a 1×1 kernel. In conventional convolution, kernels are collectively applied to all the input channels, resulting in a single combined output. The MobileNet architecture comprises 28 convolutional layers, each sequentially accompanied by batch normalization and the ReLU activation function. Furthermore, the authors suggested the integration of two hyperparameters, precisely the width and resolution multipliers, to optimize the computational efficiency and reduce the model dimensions. A width multiplier is employed to uniformly modify the width of the network by decreasing the number of input and output channels.

5.3. ShuffleNet

In the following year, the ShuffleNet algorithm [86] was developed to reduce computational complexity and memory usage through the implementation of channel shuffling and group convolutions. Channel shuffling facilitates information exchange between different groups of channels, enabling efficient communication and reducing the number of parameters specifically designed for mobile devices. The primary components of the ShuffleNet architecture include a conventional convolutional layer, followed by a sequence of ShuffleNet units arranged into three separate stages. The ShuffleNet unit resembles the ResNet block owing to its incorporation of a depthwise convolution in the 3×3 layer while replacing the 1×1 layer with a pointwise group convolution. ShufflNnet refers to channel shuffling without any corresponding enhancement in the inference speed of the model as shown in fig. 7.

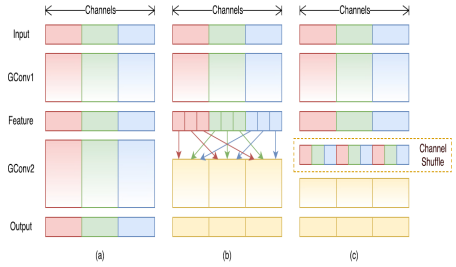


Figure 7: Two stacked group convolutions for channel shuffling. a)Two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when GConv2 (group convolution) takes data from different groups after GConv1; c) an equivalent implementation to b) using channel shuffle [39].

5.4. MobileNetv2

Sandler et al. introduced an improved version of Mobilnetv1 [24] aimed to optimize accuracy while ensuring efficiency for devices with limited resources such as mobile devices.. MobileNetv2 [65] using the main concepts of MobilNetv1 such as depth-wise separable convolutions and model parameter reduction. The MobileNetv2 architecture introduces a novel module called the **inverted residual with a linear bottleneck**, as shown in fig. 8. which aims to decrease the computational requirements and enhance accuracy. This module uses a technique to enhance low-dimensional representations of the input by transforming

them into higher dimensions. This was achieved by applying a depthwise convolution filter and projecting the results on the original lower dimensions. This methodology deviated from a traditional residual block, typically involving compression, convolution, and expansion operations. The utilized MobileNetv2 as a feature extractor for object detection, employing a computationally efficient variant of SSD [46].

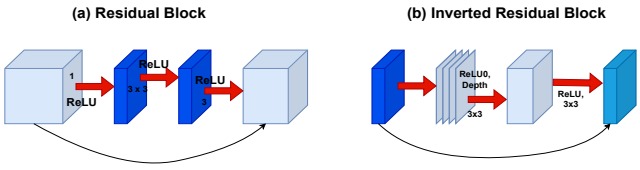


Figure 8: Difference between residual and inverted residual block. The main difference between them is residual blocks are connects with layers with higher no. of channels, whereas the inverted residual blocks connects the bottlenecks [65].

5.5. ShuffleNetv2

ShuffleNetv2 [49] has been recognized for its exceptional performance in terms of speed, accuracy, and the development of efficient network architectures.

The input was divided into two segments using a channel split layer. These segments were processed using three convolutional layers. The outputs generated by these layers were merged with the residual connection and subsequently processed through a channel shuffle layer. The downsampling model demonstrated the absence of a channel split. Simultaneously, incorporating depthwise separable convolution layers was evident through utilizing the residual connection, as shown in fig. 9.

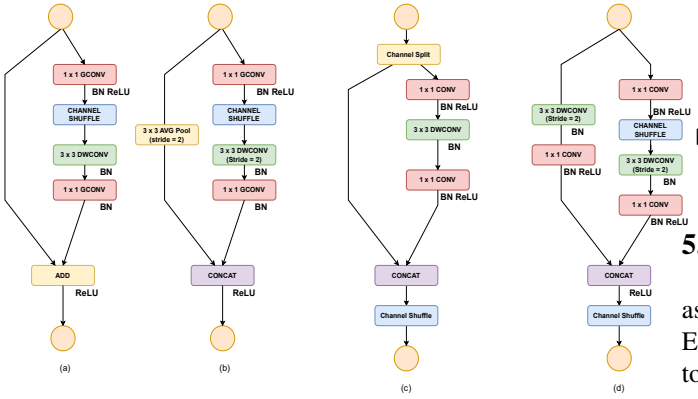


Figure 9: Difference between ShuffleNetv1 and Shufflev2.(a): the basic ShuffleNet unit; (b) the ShuffleNet unit for spatial down sampling (2×); (c) Basic unit; (d) our unit for spatial down sampling (2×). DWConv: depthwise convolution. GConv: group convolution [49].

5.6. MobileNetv3

In 2019 MobileNetv3 [28] was introduced with its two versions MobileNetv3-Large and MobileNetv3-Small. The essential contribution of Howard et al. to MobileNetv3 was the use of NAS. NAS is popular for automatically discovering optimal network architectures. This approach includes training and evaluating many candidate architectures to identify the best trade-off between accuracy and efficiency.

MobileNetv3 has undergone further modifications to eliminate specific expensive layers within its architecture, resulting in additional improvements in latency. The utilization of this methodology reduced the computational processes. MobileNetv3 combines ReLU and hard-swish activation functions as filters, with the latter predominantly utilized in the later stages of the model.

5.7. MnasNet

MnasNet is a neural network architecture inspired by an automated NAS approach [74]. The main objective of MnasNet is to design an efficient and high-performance model that can rapidly deploy resource-constrained devices while maintaining high accuracy. The search space is decomposed by dividing the CNN into separate blocks. This enables the independent exploration of operations and connections within each block, thus reducing the overall search space. Furthermore, this feature facilitated the creation of unique designs for each block. This contrasts with previous models [95, 42, 59] that utilized identical stacked blocks. However, similar to other NAS models that rely on reinforcement learning, the search for architectures in MnasNet requires extensive computational resources, as shown in fig. 10.

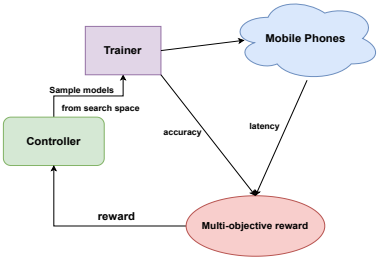


Figure 10: Overivew of platfor aware NAS for Mobile [74].

5.8. EfficientNet-BO

In 2019, EfficientNet-B0 [75] (baseline) was introduced as the smallest and lightest version among other versions of EfficientNet. The EfficientNet-B0 architecture was designed to balance high efficiency and accuracy. This renders it suitable for devices with limited resources. A combination of depth, width, and resolution scaling techniques were used, as outlined in the EfficientNet research paper. Moreover, the EfficientNet-B0 model exhibits a slightly larger size, primarily attributed to its larger target of FLOPS, which was set at 400 million. This distinguishes it from the other versions of the EfficientNet model.

5.9. GhostNet

K. Han et al. [25] introduced a new model, GhostNet. The architecture aimed to optimize computational efficiency

while increasing model accuracy by incorporating a novel component known as the Ghost Module. This module employed a novel technique known as "ghost convolution," which is a cost-effective operation that requires fewer computational resources than conventional convolutions. The Ghost module aims to enhance the generation of feature maps by utilizing cost-effective operations. The purpose of this module is to provide a flexible and customize component that can be used to improve existing CNN. Ghost bottlenecks are specifically engineered to facilitate the stacking of Ghost modules, enabling the seamless establishment of a lightweight GhostNet.

5.10. MobileViT

MobileViT [51] is a lightweight vision transformer architecture designed for mobile and embedded applications. It combines the strengths of CNNs in modeling local spatial dependencies with the capability of ViTs to capture long-range global relationships. The architecture employs an inverted residual block for initial feature extraction, followed by MobileViT blocks that replace certain convolutional layers. Each MobileViT block flattens local feature maps into sequences, applies a transformer encoder for global context modeling, and reshapes the output back to the spatial domain. This hybrid design significantly reduces computational cost while retaining the representational power of larger ViTs. While efficient on mobile hardware, MobileViT performance drops compared to heavier transformer models on complex datasets.

5.11. EfficientFormer

EfficientFormer [37] is a hardware-friendly vision transformer designed to achieve MobileNet-level latency while retaining the accuracy advantages of transformer-based architectures. The design adopts a stage-wise hybrid architecture, beginning with convolutional layers for local feature extraction, followed by multi-head self-attention layers for capturing global dependencies. It leverages dynamic token reduction, merging less informative tokens as depth increases, thereby reducing quadratic complexity. The architecture is optimized using latency-aware NAS to ensure deployment efficiency. Despite its speed advantages, EfficientFormer sacrifices some accuracy compared to full-scale ViTs on fine-grained classification tasks. Moreover, the token merging mechanism may inadvertently discard subtle but important features in dense object scenes.

5.12. EdgeFormer

EdgeFormer [20] is a lightweight vision transformer tailored for edge computing environments. It aims to balance accuracy and computational efficiency for on-device vision tasks. The architecture incorporates efficient attention mechanisms such as linear attention and low-rank factorization to reduce complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. A pyramid feature hierarchy is employed, enabling multi-scale context aggregation, which is crucial for tasks like object detection and segmentation on edge devices. EdgeFormer lightweight design can result in reduced representation capacity for very large-scale datasets. The linearized attention mechanism may also lose fidelity when modeling complex spatial relationships compared to full self-attention.

5.13. TinyFormer

TinyFormer [84] is an efficient transformer architecture developed for deployment on highly resource-constrained devices. It adopts a compact transformer backbone with aggressive parameter reduction through depthwise separable convolutions and shared projection layers. A reduced token size and simplified multi-head attention enable real-time inference while maintaining competitive accuracy on small- to medium-scale datasets like CIFAR-100 and ImageNet-1K. Although optimized for size and speed, TinyFormer underperforms on high-resolution datasets and complex tasks requiring large receptive fields. The aggressive compression may lead to loss of fine-grained feature details.

5.14. LT-IFEN

Lightweight Transformer Image Feature Extraction Network (LT-IFEN) [92] is a lightweight transformer-based model designed for efficient image feature extraction in edge computing and low-power applications. The model integrates convolutional layers for early-stage feature encoding with a transformer module for capturing non-local dependencies. Its streamlined design emphasizes reduced parameter count and memory footprint, making it suitable for embedded systems. The use of lightweight attention blocks ensures low-latency inference without compromising essential feature richness. LT-IFEN may face difficulties in learning highly abstract features when applied to large-scale, diverse datasets. Furthermore, its reduced depth limits its capacity for multi-level semantic reasoning.

5.15. LYT-NET

LYT-NET [6] is a transformer-based architecture specialized for low-light image enhancement, particularly in the YUV color space. It is tailored to address the challenges posed by noise, color distortion, and reduced contrast in poorly illuminated environments. The network processes the luminance and chrominance components separately, enabling targeted enhancement of brightness and color fidelity. A transformer backbone models long-range dependencies, while lightweight convolutional layers refine local details. The design facilitates effective enhancement with minimal computational cost, making it suitable for real-time mobile deployment. While highly effective in low-light scenarios, LYT-NET is less suitable for tasks requiring robust performance in varied lighting conditions. It also relies on paired low-light and normal-light datasets for supervised training, which may limit its applicability in certain domains.

6. Lightweight Object-Detection Models

Lightweight object detection models are designed to enhance efficiency by reducing model size, computational complexity, and memory consumption, as detailed in table

Table 5
Details of LDL object detection models used dataset.

Models	Year	Parameters	Accuracy	Dataset
Light-Head RCNN [70]	2017	–	75.1%	PASCAL VOC
Tiny-YOLO [60]	2018	15.12 M	57.1%	PASCAL VOC
Tiny-SSD [82]	2018	1.13 M	61.3%	PASCAL VOC
Pelee [79]	2018	5.98 M	70.9%	PASCAL VOC COCO
ThunderNet [58]	2019	–	75.1%	PASCAL VOC
EfficientDet [76]	2020	8.1 M	43.0%	COCO
LWUAVDet-S [52]	2024	5.2 M	46.7%	VisDrone2021

5. These models are particularly well-suited for real-time applications where resource constraints are critical. Common use cases include mobile applications, robotics, surveillance systems, and Internet of Things (IoT) devices, where fast and efficient object detection is essential.

6.1. Light-Head RCNN

The Light-Head RCNN [70] is a two-stage detection model developed with efficient design while maintaining accuracy. The primary goal is to reduce the overall weight of the network head by using a thin feature map and a cost-efficient R-CNN subnet, which comprises pooling and a solitary, fully connected layer. The object detector based on ResNet-101 and a light-head R-CNN has demonstrated superior performance compared to existing state-of-the-art object detectors on the COCO dataset, maintaining an efficient processing time. Furthermore, a significant aspect to consider is the substitution of the backbone with a smaller network, such as Xception. The Light-Head R-CNN model performs better than fast single-stage detectors such as SSD [46] and YOLOv2 [61] while exhibiting faster computational speed. The architecture of Light-Head RCNN is shown in fig. 11.

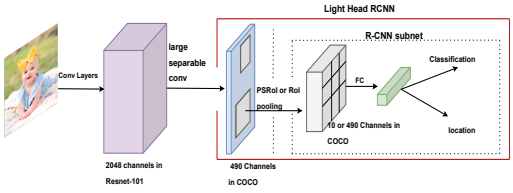


Figure 11: Model constructs feature maps with reduced dimensions before region of interest (RoI) warping by utilizing large separable convolutions. In our R-CNN subnet, we incorporate a solitary fully-connected layer that comprises 2048 channels [70].

6.2. Tiny-YOLO

Tiny-YOLO was first introduced in 2017 by Redmon et al. [60]. This is an object detection CNN-based architecture. Tiny-YOLO is approximately 422% faster than YOLO and achieves upwards of 244 FPS on a single GPU. The Fast YOLO, a light variant of the YOLO algorithm, achieves a processing rate of 155 FPS. Remarkably, it surpassed other real-time detectors in terms of mAP by a significant margin of two. Finally, the YOLO algorithm incorporates comprehensive representations of various entities.

6.3. Tiny-SSD

Tiny-SSD, described in [82], is a deep CNN explicitly designed for real-time embedded object detection. The system comprises a heterogeneous fire subnetwork stack and a heterogeneous sub-network stack comprising auxiliary convolutional feature layers based on SSD. The components mentioned above have been extensively optimized to effectively minimize the overall size of the model while ensuring that the object detection performance remains uncompromised. The developed Tiny SSD model exhibited a model size of 2.3MB, representing a reduction of approximately 26 times compared with the Tiny YOLO model. Moreover, despite its smaller dimensions, the Tiny SSD model demonstrates a mAP of 61.3% on the VOC 2007 dataset, surpassing the performance of the Tiny YOLO model by approximately 4.2%.

6.4. Pelee

Several efficient architectures, such as MobileNet, ShuffleNet, and MobileNetV2, have recently been proposed. Nevertheless, the efficacy of these models is mainly contingent on the utilization of depthwise separable convolution, a technique that is not optimally implemented in most DL frameworks. PeleeNet [79] was specifically developed to meet the requirements of real-time object detection tasks, focusing on achieving high accuracy while maintaining computational efficiency. It demonstrates a model size that is only 66% of the magnitude observed in MobileNet. The proposed method incorporates PeleeNet into a single-shot MultiBox Detector (SSD). This approach, focusing on optimizing the system architecture to improve its speed capabilities. The detection system developed by Pelee exhibited an mAP of 76.4% on the PASCAL VOC2007 dataset and 22.4 mAP on the MS COCO dataset.

6.5. ThunderNet

The task of real-time generic object detection on mobile platforms is a crucial and intricate computer vision task. It was previously observed that lightweight CNN-based detectors employed a one-stage pipeline. ThunderNet [58] utilizes the MobileNetV3 backbone, a feature extraction network known for its lightweight and efficient characteristics. It utilizes a multi-path architecture to integrate features obtained at various scales and resolutions. This facilitates the acquisition of objects with diverse dimensions and preserves the spatial data. ThunderNet performs better than previous lightweight one-stage detectors on widely recognized benchmarks such as PASCAL VOC and COCO. This achievement was attributed to utilizing a highly efficient backbone and a well-designed detection component. Moreover, this was achieved at a significantly reduced computational cost, amounting to only 40% of the computational resources required by its counterparts.

6.6. EfficientDet

EfficientDet [76] aims to enhance prevailing object detection models, including Single Shot Multibox Detector (SSD) and YOLO (You Only Look Once), by refining

Table 6
Details of LDL Segmentation Models and their Datasets.

Model	Parameters	Top Accuracy	Year	Dataset
ENet [55]	0.37 M	58.3%	2016	Cityscapes, CamVid
ERFNet [62]	–	68.0%	2017	Cityscapes
DSNet [80]	0.9 M	71.8%	2018	Cityscapes
CCC3 [54]	0.198 M	61.96%	2018	Cityscapes
SkipNet-MobileNet [71]	–	61.5%	2018	Cityscapes
SkipNet-ShuffleNet [71]	–	58.3%	2018	Cityscapes
LiteSeg [18]	20.55 M	67.81%	2019	Cityscapes
LAANet [90]	0.67 M	73.6%	2022	Cityscapes
LCNet3_11 [69]	0.74 M	75.8%	2024	Cityscapes

their architecture and scaling strategies. The methodology proposed by the authors involves implementing a compound scaling technique that ensures consistent scaling of the depth, width, and resolution of both the backbone and feature networks. EfficientDet introduced a novel feature fusion mechanism known as BiFPN. The model effectively integrates features from multiple layers and scales to process objects of diverse sizes efficiently. The detection head of EfficientDet was optimized to enhance detection accuracy while minimizing parameter usage. Further, it encompasses a range of model variants, denoted as EfficientDet-D0 to EfficientDet-D7, which exhibit depth and width variations. The minor variants (D0, D1, and D2) exhibited greater computational efficiency, whereas the significant variants (D3 to D7) provided enhanced accuracy, albeit with a concomitant increase in computational complexity. This was accomplished using a relatively small model size of 52 million parameters and a computational cost of 325 billion FLOPs.

6.7. LWUAVDET-S

LWUAVDET-S is a lightweight UAV object detection model designed for real-time performance on edge devices with limited computational power. Similar to ThunderNet, which integrates the MobileNetV3 backbone for efficient feature extraction, LWUAVDET-S incorporates a streamlined CNN-based architecture optimized for aerial imagery. It leverages a multi-path feature fusion module to enhance detection across various object scales, a key requirement for UAV scenarios where small object detection is critical. LWUAVDET-S balances detection accuracy with reduced computational demands, making it highly suitable for deployment on UAV platforms. Drawing from prior advances in lightweight detectors like ThunderNet, which operated at 24.1 FPS on ARM processors while maintaining a low resource footprint, LWUAVDET-S furthers this efficiency trend and delivers robust object detection in aerial environments [52].

7. Lightweight Segmentation Models

Lightweight segmentation models are a specialized subset of image segmentation techniques designed to operate effectively under constraints of memory and computational resources, making them particularly suited for real-time applications. Most lightweight segmentation models adopt encoder-decoder architectures, where the encoder extracts high-level features and the decoder reconstructs the segmentation maps. table 6 shows a detailed summary of these architectures. While popular lightweight segmentation models are dicuss below.

7.1. Enet

The Efficient Neural Network (Enet) [55] is a compelling architecture that was carefully developed to minimize memory consumption and computational complexity, as shown in fig. 12. This is a breakthrough DNN architecture for performing real-time semantic segmentation for a resource-constrained device. It was designed for low-latency operations. Compared to other lightweight models. Enet achieve speeds up to 18 times faster, 75 times fewer FLOPs and 79 times fewer parameters, and comparable or superior accuracy. Enet was trained and validated using CamVid, Cityscapes, and SUN datasets. It combines skip connections, max-pooling, and conventional convolution for feature extraction.

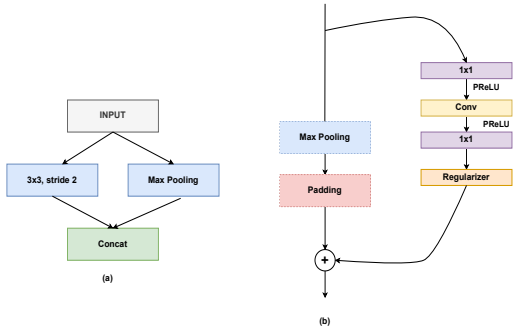


Figure 12: (a) ENet initial block (b) ENet bottleneck module. [55].

7.2. ERFNet

The ERFNet [62] architecture is a highly advanced model that can execute real-time operations while achieving accurate semantic segmentation. The architectural structure was constructed sequentially by stacking layers, informed by our innovative redesign of the residual layer.

Residual layers were first introduced in [66] and have since been extensively incorporated into various state-of-the-art architectures. The two designs proposed by the authors in [66] are bottleneck and non-bottleneck designs. Both approaches share similarities; however, a bottleneck variant is frequently employed to enhance efficiency, as exemplified by ENet. This variant achieves efficiency by internally reducing the number of computed feature maps, thereby reducing the overall computational burden. However, the absence of a bottleneck design can benefit shallow versions of ResNet, such as our architecture. This is because they have an increased internal "width" regarding feature dimensions compared with the bottleneck version. While dense-prediction tasks such as semantic segmentation may experience reduced efficiency, they can still benefit significantly from incorporating additional width.

7.3. DSNet

A novel lightweight semantic architecture, driving segmentation network (DSNet), was introduced for real-time diving scenes [80]. The DSNet architecture provides accurate results for low latency performance because it is efficient and powerful. Additionally, the authors introduced a novel loss function known as Object Weighted Focal Loss. The objective of the DSNet design was to maximize capacity while maintaining a limited level of model complexity. The DSNet model has 9M parameters and achieves a mean IoU of 71.8% and 69.3% on the Cityscapes validation and test sets, respectively. The experimental results demonstrate that DSNet achieved a 2.7% improvement in performance on minor and complex objects compared with the fast-and-accurate ERFNet model while maintaining a similar level of accuracy. A diagram of the proposed DSNet is shown in fig. 13. Spatial resolution values, such as 1024×2048, provided as an example input, demonstrate that the network can process images of varying sizes.

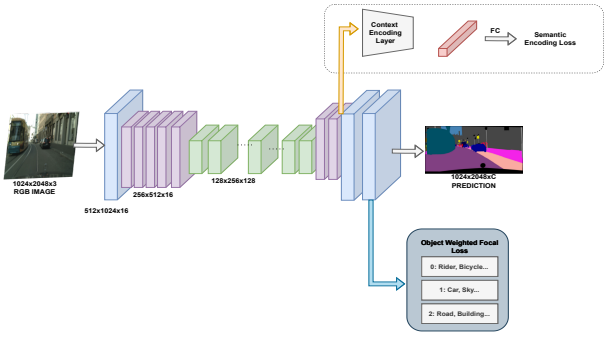


Figure 13: Proposed architecture of DSNet [80].

7.4. CCC3

A novel block, referred to as Concentrated-Comprehensive Convolution (C3) [54], was proposed to decrease the complexity of segmentation models while maintaining their original accuracy. This block incorporated asymmetric convolutions before depth-wise separable dilated convolution to reduce the information loss resulting from the dilated convolution. Applying C3 to ESPNet yielded a performance enhancement of approximately 2%. Furthermore, it exhibited a 50% reduction in the number of parameters and a 35% decrease in the number of FLOPs compared with the initial ESPNet architecture. An improved lightweight model utilizing the proposed C3 block can be developed based on ESPNet. However, it was demonstrated that the proposed C3 block can restore this feature activation, similar to the original model. This was confirmed by using Grad-CAM heatmap [66]. The proposed structure of the C3 module is shown in fig. 14.

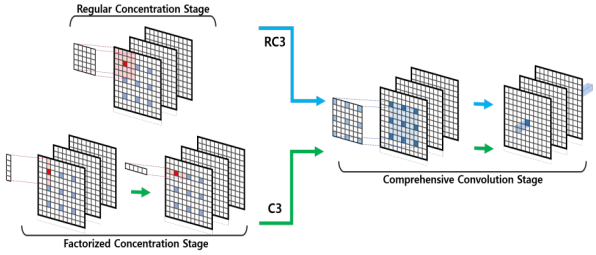


Figure 14: Structure of C3 block with regular depth-wise convolution (RC3) and Structure of C3 by factorizing concentration stage (C3) [54].

7.5. RTSEG

The segmentation architecture can be modularly decoupled into two components: feature extraction and decoding method, as discussed in a real-time semantic segmentation comparative study (RTSEG) [71]. This module decoupling is referred to as meta-architecture, as illustrated in fig. 15. The proposed system comprises multiple meta-architectures for segmentation that establish a methodology for decoding. As mentioned above, the network architecture encompasses SkipNet, UNet, and Dilation Frontend. Implementing a modular design facilitates the emergence of innovative architectures, resulting in a significant reduction of 143 times in GFLOPs compared to SegNet. A comprehensive ablation study that examines the relationship between accuracy and computational efficiency, which enabled the development of two innovative segmentation architectures utilized: MobileNet [24], and ShuffleNet [86] and incorporated multiple decoding methods.

7.6. LiteSeg

To enhance the efficiency and efficacy of semantic image segmentation, a novel lightweight architecture, LiteSeg, was developed [18]. This study introduces an enhanced iteration of the Atrous Spatial Pyramid Pooling (ASPP) module, incorporating depthwise separable convolution with short and

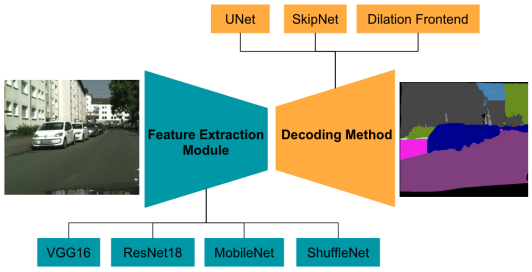


Figure 15: Overview architecture of the various components within the framework of RTSEG, specifically focusing on the decoupling of the feature extraction module and the decoding method [71].

long residual connections. The encoder-decoder architecture was combined with depthwise separable convolution and dilated convolution. LiteSeg has a unique design that can be adapted for any backbone network, as shown in fig. 16. The LiteSeg architecture has been introduced and evaluated on multiple backbone networks, such as ShuffleNet, MobileNet, and Darknet19, to provide various options regarding computational cost and accuracy trade-offs. The proposed LiteSeg model achieved a mean IoU accuracy of 70.75%, 67.81%, and 65.17% when utilizing Darknet, MobileNetV2, and ShuffleNet as backbone networks, respectively. This accuracy was achieved at a frame rate of 161 FPS and a resolution of 640 × 360 on the Cityscapes dataset.

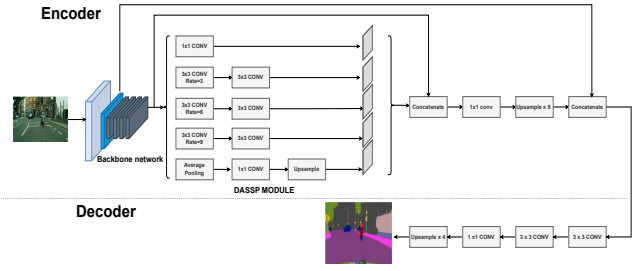


Figure 16: LiteSeg figure showing the encoder module, DASPP module, backbone network, and decoder module. An input picture is fed into the encoder module, which outputs a high-dimensional feature vector. The decoder module recovers this feature vector spatial information [18].

7.7. LAANet

LAANet [90] was introduced with two components encoder and decoder. An efficient asymmetric bottleneck (EAB) module was designed within the encoder. The EAB module comprises two branches: One utilizes a depth-wise asymmetric convolution to capture local information. By contrast, the other branch employs depth-wise dilated convolution to capture contextual information. The incorporation of the attention mechanism is observed in an enhanced version of the ASPP module, known as the attention-guided dilated pyramid pooling (ADPP) module. This module facilitated the selection of relevant features during the integration of multi-scale contextual information. Segmentation objects of different sizes can be effectively addressed by incorporating context information from the ADPP module. The proposed ADPP module improves the precision of the segmentation of smaller targets. The ADPP module is one of the two attention-guided modules created for the decoder. The second module is an attention-guided feature fusion upsampling (AFFU) module. This module effectively combines features at various levels during upsampling, using the suggested improved channel attention (ICA) and improved spatial attention (ISA) modules.

7.8. LCNET

LCNet is a lightweight CNN tailored for semantic segmentation tasks on resource-constrained devices. The network is optimized for scenarios requiring efficient and accurate segmentation, such as UAV imagery analysis or edge-based video understanding. LCNet introduces a compact encoder-decoder architecture that leverages lightweight convolutional blocks to reduce the number of parameters and computational overhead while preserving fine-grained spatial information crucial for segmentation. Moreover, LCNet integrates an efficient context aggregation module to enhance feature representations and improve boundary delineation. As demonstrated in [69], LCNet achieves a favorable trade-off between segmentation accuracy and processing efficiency, making it particularly well-suited for real-time applications on embedded platforms where power and memory limitations are critical.

8. Applications of LDL Models

LDL models are specifically engineered to address the challenges of deploying AI solutions on embedded, resource-constrained edge and mobile devices. These models are optimized to operate under stringent computational power, memory capacity, and energy consumption limitations, making them ideal for real-time and on-device applications.

8.0.1. Edge Devices

Edge devices are typically small, low-power devices deployed in environments where cloud-based processing may be infeasible due to latency, privacy, or connectivity constraints. Such devices include smartphones, IoT systems, embedded sensors, and autonomous platforms. Below are key applications where lightweight models have been effectively deployed on edge devices:

- **Real-time Object Detection:** Edge devices equipped with cameras use LDL models to detect and classify objects in real time, which is critical for applications such as security surveillance, industrial automation, and autonomous navigation.
- **Anomaly Detection:** LDL models enable on-device monitoring of sensor streams in industrial IoT (IIoT) systems, allowing for early detection of equipment malfunctions, security breaches, or abnormal system behaviors without the need for continuous cloud communication.

8.0.2. Mobile Devices

LDL models are well-suited for on-device inference, minimizing the dependency on cloud infrastructure while enabling low-latency processing and improved data privacy. Some prevalent applications of lightweight models on mobile devices include:

- **Image and Video Processing:** Lightweight models support a range of applications, including augmented reality (AR) overlays, interactive camera filters, and video enhancement, all processed directly on mobile devices to ensure real-time performance.
- **Speech Recognition and NLP:** Mobile devices leverage LDL models for tasks such as speech-to-text conversion, voice assistant functionalities, and real-time language translation. These models allow users to perform voice-based interactions and natural language tasks without requiring persistent internet connectivity.

9. Experimental Analysis

This section presents a comprehensive experimental evaluation of recent LDL models, with a particular focus on their performance on resource-constrained edge devices. The objective of this study is to systematically examine the trade-offs between model accuracy and inference speed, which are critical factors for the deployment of real-time AI applications in environments with limited computational resources. The experimental analysis was conducted using the NVIDIA Jetson Orin NX 16GB platform, which represents the latest generation of edge computing devices specifically designed to support AI workloads in embedded and mobile systems. The hardware configuration consists of a reComputer J401 carrier board, equipped with 128GB NVMe solid-state storage, an integrated aluminum heatsink and fan for thermal management, and a 12V/5A power adapter to ensure stable power delivery. The Jetson Orin NX is particularly well-suited for real-time AI tasks and is deployable on UAV platforms such as the Holybro X500 drone, making it an ideal testbed for LDL models. To evaluate the performance of lightweight models, several state-of-the-art architectures were selected, including YOLOv8 (small), YOLOv11 (small), VL2Lite, and MobileSAM. The experimental results are summarized in table 7, which reports the accuracy and inference time of each model for the respective tasks. The findings demonstrate notable variations in both accuracy and computational efficiency across different models and tasks.

10. Future Trends

Building on the comprehensive discussions presented in the preceding sections covering LDL models for image classification, object detection, and segmentation, as well as associated model compression techniques, this section delineates prospective directions aimed at enhancing the practical deployment of LDL models in real-world, resource-constrained environments.

10.1. Lightweight Transformer-Based Models

Transformer-based architectures, although dominant in natural language processing and increasingly prominent in computer vision, are computationally intensive due to their reliance on self-attention mechanisms and large token embeddings. Recent efforts such as MobileViT [51], EdgeFormer [20], and EfficientFormer [37] have introduced hybrid CNN-Transformer models and hierarchical token reduction strategies to reduce parameter count and inference latency. MobileViT integrates convolutional layers with vision transformers to preserve locality and reduce computation, while EfficientFormer employs early-stage token downsampling to minimize complexity of self-attention. Despite these advances, most lightweight transformers still lag behind CNNs in deployment readiness, particularly in terms of quantization robustness and interpretability. A major limitation lies in the sensitivity of transformer-based models to aggressive quantization and pruning, which often significantly degrade performance more than in CNN-based architectures.

10.2. Model Compression Techniques

Although model compression methods such as structured pruning [36], quantization [85], and knowledge distillation [21] have achieved considerable success, they still face substantial limitations in edge deployment. Structured pruning often leads to unbalanced channel distribution and sparse memory access patterns that are inefficient on general-purpose hardware. Quantization techniques degrade rapidly under ultra-low bit widths (e.g., INT4 or binary), especially for models with sensitive attention components. Furthermore, most distillation approaches assume availability of full teacher models and datasets, which is impractical in privacy-sensitive or bandwidth-limited environments. Future research should emphasize on energy-aware pruning guided by gradient flow and attention maps, layer-wise mixed-precision quantization adapted to activation distributions, and task-aware distillation frameworks that preserve functional alignment between student and teacher networks without relying on full model access.

11. Conclusion

Lightweight deep learning (LDL) models have demonstrated outstanding performance with continued increase in their applications in the real world, particularly when deploying models on edge devices, resource-constrained devices, and embedded systems. LDL models are a significant area of study that addresses resource-intensive challenges associated with deep neural networks (DNNs). This study reviewed the most recent literature on LDL models and provides a comprehensive overview of the metrics and datasets, convolutional neural network (CNN)

Table 7
Accuracy and Inference Time (ms) of Lightweight Models on Jetson Orin NX

Device	Model	Task (Dataset)	Accuracy (%)	Inference Time (ms)
Jetson Orin NX	YOLOv8 (small)	Image Classification (ImageNet)	65.4	24.1
	YOLOv11 (small)	Image Classification (ImageNet)	69.8	26.4
	VL2Lite	Image Classification (ImageNet)	82.41	35.6
	YOLOv8 (small)	Object Detection (VisDrone)	55.1	47.8
	YOLOv11 (small)	Object Detection (VisDrone)	62.9	34.96
	MobileSAM	Object Detection (VisDrone)	49.6	55.25
	YOLOv8 (small)	Segmentation (COCO)	36.7	96.8
	YOLOv11 (small)	Segmentation (COCO)	44.7	155.2
	MobileSAM	Segmentation (COCO)	32.5	168.1

architectures, lightweight model compression techniques, lightweight classification, object detection, segmentation models with model compression techniques, and backbone architectures of lightweight models. Furthermore, the datasets will be discussed by comparing them with the latest YOLOv8 models in the same category. Finally, the model tested a benchmark dataset on the Orin Jetson edge device, which was used to evaluate and test the results of lightweight models.

12. Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government. (MSIT) (No. 2022R1A2B5B01001994).

References

[1] Aghli, N., Ribeiro, E., 2021. Combining weight pruning and knowledge distillation for cnn compression, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 3191–3198.

[2] Aguilar, G., Ling, Y., Zhang, Y., Yao, B., Fan, X., Guo, C., 2020. Knowledge distillation from internal representations, in: Proceedings of the AAAI conference on artificial intelligence, pp. 7350–7357.

[3] Alswayed, A.S., Alhichri, H.S., Bazi, Y., 2020. Squeezenet with attention for remote sensing scene classification, in: 2020 3rd International Conference on Computer Applications & Information Security (ICCAIS), IEEE. pp. 1–4.

[4] Ayachi, R., Said, Y., Atri, M., 2021. A convolutional neural network to perform object detection and identification in visual large-scale data. Big Data 9, 41–52.

[5] Baker, B., Gupta, O., Naik, N., Raskar, R., 2016. Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167 .

[6] Brateanu, A., Balmez, R., Avram, A., Orhei, C., Ancuti, C., 2025. Lyt-net: Lightweight yuv transformer-based network for low-light image enhancement. IEEE Signal Processing Letters 32, 2065–2069. doi:10.1109/LSP.2025.3563125.

[7] Chang, J., Guo, Y., Meng, G., Xiang, S., Pan, C., et al., 2019. Data: Differentiable architecture approximation. Advances in Neural Information Processing Systems 32.

[8] Chen, F., Li, S., Han, J., Ren, F., Yang, Z., 2024. Review of lightweight deep convolutional neural networks. Archives of Computational Methods in Engineering 31, 1915–1937.

[9] Chen, J., Cheung, W.K., 2019. Similarity preserving deep asymmetric quantization for image retrieval, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 8183–8190.

[10] Chen, X., Xie, L., Wu, J., Tian, Q., 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 1294–1303.

[11] Cheng, H., Zhang, M., Shi, J.Q., 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. IEEE Transactions on Pattern Analysis and Machine Intelligence .

[12] Cheng, Y., Wang, D., Zhou, P., Zhang, T., 2018. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. IEEE Signal Processing Magazine 35, 126–136.

[13] Chin, T.W., Ding, R., Zhang, C., Marculescu, D., 2020. Towards efficient model compression via learned global ranking, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1518–1528.

[14] Cui, J., Chen, P., Li, R., Liu, S., Shen, X., Jia, J., 2019. Fast and practical neural architecture search, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 6509–6518.

[15] Deng, L., Li, G., Han, S., Shi, L., Xie, Y., 2020. Model compression and hardware acceleration for neural networks: A comprehensive survey. Proceedings of the IEEE 108, 485–532.

[16] Deng, Y., 2019. Deep learning on mobile devices: a review. Mobile Multimedia/Image Processing, Security, and Applications 2019 10993, 52–66.

[17] Elsken, T., Metzen, J.H., Hutter, F., 2019. Neural architecture search: A survey. Journal of Machine Learning Research 20, 1–21.

[18] Emara, T., Abd El Munim, H.E., Abbas, H.M., 2019. Liteseg: A novel lightweight convnet for semantic segmentation, in: 2019 digital image computing: Techniques and applications (DICTA), IEEE. pp. 1–7.

[19] Fang, J., Sun, Y., Zhang, Q., Li, Y., Liu, W., Wang, X., 2020. Densely connected search space for more flexible neural architecture search, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10628–10637.

[20] Ge, T., Chen, S.Q., Wei, F., 2022. Edgeformer: A parameter-efficient transformer for on-device seq2seq generation. arXiv preprint arXiv:2202.07959 .

[21] Gou, J., Yu, B., Maybank, S.J., Tao, D., 2021. Knowledge distillation: A survey. International Journal of Computer Vision 129, 1789–1819.

[22] Graham, B., El-Nouby, A., Touvron, H., Stock, P., Joulin, A., Jégou, H., Douze, M., 2021. Levit: a vision transformer in convnet's clothing for faster inference, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 12259–12269.

[23] Gray, R.M., Neuhoff, D.L., 1998. Quantization. IEEE transactions on information theory 44, 2325–2383.

[24] Haase, D., Amthor, M., 2020. Rethinking depthwise separable convolutions: How intra-kernel correlations lead to improved mobilenets, in: Proceedings of the IEEE/CVF conference on computer vision and

pattern recognition, pp. 14600–14609.
[25] Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C., 2020. Ghostnet: More features from cheap operations, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1580–1589.
[26] He, Y., Xiao, L., 2023. Structured pruning for deep convolutional neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence 46, 2900–2919.
[27] Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 .
[28] Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al., 2019. Searching for mobilenetv3, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 1314–1324.
[29] Hu, P., Peng, X., Zhu, H., Aly, M.M.S., Lin, J., 2021. Opq: Compressing deep neural networks with one-shot pruning-quantization, in: Proceedings of the AAAI conference on artificial intelligence, pp. 7780–7788.
[30] Huang, Z., Shen, X., Xing, J., Liu, T., Tian, X., Li, H., Deng, B., Huang, J., Hua, X.S., 2021. Revisiting knowledge distillation: An inheritance and exploration framework, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3579–3588.
[31] Jiang, Y., Hu, C., Xiao, T., Zhang, C., Zhu, J., 2019. Improved differentiable architecture search for language modeling and named entity recognition, in: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp. 3585–3590.
[32] Kim, M.S., Del Barrio, A.A., Kim, H., Bagherzadeh, N., 2021. The effects of approximate multiplication on convolutional neural networks. IEEE Transactions on Emerging Topics in Computing 10, 904–916.
[33] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. Imagenet classification with deep convolutional neural networks. Communications of the ACM 60, 84–90.
[34] Ku, J.H., Oh, J., Lee, Y., Pooniwala, G., Lee, S., 2020. A selective survey on versatile knowledge distillation paradigm for neural network models. arXiv preprint arXiv:2011.14554 .
[35] Li, F., Liu, B., Wang, X., Zhang, B., Yan, J., 2016a. Ternary weight networks. arXiv preprint arXiv:1605.04711 .
[36] Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P., 2016b. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 .
[37] Li, Y., Yuan, G., Wen, Y., Hu, J., Evangelidis, G., Tulyakov, S., Wang, Y., Ren, J., 2022. Efficientformer: Vision transformers at mobilenet speed. Advances in Neural Information Processing Systems 35, 12934–12949.
[38] Li, Z., Li, H., Meng, L., 2023. Model compression for deep neural networks: A survey. Computers 12, 60.
[39] Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., Sun, J., 2017. Light-head r-cnn: In defense of two-stage object detector. arXiv preprint arXiv:1711.07264 .
[40] Liang, T., Glossner, J., Wang, L., Shi, S., Zhang, X., 2021. Pruning and quantization for deep neural network acceleration: A survey. Neurocomputing 461, 370–403.
[41] Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L., 2020. Hrank: Filter pruning using high-rank feature map, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1529–1538.
[42] Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K., 2018a. Progressive neural architecture search, in: Proceedings of the European conference on computer vision (ECCV), pp. 19–34.
[43] Liu, D., Kong, H., Luo, X., Liu, W., Subramaniam, R., 2022. Bringing ai to edge: From deep learning's perspective. Neurocomputing 485, 297–320.
[44] Liu, H., Simonyan, K., Yang, Y., 2018b. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 .
[45] Liu, H.I., Galindo, M., Xie, H., Wong, L.K., Shuai, H.H., Li, Y.H., Cheng, W.H., 2024a. Lightweight deep learning for resource-constrained environments: A survey. ACM Computing Surveys 56, 1–42.
[46] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. Ssd: Single shot multibox detector, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14, Springer. pp. 21–37.
[47] Liu, Y., Xue, J., Li, D., Zhang, W., Chiew, T.K., Xu, Z., 2024b. Image recognition based on lightweight convolutional neural network: Recent advances. Image and Vision Computing , 105037.
[48] Luo, J.H., Wu, J., 2020. Neural network pruning with residual-connections and limited-data, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1458–1467.
[49] 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 (ECCV), pp. 116–131.
[50] Malinin, A., Mlodozeniec, B., Gales, M., 2019. Ensemble distribution distillation. arXiv preprint arXiv:1905.00076 .
[51] Mehta, S., Rastegari, M., 2021. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. arXiv preprint arXiv:2110.02178 .
[52] Min, X., Zhou, W., Hu, R., Wu, Y., Pang, Y., Yi, J., 2024. Lwuavdet: A lightweight uav object detection network on edge devices. IEEE Internet of Things Journal .
[53] Nekrasov, V., Chen, H., Shen, C., Reid, I., 2019. Fast neural architecture search of compact semantic segmentation models via auxiliary cells, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 9126–9135.
[54] Park, H., Yoo, Y., Seo, G., Han, D., Yun, S., Kwak, N., 2018. C3: Concentrated-comprehensive convolution and its application to semantic segmentation. arXiv preprint arXiv:1812.04920 .
[55] Paszke, A., Chaurasia, A., Kim, S., Culurciello, E., 2016. Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147 .
[56] Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J., 2018. Efficient neural architecture search via parameters sharing, in: International conference on machine learning, PMLR. pp. 4095–4104.
[57] Qiang, B., Zhai, Y., Zhou, M., Yang, X., Peng, B., Wang, Y., Pang, Y., 2021. Squeezenet and fusion network-based accurate fast fully convolutional network for hand detection and gesture recognition. IEEE Access 9, 77661–77674.
[58] Qin, Z., Li, Z., Zhang, Z., Bao, Y., Yu, G., Peng, Y., Sun, J., 2019. Thundernet: Towards real-time generic object detection on mobile devices, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 6718–6727.
[59] Real, E., Aggarwal, A., Huang, Y., Le, Q.V., 2019. Regularized evolution for image classifier architecture search, in: Proceedings of the aaai conference on artificial intelligence, pp. 4780–4789.
[60] Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788.
[61] Redmon, J., Farhadi, A., 2017. Yolo9000: better, faster, stronger, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263–7271.
[62] Romera, E., Alvarez, J.M., Bergasa, L.M., Arroyo, R., 2017. Efficient convnet for real-time semantic segmentation, in: 2017 IEEE Intelligent Vehicles Symposium (IV), IEEE. pp. 1789–1794.
[63] Ronald, M., Poulose, A., Han, D.S., 2021. isplinception: An inception-resnet deep learning architecture for human activity recognition. IEEE Access 9, 68985–69001. doi:10.1109/ACCESS.2021.3078184.
[64] Saikia, T., Marrakchi, Y., Zela, A., Hutter, F., Brox, T., 2019. Autodispnet: Improving disparity estimation with automl, in: Proceedings of the ieee/cvf international conference on computer vision, pp. 1812–1823.

[65] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4510–4520.
[66] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., 2020. Grad-cam: visual explanations from deep networks via gradient-based localization. International journal of computer vision 128, 336–359.
[67] Sen, J., Dasgupta, S., 2023. Adversarial attacks on image classification models: Fgsm and patch attacks and their impact. arXiv preprint arXiv:2307.02055 .
[68] Shafiq, M., Gu, Z., 2022. Deep residual learning for image recognition: A survey. Applied sciences 12, 8972.
[69] Shi, M., Lin, S., Yi, Q., Weng, J., Luo, A., Zhou, Y., 2024. Lightweight context-aware network using partial-channel transformation for real-time semantic segmentation. IEEE Transactions on Intelligent Transportation Systems 25, 7401–7416.
[70] Shrivastava, A., Gupta, A., Girshick, R., 2016. Training region-based object detectors with online hard example mining, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 761–769.
[71] Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., Jagersand, M., 2018. Rtseg: Real-time semantic segmentation comparative study, in: 2018 25th IEEE International Conference on Image Processing (ICIP), IEEE. pp. 1603–1607.
[72] Sriraman, H., Ravikumar, A., 2023. Customized fpga design and analysis of soft-core processor for dnn. Procedia Computer Science 218, 469–478.
[73] Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S., 2017. Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE 105, 2295–2329.
[74] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V., 2019. Mnasnet: Platform-aware neural architecture search for mobile, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2820–2828.
[75] Tan, M., Le, Q., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, PMLR. pp. 6105–6114.
[76] Tan, M., Pang, R., Le, Q.V., 2020. Efficientdet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10781–10790.
[77] Ullah, A., Elahi, H., Sun, Z., Khatoon, A., Ahmad, I., 2022. Comparative analysis of alexnet, resnet18 and squeezenet with diverse modification and arduous implementation. Arabian journal for science and engineering 47, 2397–2417.
[78] Wang, C.H., Huang, K.Y., Yao, Y., Chen, J.C., Shuai, H.H., Cheng, W.H., 2022a. Lightweight deep learning: An overview. IEEE consumer electronics magazine 13, 51–64.
[79] Wang, R.J., Li, X., Ling, C.X., 2018. Pelee: A real-time object detection system on mobile devices. Advances in neural information processing systems 31.
[80] Wang, W., Fu, Y., Pan, Z., Li, X., Zhuang, Y., 2020. Real-time driving scene semantic segmentation. IEEE Access 8, 36776–36788.
[81] Wang, Y., Wang, J., Zhang, W., Zhan, Y., Guo, S., Zheng, Q., Wang, X., 2022b. A survey on deploying mobile deep learning applications: A systemic and technical perspective. Digital Communications and Networks 8, 1–17.
[82] Womg, A., Shafiee, M.J., Li, F., Chwyl, B., 2018. Tiny ssd: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection, in: 2018 15th Conference on computer and robot vision (CRV), IEEE. pp. 95–101.
[83] Wu, B., Iandola, F., Jin, P.H., Keutzer, K., 2017. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving, in: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 129–137.
[84] Yang, J., Liao, J., Lei, F., Liu, M., Chen, J., Long, L., Wan, H., Yu, B., Zhao, W., 2023. Tinyformer: Efficient transformer design and deployment on tiny devices. arXiv preprint arXiv:2311.01759 .
[85] Yang, J., Shen, X., Xing, J., Tian, X., Li, H., Deng, B., Huang, J., Hua, X.s., 2019. Quantization networks, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 7308–7316.
[86] Yun, S., Wong, A., 2021. Do all mobilenets quantize poorly? gaining insights into the effect of quantization on depthwise separable convolutional networks through the eyes of multi-scale distributional dynamics, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2447–2456.
[87] Zaidi, S.S.A., Ansari, M.S., Aslam, A., Kanwal, N., Asghar, M., Lee, B., 2022. A survey of modern deep learning based object detection models. Digital Signal Processing 126, 103514.
[88] Zaman, K.S., Reaz, M.B.I., Ali, S.H.M., Bakar, A.A.A., Chowdhury, M.E.H., 2021. Custom hardware architectures for deep learning on portable devices: A review. IEEE Transactions on Neural Networks and Learning Systems 33, 6068–6088.
[89] Zeiler, M.D., Fergus, R., 2014. Visualizing and understanding convolutional networks, in: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13, Springer. pp. 818–833.
[90] Zhang, X., Du, B., Wu, Z., Wan, T., 2022. Laanet: lightweight attention-guided asymmetric network for real-time semantic segmentation. Neural Computing and Applications 34, 3573–3587.
[91] Zhang, Y., Chen, H., Chen, X., Deng, Y., Xu, C., Wang, Y., 2021. Data-free knowledge distillation for image super-resolution, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7852–7861.
[92] Zheng, W., Lu, S., Yang, Y., Yin, Z., Yin, L., 2024. Lightweight transformer image feature extraction network. PeerJ Computer Science 10, e1755.
[93] Zheng, X., Ji, R., Tang, L., Zhang, B., Liu, J., Tian, Q., 2019. Multinomial distribution learning for effective neural architecture search, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 1304–1313.
[94] Zhu, F., Gong, R., Yu, F., Liu, X., Wang, Y., Li, Z., Yang, X., Yan, J., 2020. Towards unified int8 training for convolutional neural network, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1969–1979.
[95] Zoph, B., Le, Q.V., 2016. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 .
[96] Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V., 2018. Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8697–8710.
[97] Zou, Z., Chen, K., Shi, Z., Guo, Y., Ye, J., 2023. Object detection in 20 years: A survey. Proceedings of the IEEE 111, 257–276.



Syed Muhammad Raza is studying his graduate (M.S.) degree majoring in IT Convergence Engineering in 2023 at WENS Lab, Kumoh National Institute of Technology (KIT), South Korea. Currently, he is working as a graduate fellow and researcher at KIT. His research interests mainly include image processing, lightweight deep learning models, and Unmanned Vehicles (UxV)



Syed Murtaza Hussain Abidi is pursuing a Master's degree in IT Convergence Engineering at the Wireless and Emerging Network Systems Laboratory (WENS Lab), Kumoh National Institute of Technology (KIT), South Korea. Currently, he is a graduate fellow and researcher at KIT. His research interests primarily focus on Computer Vision, Vision-Language Models, and ROS.



Md Masuduzzaman is a Post-doctoral Fellow at McMaster University, Canada. He earned his Ph.D. in IT Convergence Engineering from the Wireless and Emerging Network Systems Laboratory (WENS Lab) at Kumoh National Institute of Technology (KIT), South Korea. He also holds a B.Sc. and M.Sc. in Computer Science from the American International University-Bangladesh (AIUB), where he later served as a lecturer for four years. His research interests encompass Blockchain, Deep Learning, Internet of Things (IoT), Network Security, and Unmanned Vehicles (UxVs).



Soo Young Shin (M'07–SM'17) received his Ph.D. in electrical engineering and computer science from Seoul National University in 2006. He served at Samsung Electronics (2007–2010) and joined Kumoh National Institute of Technology, where he is an Associate Professor. He was a postdoctoral researcher at the University of Washington (2007) and a visiting scholar at the University of British Columbia (2017). His research interests include future wireless communications, signal processing, unmanned mobility, quantum neural network, Internet of things, AI/Deep learning applications, etc.