Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Optimizing MobileNet with Knowledge Distillation and Federated Learning: A Comprehensive Survey

FEDERICO DITTARO[1, 2], (Fellow, IEEE), KYANDOGHERE KYAMAKYA[2], AND WITESYAVWIRWA VIANNEY KAMBALE.[2], (Member, IEEE)

[1]Department of Mathematics, Computer Science and Physics, University of Udine, 33100 Udine, Italy (fede.dittaro@gmail.com)
[2] Institute for Smart Systems Technologies, Universitaet Klagenfurt, 9020 Klagenfurt am Wörthersee, Austria (witesyavwirwa.kambale@aau.at)

Corresponding author: kyandoghere.kyamakya@aau.at

ABSTRACT This report provides a comprehensive exploration of the integration of MobileNet, Federated Learning (FL), and Knowledge Distillation (KD), three pivotal innovations driving advancements in machine learning for edge devices and resource-constrained environments. MobileNet, a family of convolutional neural networks designed for efficiency, is optimized for mobile and embedded systems but faces challenges in achieving high performance without compromising computational and memory resources. Federated Learning addresses the growing demand for privacy-preserving training methods by enabling decentralized learning directly on user devices, ensuring that data remains local while collaborative model training occurs across a distributed network. Complementing this, Knowledge Distillation offers an effective strategy to compress large, complex models into smaller, lightweight counterparts, maintaining high accuracy and generalization capabilities, thereby aligning well with MobileNet's design philosophy.
The primary objectives of this report are to investigate two key aspects: first, how Knowledge Distillation techniques can significantly improve the accuracy, efficiency, and deployment readiness of MobileNet models, and second, how Federated Learning frameworks enhance the adaptability and privacy-preserving capabilities of MobileNet in decentralized settings. By delving into the theoretical foundations and practical applications of KD and FL, this report demonstrates their synergistic impact on MobileNet, enabling it to perform effectively in real-world scenarios such as autonomous systems, IoT applications, and mobile vision tasks. Case studies and experiments discussed in this report illustrate substantial performance gains achieved through the integration of KD and FL into MobileNet, including improvements in training efficiency, inference latency, and scalability across heterogeneous devices. Finally, the report outlines key challenges, including optimizing communication efficiency in FL, addressing privacy concerns, and scaling KD for more complex MobileNet variants, while also highlighting future research directions to enhance the synergy between these cutting-edge technologies.

INDEX TERMS MobileNet, Convolutional Neural Networks (CNNs), Deep Learning, Edge Computing, Federated Learning (FL), Knowledge Distillation (KD), Model Compression, Lightweight Neural Networks, Mobile Vision, Neural Architecture Search (NAS), Efficient Deep Learning, Object Detection, Image Classification, Real-time Inference, Embedded Systems.

I. INTRODUCTION

MOBILENET is a family of convolutional neural networks (CNNs) designed with efficiency and compactness in mind, specifically tailored for mobile and edge devices with limited computational resources. Initially introduced in 2017, MobileNet revolutionized deep learning on mobile platforms by employing depthwise separable convolutions. This innovation significantly reduced the number of parameters and computations compared to standard convolutions, enabling MobileNet to achieve a balance between computational efficiency and accuracy. The design philosophy of MobileNet emphasizes trade-offs between accuracy and resource constraints through hyperparameters like the width multiplier and resolution multiplier. Subsequent iterations, including MobileNetV2, MobileNetV3, and the latest MobileNetV4, have introduced advancements like inverted

residuals with linear bottlenecks, automated neural architecture search (NAS), and improved activation functions. These developments have cemented MobileNet's role in applications requiring lightweight models, such as autonomous driving, real-time object detection, and mobile applications. Despite its success, challenges persist, including maintaining accuracy under severe resource constraints and adapting to increasingly complex real-world tasks.

Federated Learning (FL) is a decentralized machine learning paradigm designed to address the challenges of data privacy and distribution. Traditional machine learning often involves aggregating data from various sources into a centralized server for model training. However, this approach raises significant concerns about privacy, security, and compliance with regulations like GDPR. FL mitigates these concerns by enabling collaborative training of models across multiple devices or clients without requiring data centralization. The core principles of FL include local training on distributed data, privacy preservation through the transmission of model updates rather than raw data, and optimized communication strategies to minimize overhead. This paradigm is particularly effective for edge devices, where data is often sensitive and scattered across a vast network of devices. By training on diverse, real-world data directly on the devices, FL ensures that models are both robust and contextually relevant. However, challenges such as non-IID (non-independent and identically distributed) data, communication inefficiencies, and computational limitations on edge devices remain significant barriers to its widespread adoption.

Knowledge Distillation (KD) is a model compression technique that transfers knowledge from a large, complex "teacher" model to a smaller, more efficient "student" model. This process enables the deployment of high-performance models on resource-constrained devices. KD involves training the student model to mimic the output probabilities of the teacher model, thereby inheriting its generalization capabilities. The concept of knowledge in this context extends beyond raw parameter values to include the nuanced mappings between inputs and outputs, encapsulated in the teacher's class probabilities. There are three primary types of KD: Response-Based KD, which focuses on aligning the output probabilities (soft targets) of the teacher and student models; Feature-Based KD, which transfers intermediate feature representations from the teacher to the student, enhancing the student's ability to learn complex patterns; and Relation-Based KD, which captures and transfers relationships between data points or features within the teacher's learned representations. Applications of KD span various domains, including speech recognition, image classification, and autonomous systems. Its ability to improve the efficiency and scalability of models without a significant loss of accuracy makes KD a valuable tool for deploying deep learning models on edge devices.

The integration of FL and KD with MobileNet addresses critical challenges in deploying deep learning on edge devices. FL's decentralized approach aligns seamlessly with MobileNet's lightweight architecture, enabling real-time learning and adaptation on mobile and IoT devices while ensuring data privacy. By leveraging the strengths of both technologies, it becomes possible to train MobileNet models collaboratively across distributed networks, reducing the reliance on centralized data aggregation and enhancing their robustness to diverse data distributions. KD, on the other hand, complements MobileNet by further optimizing its efficiency and accuracy. The compact nature of MobileNet benefits significantly from the knowledge transfer mechanisms of KD, which can distill the strengths of larger models into its lightweight architecture. This synergy is particularly crucial for edge applications where computational resources are scarce, but high performance is non-negotiable. The combined application of FL and KD with MobileNet holds transformative potential for edge computing by ensuring that models are not only efficient and privacy-preserving but also capable of maintaining high accuracy across diverse and dynamic environments. This paper delves deeper into the mechanisms and benefits of this integration, highlighting its implications for real-world applications.

II. MOBILENET: ARCHITECTURE OVERVIEW

MobileNet is a family of convolutional neural networks (CNNs) [1] designed to perform efficiently on devices with limited computational resources. By leveraging depthwise separable convolutions, MobileNet significantly reduces the number of parameters and computational costs compared to traditional architectures. This chapter explores the progression of MobileNet architectures from its initial version (MobileNetV1) to the most recent advancements (MobileNetV4), emphasizing the design improvements introduced in each iteration.

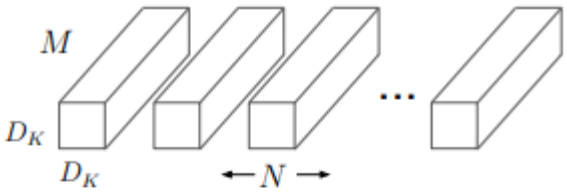
A. MOBILENETV1

The first version of MobileNets was introduced in 2017 [2] and features an efficient network architecture and two hyperparameters designed to create small, low-latency models that meet the design requirements for mobile and embedded vision applications.

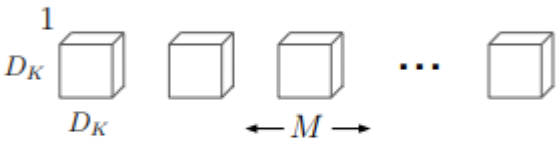
The MobileNetV1 model is based on depthwise separable convolutions, a type of factorized convolution that splits a standard convolution into a depthwise convolution and a 1×1 convolution, referred to as a pointwise convolution. In MobileNetV1, the depthwise convolution applies a single filter to each input channel, while the pointwise convolution combines the outputs of the depthwise convolution through a 1×1 operation. Compared to standard convolutions [57], this factorization drastically reduces the computation and model size. MobileNetV1 uses 3×3 depthwise separable convolutions, which require 8 to 9 times less computation than standard convolutions, with only a small reduction in accuracy.

The architecture is primarily built using depthwise separable convolutions, except for the first layer, which is a full convolution. All layers are followed by batch normalization [6] and

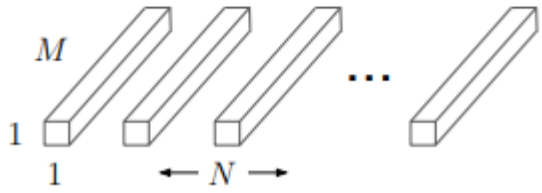
ReLU nonlinearity [58], except for the final fully connected layer, which has no nonlinearity and feeds into a softmax layer [59] for classification. A final average pooling operation reduces the spatial resolution to 1 before the fully connected layer. Counting depthwise and pointwise convolutions as separate layers, MobileNetV1 has a total of 28 layers.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

FIGURE 1. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter. In this picture M represents the number of input channels, N the number of output channels and $D_k \times D_k$ the kernel size. Picture taken from [2].

Although the base MobileNetV1 architecture is already small and low-latency, specific use cases or applications may require an even smaller and faster model. To achieve this, MobileNetV1 introduces two hyperparameters that can be adjusted individually or combined:

- **Width Multiplier**: this hyperparameter thins the network uniformly at each layer, reducing the computational cost and the number of parameters quadratically by roughly α^2. For a given layer and width multiplier α, the number of input channels M becomes αM and the number of output channels N becomes αN where $\alpha \in (0, 1]$.
- **Resolution Multiplier**: this hyperparameter reduces the input image resolution, and subsequently, the internal representations of all layers are scaled by the same factor. The resolution multiplier $\rho \in (0, 1]$ reduces computational cost by ρ^2.

Experimental results on the ImageNet dataset [5], show that MobileNetV1 achieves nearly the same accuracy as VGG16 [4] while being 32 times smaller and 27 times less compute-intensive. It is also more accurate than GoogleNet [3] while being smaller and more than 2.5 times less computationally expensive. Beyond classification, MobileNetV1 has been successfully applied to other tasks, including recognition on the Stanford Dogs dataset [5], large-scale geolocalization, object detection, and face attribute and embedding tasks, achieving significant results.

B. MOBILENETV2

MobileNetV2, introduced in 2019 [7], builds upon the foundation of MobileNetV1 [2] by retaining its simplicity and requiring no specialized operators, while significantly improving accuracy for various image classification and detection tasks designed for mobile applications.
The key innovation in MobileNetV2 is the introduction of the inverted residual with linear bottleneck. This module processes input through three stages:

1) It first expands the low-dimensional compressed representation to a higher dimension.
2) It filters this high-dimensional representation using a lightweight depthwise convolution.
3) Finally, it projects the features back to a low-dimensional space using a linear convolution.

For an input set of real images, the set of layer activations (for any layer L_i) forms a "manifold of interest". There are two properties that are indicative of the requirement that the manifold of interest should lie in a low-dimensional subspace of the higher-dimensional activation space:

- If the manifold of interest remains non-zero volume after ReLU transformation [58], it corresponds to a linear transformation.
- ReLU is capable of preserving complete information about the input manifold, but only if the input manifold lies in a low-dimensional subspace of the input space.

These two insights provide an empirical hint for optimizing the neural architectures: assuming the manifold of interest is low-dimensional it can be captured by inserting linear bottleneck layers into the convolutional blocks. Experiments show that using linear layers is crucial as it prevents non-linearities from destroying too much information while using non-linear layers indeed hurts the performance by several percent.

Figure 2 provides a schematic visualization of the difference between the residual block and the inverted residual block. The motivation for inserting shortcuts is similar to that of classical residual connections [60]: improve the ability of a gradient to propagate across multiplier layers.
One interesting property of the architecture is that it provides a natural separation between the input/output domains of the building blocks (bottleneck layers), and the layer transformation – that is a non-linear function that converts input to the output. The former can be seen as the capacity of the network

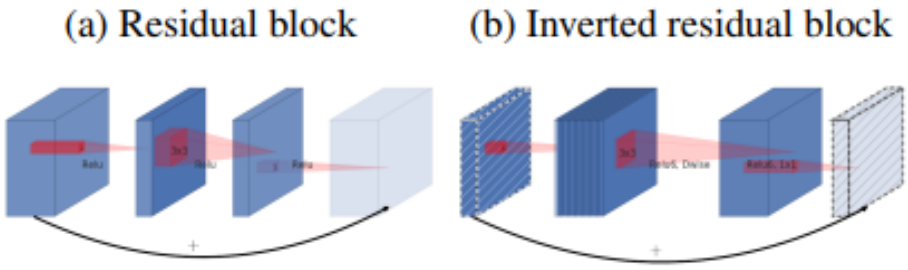


FIGURE 2. The difference between residual block and inverted residual. Diagonally hatched layers do not use non-linearities. It has been used thickness of each block to indicate its relative number of channels. Image taken from [7].

at each layer, whereas the latter as the expressiveness.
The MobileNetV2 architecture contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. It uses ReLU6 as the non-linearity and a kernel size 3×3. With the exception of the first layer, it uses constant expansion rate throughout the network (expansion rates between 5 and 10 result in nearly identical performance curves, with smaller networks being better off with slightly smaller expansion rates and larger networks having slightly better performance with larger expansion rates).
Moreover the inverted residual bottleneck layers allow a particularly memory efficient implementation which is very important for mobile applications. In those layers the total amount of memory would is dominated by the size of bottleneck tensors, rather than the size of tensors that are internal to bottleneck (which number is way larger).
MobileNetV2 has been compared with the first version for different tasks like object detection and semantic segmentation. Table 1 shows the performance of MobileNetV1 [2], ShuffleNet [8] and NASNet-A models [61] on the ImageNet dataset [5].

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Table 1. Performance on ImageNet. As is common practice for ops, it counts the total number of Multiply-Adds. In the last column it is reported the running time in milliseconds (ms) for a single large core of the Google Pixel 1 phone (using TF-Lite). Table taken from [8].

C. MOBILENETV3

MobileNetV3, published in 2019 [9], is defined as two models: MobileNetV3-Large and MobileNetV3-Small. These models are targeted at high and low resource use cases respectively. Differently from the version two, MobileNetV3 introduces:

- Complementary search techniques.
- New efficient versions of non-linearities practical for the mobile setting.
- New efficient network design.
- A new efficient segmentation decoder.

Network search has proven to be a powerful method for discovering and optimizing architectures. For MobileNetV3, platform-aware neural architecture search (NAS) [10] is used to determine the global network structure, while the NetAdapt algorithm optimized the number of filters per layer. Specifically, the following approach is applied:

1) Starts with a seed network architecture found by platform-aware NAS.
2) For each step:
   a) Generate a set of new proposals. Each proposal represents a modification of an architecture that generates at least δ reduction in latency compared to the previous step.
   b) For each proposal use the pre-trained model from the previous step and populate the new proposed architecture, truncating and randomly initializing missing weights as appropriate. Finetune each proposal for T steps to get a coarse estimate of the accuracy.
   c) Selected best proposal according to some metric.
3) Iterate previous step until target latency is reached.

Allowed proposals included reducing the size of expansion layers and bottlenecks in blocks with shared bottleneck sizes to maintain residual connections.
In addition to network search, it also introduces several new components to the model to further improve the final model. It redesign the computationally-expensive layers at the beginning and at the end of the network and introduce a new nonlinearity, h-swish, a modified version of the swish nonlinearity, which is faster to compute and more quantization-friendly. The first modification reworks how the last few layers of the network interact in order to produce the final features more efficiently. MobileNetV2's inverted bottleneck structure and variants use 1x1 convolution as a final layer in

order to expand to a higher-dimensional feature space. This layer is critically important in order to have rich features for prediction. However, this comes at a cost of extra latency. To reduce latency and preserve the high dimensional features, this layer is moved past the final average pooling. This final set of features is now computed at 1x1 spatial resolution instead of 7x7 spatial resolution. The outcome of this design choice is that the computation of the features becomes nearly free in terms of computation and latency. This observation allows also to remove the projection and filtering layers in the previous bottleneck layer, further reducing computational complexity. The original and optimized last stages can be seen in figure 3.

Another expensive layer is the initial set of filters. MobileNetV2 uses 32 filters in a full 3x3 convolution to build initial filter banks for edge detection. By using the hard swish non linearity the number of filters can be reduced to 16 while maintaining the same accuracy as 32 filters.

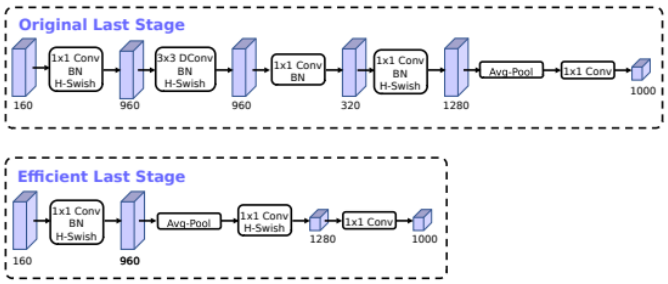


FIGURE 3. Comparison of original last stage and efficient last stage. Picture taken from [9].

MobileNetV3 has been tested on classification, detection and segmentation showing significantly improvements compared to the previous version. As an example, based on the ImageNet dataset [5], figure 4 shows that MobileNetV3-Large and MobileNetV3-Small outperform MnasNet [10], ProxylessNas [11] and MobileNetV2 [7] architectures.

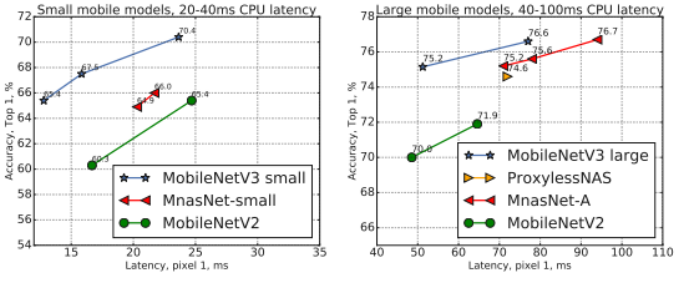


FIGURE 4. The trade-off between Pixel 1 latency and top-1 ImageNet accuracy. Picture taken from [9].

D. MOBILENETV4

The last generation of MobileNets, published in 2024 [12], improves the significant challenge of balancing accuracy and efficiency. To this end, it introduces UIB and Mobile MQA, two innovative building blocks integrated via a refined NAS recipe to create a series of universally mostly-Pareto-optimal mobile models.

The Universal Inverted Bottleneck (UIB) is an advanced architectural component introduced to enhance the efficiency and adaptability of neural networks for mobile devices. It builds upon the Inverted Bottleneck (IB) concept from MobileNetV2 [7], integrating additional features and design flexibility to address the computational constraints of diverse hardware platforms. The UIB block is a versatile and unified structure that combines elements of existing micro-architectures, including the standard Inverted Bottleneck, ConvNext blocks [21], and Feed Forward Networks (FFNs) [22]. A key innovation within the UIB is the introduction of the Extra Depthwise (ExtraDW) variant, which extends the network's receptive field and depth while maintaining computational efficiency.

UIB offers remarkable flexibility, allowing it to adapt spatial and channel mixing strategies to optimize performance for specific hardware requirements. By incorporating optional depthwise convolutions both before and after the expansion layer, UIB provides a highly configurable structure that can balance accuracy and latency. This flexibility ensures that the block can achieve optimal computational utilization and enlarge the receptive field as needed without excessive resource demands.

The UIB design significantly improves the operational efficiency of networks on mobile hardware. Its simplified yet robust structure makes it highly compatible with Neural Architecture Search (NAS) frameworks, which leverage UIB to create models that are well-suited to varying computational and memory bandwidth constraints. This approach allows UIB to unify multiple architectural paradigms while introducing its own enhancements, such as the ExtraDW variant. The result is a scalable and efficient building block that ensures neural networks are adaptable to a broad spectrum of mobile devices, including CPUs, GPUs, DSPs, and accelerators like Google EdgeTPU.

To effectively instantiate the UIB blocks, the TuNAS [23] (Two-stage NAS) framework is employed with tailored enhancements for improved performance. Unlike the fixed scaling rules used in architectures like EfficientNet [24], TuNAS adopts per-size searches and custom search spaces to maximize the potential of UIB blocks. This enhanced search strategy mitigates the bias of TuNAS toward smaller filters and expansion factors caused by parameter sharing. It employs a two-stage search process to address variances in parameter counts between UIB's depthwise layers and other search options. The first stage, or coarse-grained search, identifies optimal filter sizes while keeping parameters fixed, using an inverted bottleneck block with a default expansion factor of 4 and a 3x3 depthwise kernel. The second stage, or fine-grained search, refines the configuration of UIB's two depthwise layers, including their presence and kernel size (either 3x3 or 5x5), while maintaining a constant expansion factor. This strategy leads to enhanced efficiency and model quality compared to conventional one-stage search methods.

Block	Top-1	GMACs	MParams	P8 EdgeTPU
UIB	**83.3%**	**6.2**	**33.0**	**2.68 ms**
CN	83.2%	6.9	35.1	2.69 ms
IB	82.3%	6.1	32.4	2.61 ms

Table 2. Comparison between searches using Inverted Bottleneck blocks, ConvNextLike blocks, and full UIB blocks. Table taken from [12].

The second novel block is the Mobile MQA (Multi-Query Attention), a novel attention mechanism designed specifically for mobile accelerators. Its development addresses the need for balancing computational efficiency and memory access in neural networks optimized for mobile devices. Mobile MQA achieves significant speed and efficiency improvements, delivering over a 39% speedup on mobile accelerators compared to traditional Multi-Head Self-Attention (MHSA) [25]. Additionally, it reduces both Memory Access Costs (MACs) and model parameters by more than 25%, making computations faster and more resource-efficient.

Unlike conventional methods that focus solely on minimizing MACs, Mobile MQA enhances the operational intensity of computations, optimizing the ratio of arithmetic operations to memory access. This approach is particularly beneficial for hardware like accelerators, where computational capacity often exceeds memory bandwidth. Mobile MQA employs an innovative approach by sharing keys and values across all query heads, thereby lowering memory bandwidth requirements without compromising accuracy. Asymmetric spatial down-sampling further enhances efficiency by reducing the resolution of keys and values in later stages, a design choice that improves computational efficiency while maintaining high levels of accuracy.

Mobile MQA's compatibility with accelerators is a defining feature. Optimizations such as tailored Einsum operations and lightweight spatial reduction techniques ensure its effectiveness on platforms like Google EdgeTPU and Samsung S23 GPUs. Experimental results show negligible accuracy loss, approximately -0.03% compared to MHSA, even as it significantly improves latency. For example, Mobile MQA reduces latency on the Pixel 7 EdgeTPU from 9.69 ms to 5.16 ms and on the Samsung S23 GPU from 16.46 ms to 15.10 ms. These improvements are complemented by the use of stride-2 spatial reductions, which enhance efficiency by an additional 23%.

Experiments were made on the ImageNet-1K [26] classification dataset and COCO [27] object detection. The first one proves that MobileNetV4 are roughly 2x faster than MobileNetV3 while mantainig the same accuracy level while the second one shows that MobileNetV4 is 23% faster than MobileNetV2. MobileNetV4 was also tested on different platforms, showing that differently from the previous versions (which performs well on CPUs) it achieves mostly-Pareto-optimal performance across CPUs, GPUs, DSPs, the Apple Neural Engine, and Google EdgeTPUs.

III. MOBILENET REAL-WORLD IMPLEMENTATIONS

MobileNet lightweight architecture makes it particularly suitable for applications such as mobile vision tasks, autonomous systems, and sensor-driven applications. This section explores two domains where MobileNet is utilized:

- Applications in autonomous driving and mobile vision tasks.
- Integration in sensor-based systems

By examining these implementations, we highlight the model's effectiveness in delivering high-performance deep learning solutions within constrained environments.

A. APPLICATIONS IN AUTONOMOUS DRIVING AND MOBILE VISION TASKS

The exploration of MobileNet versions demonstrates their effectiveness in various real-world applications, particularly in automotive and transportation-related scenarios. For instance, the enhanced MobileNetV3 architecture is employed in a simultaneous vehicle and lane detection system to facilitate car-following scenarios [16]. This model integrates multi-task learning with advanced techniques such as Atrous Spatial Pyramid Pooling (ASPP) [17] and Feature Pyramid Networks (FPN) [62] to enhance feature extraction, achieving higher accuracy and faster processing speeds compared to previous methods. The application emphasizes its suitability for real-time autonomous driving, addressing challenges like efficient lane and vehicle detection under dynamic conditions.

Similarly, the MobDet3, a lightweight object detection model based on MobileNetV3, is tailored for autonomous driving tasks [15]. By optimizing the model for edge devices, MobDet3 ensures high accuracy and real-time processing capabilities, achieving a remarkable balance between computational efficiency and detection precision. It highlights the adaptability of MobileNet architectures for resource-constrained environments, enhancing the feasibility of deploying advanced object detection systems in modern vehicles.

Another practical implementation is seen in nighttime vehicle detection using the M-YOLO model [14]. By combining MobileNetV2 with YOLOv3 [18], the system addresses the unique challenges of low-visibility conditions. The model demonstrates superior accuracy and performance in constrained environments, making it ideal for embedded systems in vehicles. This integration underlines the flexibility of MobileNet in tackling specific real-world challenges like nighttime traffic safety.

Lastly, cyclist safety is improved through monocular detection systems leveraging MobileNetV2 and EfficientDet Lite [19] [13]. These systems focus on real-time detection of cyclists in urban environments, particularly in high-risk areas like intersections and blind spots of large vehicles. With rapid inference speeds and high detection accuracy, these systems are poised to significantly enhance safety by mitigating collisions and enabling proactive measures in dynamic traffic conditions.

These diverse applications of MobileNet architectures showcase their adaptability and efficiency in addressing complex real-world problems, particularly in enhancing safety, efficiency, and performance in transportation systems.

B. INTEGRATION IN SENSOR-BASED SYSTEMS

Sensor-based systems have become an integral part of modern technological advancements, especially in applications that require real-time data processing, such as Internet of Things (IoT) devices, and edge computing solutions. MobileNet's lightweight architecture makes it particularly well-suited for sensor-driven environments that require efficient, real-time processing with limited computational resources. These environments typically include vision sensors, accelerometers, and other embedded devices that collect and analyze large volumes of data. The combination of MobileNet with sensor-based networks enables intelligent processing while maintaining low power consumption and high accuracy. In IoT applications, MobileNet has demonstrated its capability in various domains, including smart surveillance, wearable health monitoring, and smart home automation. For instance, edge cameras integrated with MobileNet models can perform real-time object detection without relying on cloud-based processing [72] [73] [74], thereby reducing latency and bandwidth usage. Similarly, MobileNet is leveraged in wearable health monitoring systems to classify motion sensor data, facilitating the detection of anomalies such as irregular heartbeats or sudden falls [75]. In smart home environments, MobileNet-based vision sensors enable gesture and facial recognition, improving user interaction and automation [76] [77]. The efficiency of MobileNet is also evident in industrial IoT applications, where it is employed in predictive maintenance and quality control. Vision sensors equipped with MobileNet models analyze industrial components, identifying defects and potential failures before they impact production [78] [79] [80]. In agricultural automation, MobileNet processes multispectral sensor data from drones to assess crop health, detect diseases, and optimize yield predictions [81]. Additionally, in wildlife monitoring, motion-activated camera traps powered by MobileNet classify and log species in remote areas without needing cloud connectivity [82] [83] [84].

IV. COMPARISON OF MOBILENET WITH OTHER LIGHTWEIGHT MODELS

MobileNet has established itself as a leading lightweight deep learning model for mobile and edge applications. However, several other models have been developed with similar objectives, focusing on efficiency, accuracy, and computational cost. This section provides a comparative analysis of MobileNet against other notable lightweight architectures such as ShuffleNet [8] and EfficientNet [24]. The comparison evaluates key performance metrics, including accuracy, model size, computational efficiency, and energy consumption, which are specific for resource-constrained environments.

ShuffleNet is a highly efficient convolutional neural network (CNN) designed specifically for mobile and edge applications. It achieves its efficiency primarily through two key techniques:

- **Pointwise Group Convolution**: reduces computational complexity by limiting the number of connections between input and output channels, making convolutions more efficient.
- **Channel Shuffling**: to maintain representation learning capacity, ShuffleNet introduces a shuffling operation that ensures information is effectively distributed across channels after the grouped convolutions.

These innovations allow ShuffleNet to significantly lower the number of computations while maintaining competitive accuracy making it particularly suitable for applications requiring real-time inference on low-power devices.

On the other hand, EfficientNet is a family of deep learning models that optimize both accuracy and computational efficiency through a method called compound scaling. Unlike traditional architectures that scale depth, width, or resolution independently, EfficientNet uses a unified approach to balance these factors:

- **Compound Scaling**: EfficientNet scales all dimensions of a model—depth (number of layers), width (number of channels), and resolution (input image size)- using a fixed set of scaling coefficients derived through neural architecture search.
- **Mobile Optimization**: EfficientNet models, particularly EfficientNet-B0 to EfficientNet-B3, are designed to perform well on mobile and edge devices by leveraging depthwise separable convolutions and optimized activation functions like Swish [85].

The EfficientNet architecture achieves state-of-the-art accuracy making it a strong contender for mobile AI applications.

A. ACCURACY AND COMPUTATIONAL EFFICIENCY

One of the primary considerations when selecting a lightweight model is the trade-off between accuracy and computational efficiency. The Top-1 accuracy it's a standard benchmark for evaluating classification performance. This metric measures how often the model's top prediction matches the correct label (a higher accuracy indicates better predictive capability but often comes at the cost of increased model complexity). Another benchmark is the number of trainable parameters (measured in millions (M)) which directly impacts the model size and memory footprint. Fewer parameters generally lead to a smaller model that requires less storage and is easier to deploy on devices with limited computational resources. The last parameter that will be taken into account is the Multiply-Adds (MAdds): this metric quantifies the number of multiplication and addition operations (in millions) required to process a single input image. Lower MAdds indicate a more computationally efficient model, making it well-suited for real-time inference on mobile and embedded hardware.

Table 3 summarizes the Top-1 accuracy on the ImageNet dataset [5] along with model complexity in terms of Multiply-Adds (MAdds) and parameter count.

Model	Top-1 Acc	Parameters (M)	MAdds (M)
MobileNetV1	70.9%	4.2	575
MobileNetV2	72.1%	3.4	300
MobileNetV3	75.3%	5.4	219
MobileNetV4	78.4%	6.1	180
ShuffleNet	73.7%	5.4	524
EfficientNet-B0	76.3%	5.3	390

Table 3. Top-1 accuracy, number of Parameters and MAdds on the ImageNet dataset.

B. ENERGY CONSUMPTION AND INFERENCE SPEED

Energy efficiency is another factor for mobile applications. MobileNet models are optimized for low-power devices by leveraging depthwise separable convolutions, which reduce computational overhead. ShuffleNet enhances efficiency using pointwise group convolutions, while EfficientNet applies neural architecture search to achieve an optimal balance. There are two main parameters that are taken into account for evaluating this metrics: the power consumption (W) that represents the amount of electrical power (in watts) required by the model to perform inference (lower power consumption is desirable for battery-operated devices, ensuring longer operational lifetimes without frequent recharging) and inference time which is measured in milliseconds (ms) and indicates the time taken by the model to process a single input image and generate an output prediction. Both of these factors are influenced by the model's architecture, including the number of computations required (MAdds), memory access patterns, and hardware optimizations. Efficient models minimize energy usage while maintaining fast inference speeds. Table 4 summarizes the energy consumption and the inference speed on the ImageNet dataset.

Model	Power Consumption (W)	Inference Time (ms)
MobileNetV1	0.5	113
MobileNetV2	0.35	75
MobileNetV3	0.28	56
MobileNetV4	0.22	48
ShuffleNet	0.4	80
EfficientNet-B0	0.55	110

Table 4. Energy consumption and inference speed on the ImageNet dataset.

C. SUMMARY OF FINDINGS

The comparative analysis presented in the tables highlights key trade-offs between MobileNet, ShuffleNet, and EfficientNet in terms of accuracy, computational efficiency, energy consumption, and communication costs.

- Accuracy vs. Computational Cost: MobileNetV4 and EfficientNet-B0 achieve higher accuracy than other lightweight models but at different computational costs. MobileNetV4 maintains an efficient structure while outperforming earlier MobileNet versions.
- Energy Efficiency and Inference Speed: MobileNet models demonstrate the lowest power consumption and fastest inference times, making them the most suitable for low-power devices. EfficientNet, while accurate, requires higher energy consumption.

MobileNetV4 emerges as the best option for mobile and edge deployments, balancing accuracy, efficiency, and energy consumption.

V. DEFINITION AND PRINCIPLES OF FEDERATED LEARNING

The rapid proliferation of decentralized data sources such as smartphones, IoT devices, and distributed systems has created vast opportunities for machine learning while simultaneously presenting challenges related to privacy, security, and compliance with data protection regulations. *Federated Learning* (FL) is an innovative approach designed to harness the potential of this decentralized data without compromising user privacy. It enables collaborative model training across multiple devices, referred to as clients, coordinated by a central server, without requiring data centralization. In FL, each client retains its local dataset and performs computations to generate updates for a global model maintained by the central server. Only these updates, rather than raw data, are transmitted to the server. This decoupling of model training from direct access to raw data significantly reduces privacy and security risks while maintaining the integrity of the training process.

FIGURE 5. An example application of federated learning for the task of next-word prediction on mobile phones. In this setup, remote devices communicate with a central server periodically to learn a global model. At each communication round, a subset of selected phones performs local training on their user data, and sends these local updates to the server. After incorporating the updates, the server then sends back the new global model to another subset of devices. This iterative training process continues across the network until convergence is reached or some stopping criterion is met. Picture taken from [66].

The key principles of Federated Learning are:

- **Data Decentralization**: FL shifts from the traditional approach of aggregating data in a central repository.

Training occurs locally on distributed devices, ensuring that sensitive data remains on the user's device.

- **Privacy Preservation**: by transmitting only model updates and not the raw data, FL mitigates potential privacy violations. This principle minimizes the attack surface, limiting vulnerabilities to device-level access rather than extending them to a central data repository.
- **Optimized Learning on Real-World Data**: leveraging real-world data from end-user devices ensures that models are trained on diverse and relevant datasets, unlike proxy data typically used in centralized training.
- **Communication Efficiency**: given that devices may have limited connectivity and bandwidth, FL emphasizes reducing communication overhead through strategies such as selective updates and model compression.
- **Adaptability to Real-World Constraints**: FL is designed to accommodate the following unique challenges:
  - Non-IID Data: local datasets often reflect user-specific behaviors, making them non-identically and independently distributed [30].
  - Unbalanced Data: the volume of local data varies significantly between devices, reflecting different usage patterns [63].
  - Massive Distribution: FL involves potentially thousands or millions of devices, each contributing small amounts of data.
  - Intermittent Connectivity: mobile and IoT devices are frequently offline or on low-bandwidth connections, necessitating robust and fault-tolerant communication protocols [29].
- **Federated Optimization**: the optimization process in FL involves balancing computational and communication efficiency:
  - Increased Parallelism: employing more devices in each training round.
  - Enhanced Local Computation: performing more complex computations on each client before communicating with the server.

The FL process typically unfolds in iterative communication rounds: a random subset of devices (clients) is selected at the start of each round, then the central server sends the current global model to these clients that performs a local training using its dataset and computes model updates. The server aggregates the updates from all participating clients and applies them to the global model. This iterative process continues until the global model achieves the desired level of performance. By optimizing the balance between computation on devices and communication between devices and the server, FL achieves efficient and scalable training.

VI. DEFINITION AND PRINCIPLES OF KNOWLEDGE DISTILLATION

Training for tasks like speech and object recognition involves extracting structure from large, highly redundant datasets. While this process demands significant computational resources, it does not require real-time operation. In contrast, deploying models to a wide user base imposes stricter constraints on latency and computational efficiency. Knowledge Distillation (KD) has emerged as a transformative technique in deep learning, facilitating the transfer of knowledge from a large, complex model (the teacher) to a smaller, more efficient model (the student). The teacher model could be an ensemble of independently trained models or a single, very large model trained with robust regularization techniques, such as dropout [34]. After training the teacher model, a specialized training process called "distillation" is used to transfer its knowledge to the smaller model, making it more suitable for deployment.

An important point is to understand which is the meaning of "knowledge" in the context of neural networks models. Knowledge can be interpreted either as the learned parameter values or as the learned mapping from input vectors to output vectors. The first perspective ties knowledge to a specific model structure, making it difficult to transfer. In contrast, the second, more abstract view detaches knowledge from any particular instantiation, enabling greater flexibility.

For cumbersome models trained to classify a large number of classes, the typical objective is to maximize the average log probability of the correct answers. However, during training, the model also assigns probabilities to incorrect answers. While these probabilities may be small, their relative magnitudes reveal valuable insights into how the model generalizes. This information can be leveraged during knowledge distillation, where the small model is trained to generalize in the same way as the larger model. If the cumbersome model generalizes well, a small model trained to mimic its generalization behavior often performs significantly better on test data compared to a small model trained conventionally on the same dataset.

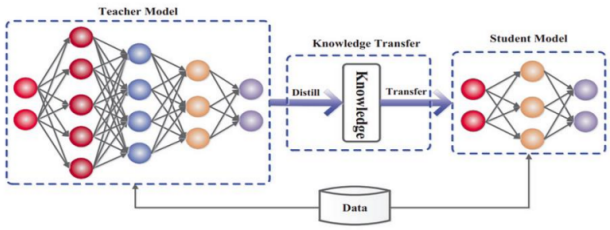


FIGURE 6. Overview of the Teacher–Student Knowledge Distillation Framework.

A practical method for transferring the generalization capability of the cumbersome model is to use its class probabilities as "soft targets" for training the small model. To achieve this, the temperature of the final softmax layer in the cumbersome model is increased, producing a softer set of target probabilities. The same elevated temperature is used when training the small model to align its predictions with these soft targets.

The dataset used for training the small model, known as the transfer set, can either be entirely unlabeled data or the

original training set. Empirical results suggest that using the original training set works effectively, especially when a small term is added to the objective function to encourage the small model to predict both the true targets and the soft targets provided by the cumbersome model.
Specifically Neural networks typically produce class probabilities by using a softmax output layer that converts the logit, z_i, computed for each class into a probability, q_i, by comparing z_i with the other logits using this formula:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

where T is a temperature that is normally set to 1. Using a higher value for T produces a softer probability distribution over classes. In the simplest form of distillation, knowledge is transferred to the distilled model by training it on a transfer set and using a soft target distribution for each case in the transfer set that is produced by using the cumbersome model with a high temperature in its softmax. The same high temperature is used when training the distilled model, but after it has been trained it uses a temperature of 1. When the correct labels are known for all or some of the transfer set, this method can be significantly improved by also training the distilled model to produce the correct labels by simply using a weighted average of two different objective functions:

- The cross entropy with the soft targets: it is computed using the same high temperature in the softmax of the distilled model as was used for generating the soft targets from the cumbersome model.
- The cross entropy with the correct labels: it is computed using exactly the same logits in softmax of the distilled model but at a temperature of 1.

In the recent years have been introduced key innovations in architectures, training paradigms, and application domains. The paper [36] examines recent advancements in KD with the aim to provide researchers and practitioners with a comprehensive understanding of the state-of-the-art in knowledge distillation, bridging foundational concepts with the latest methodologies and practical implications. The main architectures can be summarized as follows (all the techniques can be seen in picture 7):

- **Types of Knowledge Distillation**:
  - Response-Based: is a technique where a student model learns to mimic a teacher model's soft output probabilities, or "soft targets." This approach helps the student model understand the teacher's decision-making process, improving performance and avoiding overfitting. The loss function of response-based KD focuses on matching the output of the teacher and student models. This is done by minimizing the Kullback-Leibler (KL) divergence [69] between the softened output probabilities of the teacher and the student networks.
  - Feature-Based: transfers internal representations from a teacher model to a student model, enabling the student to capture intricate structures and relationships encoded in the teacher's feature maps. This approach provides richer knowledge transfer than simply mimicking output probabilities. The loss function of feature-based KD focuses on aligning the intermediate feature representations between the teacher and student networks.
  - Relation-Based: is an advanced technique that preserves structural relationships and dependencies within data learned by a teacher model. Relation-based KD captures and transfers relational information between different data points or features, such as maintaining pairwise similarities in the teacher's feature space, angular relationships between data points, or the correlation between instances. The loss function focuses on preserving the relational structure of the data learned by the teacher to transfer the relational knowledge between pairs or groups of data points from the teacher network to the student network. After establishing a relational measure between data points, the correlation between the teacher's and student's feature maps is calculated.
- **Knowledge Distillation approaches**:
  - Offline Distillation: is a technique where knowledge is transferred from a pretrained teacher model to a smaller student model during a separate training phase. The offline knowledge distillation process typically involves two main steps. First, a teacher model is pretrained on the dataset. Next, the outputs from this teacher model are used as soft labels to guide the training of a student model. In this stage, the student model learns by minimizing the loss between the true labels and the teacher's soft labels.
  - Online Distillation: is an innovative approach that involves the simultaneous training of multiple model branches in a mutual learning environment. Unlike offline distillation, which requires a pretrained teacher model, online KD allows for continuous feedback and knowledge sharing between peer models during the learning process. The standard process of online knowledge distillation involves training multiple student models concurrently. The outputs of these models are aggregated and compared with the true labels as well as with each other.
  - Self-Distillation: is a variant of knowledge distillation where a model acts as both the teacher and the student, using its own knowledge to guide its learning process. The main superiority of self distillation over teacher-student approach is that no extra teacher is required. Unlike traditional distillation, where a large teacher model transfers knowledge to a smaller student model, self-distillation encourages the model to refine its own predictions

iteratively during training.

VII. METRICS AND EVALUATION OF FD AND KD

Evaluating the effectiveness of both Knowledge Distillation and Federated Learning within the MobileNet framework requires a multidimensional approach. The choice of metrics depends on the objectives of the evaluation, such as accuracy, efficiency, scalability, and robustness. The key performance indicators are:

1) **Accuracy**:
   - **Top-1** and **Top-5 Accuracy**: these metrics evaluate the proportion of correctly classified samples where the true label is among the top predictions. This is particularly important for assessing how KD affects the model's ability to generalize.
   - **Precision, Recall**, and **F1 Score**: these metrics provide insights into the model's balance between precision (correctly identifying relevant objects) and recall (identifying all relevant objects) [64].
2) **Efficiency**:
   - **Latency**: measures the time taken by the model to perform inference. Lower latency is essential for real-time applications.
   - **Energy Consumption**: KD and FL methods are evaluated on their ability to reduce computational overhead and energy usage.
   - **Model Size**: the reduction in model size is a key metric, with smaller models preferred for devices with limited storage and memory.
3) **Scalability and Robustness**:
   - **Scalability in Federated Learning**: the ability of the system to handle a growing number of clients without significant degradation in performance.
   - **Non-IID Data Performance**: the model's robustness in maintaining accuracy despite data heterogeneity.
4) **Communication Efficiency**:
   - **Update Compression** and **Bandwidth Usage**: the compression ratio of updates sent between clients and the central server, as well as the overall bandwidth consumed during training.
   - **Number of Communication Rounds**: the total number of rounds required for the model to converge.
5) **Real-World Applicability**:
   - **Task-Specific Metrics**: for applications such as autonomous driving or real-time object detection, domain-specific metrics like Intersection over Union (IoU) and Frames Per Second (FPS) are used to evaluate practical performance.
   - **Cross-Device Compatibility**: evaluating how well the model performs across various hardware platforms, such as CPUs, GPUs, and mobile accelerators, ensures broad applicability.

VIII. REAL-WORLD IMPLEMENTATIONS

Federated Learning (FL) has been successfully applied across various domains, including image classification (enhancing user preference predictions based on locally stored images) and language models (improving voice recognition, next-word prediction, and intelligent reply suggestions [28]). The concept of distributed training has also been explored in fundamental models such as the perceptron [31] and deep neural networks (DNNs) for speech recognition [32]. A practical implementation of a Federated Learning algorithm can be found in [33], which introduces the FederatedAveraging Algorithm. This approach selects a C-fraction of clients in each training round and computes the gradient of the loss across all data stored on those clients. A real-world Federated Learning implementation applied to MobileNet is discussed in Section IX.

Similarly, Knowledge Distillation (KD) has demonstrated its effectiveness in deep learning optimization. A notable example is presented in [37], which applies KD to the MNIST dataset [35] and examines its role in ensembling DNN acoustic models for Automatic Speech Recognition (ASR). The study shows that distillation enables a compact student model to achieve comparable or superior performance compared to an ensemble of larger models, significantly improving efficiency. More practical applications of Knowledge Distillation with MobileNet are detailed in Section X.

The increasing adoption of sensor-driven systems across industries calls for efficient data processing frameworks that deliver low latency, high accuracy, and privacy preservation. When optimized with FL and KD, MobileNet becomes a lightweight yet powerful tool for distributed sensor intelligence. In smart city applications, edge devices such as surveillance cameras [86] [87] [88], air quality sensors [89] [90] [91], and traffic monitors [92] [93] [94] generate vast amounts of real-time data. Traditional cloud-based processing poses challenges related to latency, bandwidth consumption, and privacy risks. FL mitigates these issues by enabling edge devices to collaboratively train MobileNet models while keeping data localized, thereby enhancing privacy and reducing communication overhead. KD further optimizes deployment by compressing trained models into lightweight versions suitable for ultra-low-power sensor nodes. For autonomous vehicles, real-time sensor fusion from LIDAR [95], radar, and vision sensors is essential for split-second decision-making. FL facilitates cross-vehicle learning, where models are continuously updated using aggregated insights from multiple vehicles without exposing raw data. Meanwhile, KD ensures that computationally intensive models are distilled into efficient versions, enabling real-time inference on resource-constrained edge processors. This synergy significantly improves object detection, lane recognition, and pedestrian tracking. Beyond transportation, sensor-based healthcare systems—such as wearable monitors and remote patient monitoring devices—greatly benefit from FL and KD. These technologies enable continuous learning from distributed patient data while adhering to strict privacy

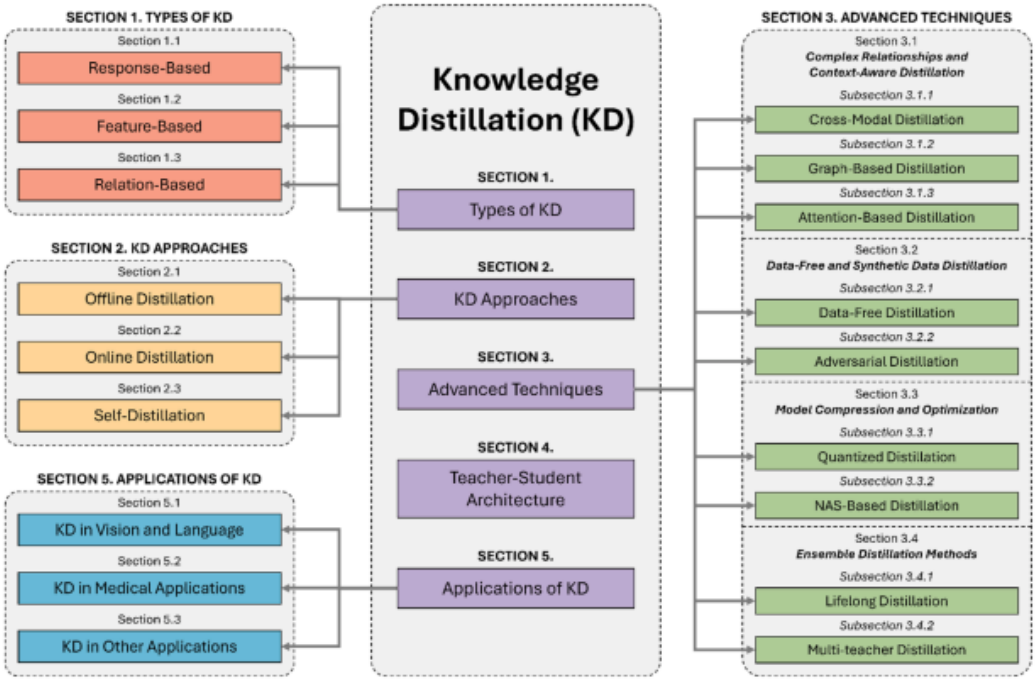


FIGURE 7. The fundamentals of knowledge distillation approaches, advanced techniques, and applications of knowledge distillation. Picture taken from [36].

standards. Their integration enhances real-time diagnostics, anomaly detection, and personalized health monitoring, making remote healthcare more reliable and scalable. By leveraging FL and KD, MobileNet can be effectively deployed in real-world sensor environments, addressing the challenges of latency, computational efficiency, and data privacy across multiple industries.

IX. FEDERATED LEARNING APPLIED TO MOBILENET

This chapter will presents a real-world frameworks to show how it is possible to improve the MobileNet performance using Federated Learning. This framework refers to the paper [38] which introduces *Oort*, a real-world application of Federated Learning (FL) that uses MobileNetV2 and implemented in Python [41]. Oort addresses the critical challenge of selecting an optimal subset of clients to participate in each training round. Most existing approaches tackle this issue by randomly selecting participants [33] [65] [66]. While random selection is straightforward to implement, it often leads to suboptimal performance in federated training due to significant variations in device speed and data characteristics. Moreover, it can result in biased testing datasets and diminished confidence in the outcomes.

In contrast, Oort employs a guided participant selection strategy throughout the FL model's lifecycle. By carefully choosing participants, Oort enhances the time-to-accuracy performance of federated training and allows developers to define specific testing criteria for federated model evaluation. Oort operates as part of the FL framework coordinator, interacting with the execution driver (e.g. PySyft [39] or Tensorflow Federated [40]). Based on the developer-specified criteria, Oort selects and returns a list of participants, while the driver manages the execution of training on these remote participants. The architecture of Oort is designed to optimize these processes (as illustrated in figure 8):

1) Job submission: the developer submits and specifies the participant selection criteria to the FL coordinator in the cloud.
2) Participant selection: the coordinator enquires the clients meeting eligibility properties and forwards their characteristics to Oort which selects participants based on the given criteria and notifies the coordinator of this participant selection.
3) Execution: the coordinator distributes the model to these participants, and then each participant independently computes its own result.
4) Aggregation: when participants complete the computation, the coordinator aggregates updates from participants.

During federated training, where the coordinator initiates the next training round after aggregating updates from enough number of participants, it iterates over 2-4 in each round. Every few training rounds, federated testing is often used to detect whether the cut-off accuracy has been reached.

Oort employs two distinct selectors that developers can access via a client library during FL training and testing:

- Training selector: it captures the utility of clients in training, and efficiently explores and selects high-utility clients at runtime.
- Testing selector: supports two types of selection criteria. When the individual client data characteristics are not provided, the testing selector determines the

number of participants needed to cap the data deviation of participants from the global. Otherwise, it cherry-picks participants to serve the exact developer-specified requirements on data while minimizing the duration of testing.

To ensure privacy and prevent training loss from revealing any raw data, Oort employs three key techniques. First, it uses aggregate training loss, computed locally by the client across all its samples, without exposing the loss distribution of individual samples. Second, when even aggregate loss raises privacy concerns, clients can add noise to their loss values before uploading them. Third, Oort can accommodate alternative definitions of statistical utility as needed.

To strike an optimal balance between selecting clients with high statistical efficiency (i.e. the number of rounds required to achieve the target accuracy) and those with high system efficiency (i.e. the duration of each training round), Oort follows a structured process. It identifies clients with high statistical utility by evaluating their most recent aggregate training loss, adjusted for spatiotemporal variations. It then adaptively allows longer training rounds for clients with higher statistical utility. Additionally, Oort employs an online exploration-exploitation strategy to probabilistically select participants from among high-utility clients, ensuring robustness against outliers. Compared to other state-of-the-art selection techniques, Oort significantly improves time-to-accuracy performance by a factor of 1.2× to 14.1× and enhances final model accuracy by 1.3% to 9.8% in federated model training, while achieving performance close to the theoretical upper bound of statistical efficiency.

Specifically to reconcile the demand for both efficiencies, it maximize the statistical utility per unit time. As such, it formulates the utility of client i by associating its statistical utility with a global system utility in terms of the duration of each training round:

$$\text{Util}(i) = \underbrace{|B_i|\sqrt{\frac{1}{|B_i|}\sum_{k\in B_i}\text{Loss}(k)^2}}_{\text{Statistical utility } U(i)} \times \underbrace{\left(\frac{T}{t_i}\right)^{1_{\{T<t_i\}}\times\alpha}}_{\text{Global sys utility}}$$

where T is the developer-preferred duration of each round, t_i is the amount of time that client i takes to process the training, which has already been collected by coordinator from past rounds and $1(x)$ is an indicator function that takes value 1 if x is true and 0 otherwise. This way, the utility of those clients who may be the bottleneck of the desired speed of current round will be penalized by a developer-specified factor α. Based on the definition of client utility, in order to select participants with the highest utility in each training round it is needed to address the following practical concerns:

- Scalability: a client's utility can only be determined after it has participated in training, making it necessary to decide how to choose from clients at scale without having to try all clients once.
- Staleness: since not every client participates in every round, it is important to account for the change in a client's utility since its last participation.
- Robustness: ensuring robustness to outliers is crucial in the presence of corrupted clients (e.g. with noisy data).

At the beginning of each selection round, Oort receives the feedback of the last training round, and updates the statistical utility and system performance of clients. Meanwhile, it samples $\epsilon \in [0, 1]$ fraction of participants to explore potential participants that had not been selected before. Oort employs two strategies to account for the dynamics in client utility over time. First it gradually increases the utility of a client if it has been overlooked for a long time. So those clients accumulating high utility since their last trial can still be repurposed again. Second, it allows a confidence interval c on the cut-off utility by admitting clients whose utility is greater than the c% of the top $(1-\epsilon)\times K$-th participant. To account also the robustness Oort removes the client in selection after it has been picked over a given number of rounds and clips the utility value of a client by capping it to no more than an upper bound.

Oort has been evaluated on tasks such as speech recognition, image classification, and language modeling. Specifically, for the image classification task, MobileNetV2 was trained using the OpenImage dataset [42], with time-to-accuracy performance and final model accuracy serving as the primary metrics. The results demonstrate that Oort achieves the target accuracy 3.3× to 14.1× faster in terms of wall-clock time on the mid-sized OpenImage dataset, while requiring 1.8× to 4.8× fewer training rounds.

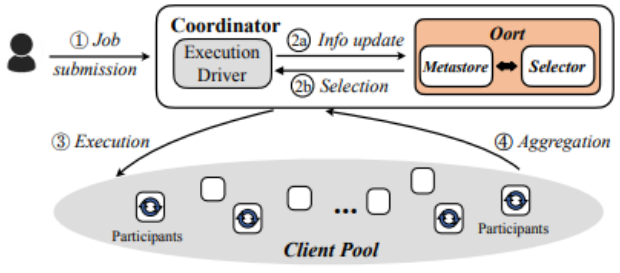


FIGURE 8. Oort architecture overview. Picture taken from [38].

X. KNOWLEDGE DISTILLATION APPLIED TO MOBILENET

This chapter will presents three real-world frameworks to show how it is possible to improve the MobileNet performance using Knowledge Distillation.

A. FIRST CASE

The first one, presented in the paper [43] introduces a Knowledge Distillation approach specifically designed for MobileNet. In this framework (illustrated in figure 9) the student model is trained to align with the soft labels predicted by the teacher model. This approach is particularly effective because images often contain multiple objects, making

manually labeled one-hot hard labels insufficient to fully capture this complexity. By leveraging soft labels, the need for manual annotations is eliminated, allowing unlabeled data to be directly utilized to maximize the knowledge transferred from the teacher model. The proposed method tries to transfer knowledge learned by a teacher model Q to a student model P, with the training dataset T and a carefully collected unlabeled dataset U, which is a subset of a much larger unlabeled image gallery X_U. Both the training set T and the unlabeled set U are used for knowledge distillation. The teacher model is also leveraged to perform unlabeled data filtering: the filtering step is designed to make sure that valuable unlabeled data is selected to help the distillation process and the selected data is quite different from the validation dataset V. The framework only force that predictions of the student model and teacher model are well aligned. Such a constraint is achieved by minimizing the Jensen–Shannon divergence [67].

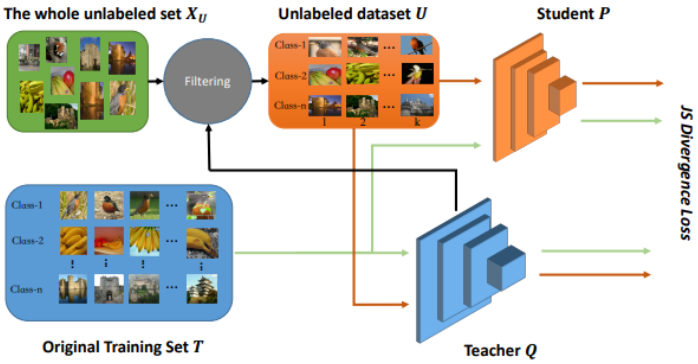


FIGURE 9. Illustration of the overall framework. Picture taken from [43].

In the framework the teacher model is ResNet50-D [44] while the student is one between MobileNetV1, MobileNetV2, MobileNetV3-Small and MobileNetV3-Large which are trained with only soft-labels.
The training and validation set are denoted as $T = \{(x_i, y_{xi})\}$ and $V = \{(x_i, y_{xi})\}$ respectively, where x_i means a input image and y_{xi} is the corresponding label. Besides there is another unlabeled $U = \{x_i\}$. For the labeled dataset, the complete ImageNet-1K [26] training data, consisting of 1.2 million images, is utilized. For the unlabeled dataset, 4 million images are selected from ImageNet-22K [51]. To ensure fairness in evaluation, images from ImageNet-22K that are visually similar to the validation images of ImageNet-1K are filtered out. This filtering process uses SIFT [45] to match similar image pairs between the two datasets. The purpose of this step is to prevent the student model from being trained on images resembling the validation set, thereby maintaining the integrity of the evaluation results. Next, the teacher model generates prediction results for the remaining images in ImageNet-22K. The images are then ranked within each category based on their scores, and the top k images from each category are selected to form the final unlabeled dataset U. By default k is set to 4000, so the unlabeled data has a total of 4M images and in total 5.2M images are used for distillation. Table 5 shows the experimental results achieved by applying this framework to the four MobileNet architectures. It is clear that this solution can consistently increase the model accuracy by a very large margin.

Model	Top-1 Acc	New Top-1 Acc
MobileNetV1	70.9%	77.9% (+7.0)
MobileNetV2	72.1%	76.4% (+4.3)
MobileNetV3-small	68.2%	71.3% (+3.1)
MobileNetV3-large	75.3%	79.0% (+3.7)

Table 5. Accuracy improvement after applying the Knowledge Distillation method. Table taken from [43].

B. SECOND CASE

The second example of Knowledge Distillation applied to MobileNet refers to the paper [46]. In this case the distillation is interpreted as a task of matching the functions implemented by the teacher and student. This idea it is based on two principles of Knowledge Distillation:

1) Teacher and student should process the exact same crop and augmentations.
2) The functions should match on a large number of support points to generalize well.

Based on this two observations, consistent image views, aggressive augmentations and very long training schedules are the key to make model compression via knowledge distillation work well in practice. The framework adopts the model BiT-ResNet [68] pretrained on ImageNet-21k [51] as teacher (referred as T) and MobileNetV3-Large [9] as student (referred as S). It uses as distillation loss the KL-divergence [69] between the teacher's p_t, and the student's p_s predicted class probability vectors.
This distillation framework is based on the "consistent and patient teacher" hypothesis which argues that distillation performs better when seen as function matching, i.e. when the student and teacher see consistent views of the input images, and when student is trained using long training schedule (i.e. the "teacher" is patient). To demonstrate first that the consistency criterion is valid it has been defined multiple distillation configurations:

- **Fixed Teacher**: in this approach, the teacher's predictions remain constant for a given image, using precomputed targets. Several variations are considered:
  - **Fix/rs**: this is the simplest configuration, where both the teacher and the student receive an image resized to 224 pixels. This method serves as a baseline but performs poorly.
  - **Fix/cc**: a more common setup, where the teacher receives a fixed central crop of the image, while the student is trained on mildly randomized crops.

  - **Fix/ic**: a data augmentation-heavy configuration where the teacher's predictions are computed as the average output over 1,000 inception crops. This strategy has been validated to improve the teacher's performance. The student is trained with random inception crops.
- **Independent Noise**: this strategy applies independent augmentations to the teacher and student inputs. Two variations are considered:
  - **Ind/rc**: both the teacher and the student receive independently generated mild random crops of the same image.
  - **Ind/ic**: a more augmentation-intensive setup where both teacher and student inputs are generated through independent inception crops.
- **Consistent Teaching**: in this configuration, the image is randomly cropped only once, and the same crop is used as input for both the teacher and the student. Two variations are considered:
  - **Same/rc**: a mild random crop is applied.
  - **Same/ic**: a heavy inception crop is used.

  This approach ensures consistency between the teacher and student inputs, providing a controlled setting for distillation.
- **Function Matching**: An extension of consistent teaching, this method expands the input manifold using mixup. Both the teacher and the student receive the same augmented input to ensure alignment in their learning process.

If distillation is interpreted as a function matching, and, crucially, make sure to provide consistent inputs to the student and teacher it is possible to be very aggressive with image augmentations: even if an image view is too distorted, it will still make a progress towards matching the relevant functions on this input. Thus, augmentations can be more opportunistic and avoid overfitting by doing aggressive image augmentations and, if true, optimize for very long time until the student's function comes close to the teacher's. The figure 10 shows the power of applying the "consistent and patient teacher" hypothesis.

The framework has been tested for a broad range of practical settings to ensure its robustness and evaluated on the classification accuracy. In particular it was experimented on five popular image classification datasets: flowers102 [47], pets [48], food101 [49], sun397 [50] and ImageNet [5]. By using the latter dataset, the distillation method allows MobileNetV3-Large to reach 74.60% after 300 epochs, and 76.31% after 1200 epochs.

C. THIRD CASE

The last example of Knowledge Distillation applied to MobileNet is the one reported in the paper [12]. It is based on the previous example [46] and addresses the TuNAS component. The success of TuNAS hinges on accurately evaluating architecture quality but it is notably affected by data augmentation, regularization, and hyper-parameter choices. To address this problem it has been applied the JFT dataset [70], an offline distillation dataset, eliminating the need for extra augmentations and reducing sensitivity to regularization and optimization settings with notable improvements shown in figure 11.

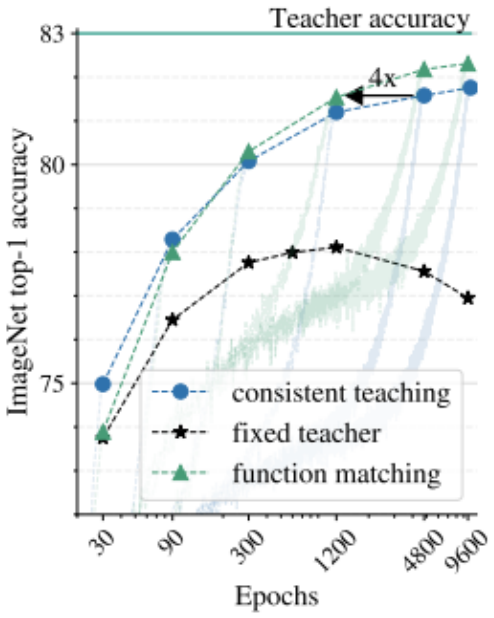


FIGURE 10. Distillation works the best when it is patiently trained for a large number of epochs and the image views of the teacher and student models are consistent (green and blue lines) Figure taken from [46].

To further improve performance it introduces two novel techniques:

1) **Dynamic Dataset Mixing**: Data augmentation is crucial for distillation performance. Instead of relying on a fixed augmentation sequence, dynamically mixing multiple datasets with diverse augmentation strategies improves distillation. The experiments use three distillation datasets:
   - D_1: Inception Crop [52] followed by RandAugment [53] applied to 500 ImageNet-1k replicas.
   - D_2: Inception Crop followed by extreme Mixup [54] applied to 1000 ImageNet1k replicas (mirroring the Patient Teacher approach).
   - $D_1 + D_2$: a dynamic mixture of D_1 and D_2 during training.

   Results of the experiment show that D_2 outperforms D_1 (84.1% vs. 83.8% student accuracy), but a dynamic mixture of the two ($D_1 + D_2$) elevates accuracy to 84.4%.
2) **JFT Data Augmentation**: to increase training data volume adds indomain class-balanced data by resampling the JFT-300M dataset [55] to 130K images per class (130M total). Selects images with a relevance threshold above 0.3. For classes with abundant data, chooses the top 130K images; for rare classes, it repli-

cates images for balance. This dataset is replicated 10x and represents the dataset D_3. Experiment results shows that using solely D_3 drops accuracy by 2%. However, combining ImageNet and JFT data ($D_1 + D_2 + D_3$) raises accuracy by +0.6

The combined distillation recipe dynamically mixes datasets $D1$, $D2$, and $D3$ for diverse augmentations and leverages class-balanced JFT data. The method improves top-1 accuracy $> 0.8\%$ while the student have 15x fewer parameters and 48x fewer MACs than its teacher (EfficientNet-L2 [56]), but is only 1.6% less accurate.

NAS Dataset	Top-1 Acc (Val/Train)	MACs	Params	Pixel 4 GPU	Pixel 6 CPU
ImageNet	82.4 / 72.9	7.2G	43.5M	59.2ms	70.4ms
JFT distill	82.3 / 74.0	6.2G	34.4M	51.0ms	67.3ms
Gain	-0.1 / +1.1	**+13.9%**	**+20.9%**	**+13.9%**	**+4.4%**

FIGURE 11. Performance Boost from JFT Distillation. Figure taken from [12].

To create the offline distillation dataset, it first augments a candidate image and then compress it using JPEG encoding. It then decode the image and run inference using the teacher. The teacher's predicted probability (the softmax output) of the 1k classes is recorded and will be used later as soft labels to train the student. During student training, all data augmentation are disabled except left-righ flip. Dropout is also applied to the final fully connected layer as the only regularization method. Cross-entropy is used between the teacher's soft labels and the student's predicted class probability vectors as our training loss. The student is trained with the AdamW optimizer [71]. Table 6 shows the accuracy improvement.

Model	IN-1k Only	SOTA Distill	New Distill	Gain
MNv4-Conv-S	73.8	-	75.5	+1.7
MNv4-Conv-M	79.9	81.5	82.7	+2.8
MNv4-Hybrid-M	80.7	82.7	83.7	+3.0
MNv4-Conv-L	82.9	84.4	85.9	+3.0
MNv4-Hybrid-L	83.4	85.7	86.6	+3.2

Table 6. Top-1 Accuracy Comparison Across Training Approaches. Table taken from 11.

XI. COMPARISON BETWEEN THE KNOWLEDGE DISTILLATION CASES

The application of Knowledge Distillation (KD) to MobileNet has demonstrated significant potential in improving the performance and efficiency of lightweight models tailored for edge devices. This section compares the approaches described in the analyzed cases, highlighting their strategies, benefits, and limitations.

The first KD framework focuses on leveraging unlabeled datasets to enhance the knowledge transfer process. By utilizing soft labels generated by a teacher model, it maximizes the utility of unlabeled data. This approach effectively reduces dependency on manually labeled datasets and boosts student model accuracy. Notably, applying this method to various MobileNet architectures (V1, V2, V3-Small, and V3-Large) resulted in substantial accuracy gains, as high as +7% in some cases. The success is attributed to filtering and ranking the unlabeled data to ensure alignment with the teacher model's predictions, which optimizes the quality of transferred knowledge.

In contrast, the second case emphasizes a "consistent and patient teacher" hypothesis, where both the teacher and student process identical, augmented image views. This consistency, combined with long training schedules, improves function matching between teacher and student models. This strategy demonstrated a significant performance boost, particularly when aggressive augmentations were applied, allowing the student to better generalize complex patterns learned by the teacher. Experiments with MobileNetV3-Large showcased the effectiveness of consistent views and prolonged training, with accuracies improving notably across various datasets.

The third approach extends the consistent teacher-student paradigm by incorporating dynamic dataset mixing and large-scale pretraining on augmented datasets. By dynamically combining diverse augmentation strategies and leveraging class-balanced datasets (e.g., JFT), this method further enhances student model performance. The introduction of advanced data augmentation and distillation recipes tailored to the MobileNetV4 architecture resulted in over 3% accuracy improvements while maintaining low computational overhead. A common thread across these cases is the reliance on robust teacher models, such as ResNet variants, to guide lightweight MobileNet students. However, each approach uniquely balances trade-offs between computational efficiency and accuracy gains. The first framework excels in unlabeled data utilization, while the second and third cases prioritize function consistency and augmentation diversity, respectively.

Overall, the advancements in KD tailored to MobileNet architectures highlight the versatility of distillation methods. Each approach addresses distinct challenges, from leveraging unlabeled data to optimizing augmentations and long-term training. These methods pave the way for deploying highly efficient and accurate models in resource-constrained environments. Further research may explore the hybridization of these techniques to harness their combined strengths for future MobileNet variants.

XII. CHALLENGES AND FUTURE DIRECTIONS

Federated Learning (FL) faces significant communication challenges, particularly when applied to resource-constrained environments involving MobileNet. Communication overhead in FL arises due to the frequent exchange of model updates between distributed devices and the central server. Optimizing communication efficiency is critical for enabling FL on mobile and edge devices where bandwidth is often limited. Potential strategies include:

- Model compression techniques: reducing the size of updates through quantization, pruning, and efficient encoding can significantly lower communication costs.

- Asynchronous communication: implementing asynchronous update mechanisms can reduce waiting times during aggregation and allow devices to participate flexibly without requiring synchronized communication.
- Federated Averaging optimization: enhancing algorithms such as Federated Averaging by integrating adaptive learning rates and selecting high-utility clients can further reduce the number of communication rounds needed for convergence.

Research into lightweight protocols tailored for MobileNet architectures will also play a vital role. These protocols should consider the unique design of MobileNet, focusing on its compact and efficient structure to maximize compatibility with FL. The decentralized nature of FL poses challenges related to data security and privacy. While FL inherently avoids transmitting raw data, vulnerabilities still exist in the form of adversarial attacks, data leakage through gradients, and poisoning attacks. Addressing these issues in the context of MobileNet requires targeted efforts:

- Secure aggregation protocols: techniques such as homomorphic encryption and differential privacy must be further optimized for MobileNet to ensure secure and efficient training.
- Robustness to adversarial attacks: MobileNet-specific adversarial defenses should be developed to mitigate the risk of poisoned or compromised updates that could affect model accuracy.
- Decentralized privacy mechanisms: techniques like federated dropout and local differential privacy can offer additional layers of security without overburdening computational resources on mobile devices.

Collaborative frameworks that incorporate privacy-preserving machine learning techniques while leveraging MobileNet's lightweight nature will drive future advancements in this area.

Scaling Knowledge Distillation (KD) to handle a broad range of MobileNet variants presents both opportunities and challenges. As newer iterations of MobileNet (e.g., MobileNetV4) introduce more sophisticated architectural components, KD methods must adapt to maintain efficiency and effectiveness:

- Adapting to architecture-specific features: tailoring KD techniques to leverage unique features, such as inverted residuals and linear bottlenecks, in MobileNet variants.
- Efficient use of teacher models: ensuring that the teacher model provides meaningful and actionable guidance to student models across diverse tasks and architectures.
- Balancing accuracy and compression: striking the right trade-off between accuracy improvements and model compression, particularly for edge-device deployments.

Future research should focus on automated KD pipelines capable of dynamically adapting to different MobileNet architectures and tasks. MobileNet's applicability extends beyond simple classification tasks to more complex domains, such as object detection, semantic segmentation, and real-time video analysis. Scaling KD for these tasks introduces several challenges:

- Multi-Task learning: KD frameworks must be extended to transfer knowledge for multi-task scenarios, enabling MobileNet to simultaneously handle diverse objectives.
- Knowledge Representation for Complex Tasks: beyond response-based KD, feature- and relation-based KD methods must be enhanced to encode and transfer richer knowledge representations suitable for complex applications.
- Domain adaptation: KD techniques must evolve to support domain-specific optimizations, ensuring MobileNet can adapt to unique datasets and real-world conditions without retraining.

Innovations in KD that address these challenges will unlock new possibilities for MobileNet, enabling it to excel in highly demanding and diverse applications while maintaining its lightweight and efficient design.

XIII. CONCLUSION

This study provides a comprehensive exploration of the synergistic integration of Knowledge Distillation (KD) and Federated Learning (FL) with the MobileNet architecture, demonstrating their transformative potential for edge-device applications. The findings highlight that KD significantly enhances the accuracy and efficiency of MobileNet models by compressing large, complex teacher models into smaller, resource-efficient student models without compromising performance. Innovative KD techniques, such as response-based, feature-based, and relation-based approaches, enable substantial accuracy improvements while maintaining computational efficiency. Similarly, Federated Learning has proven effective in enhancing MobileNet's privacy-preserving capabilities by enabling decentralized model training directly on edge devices, reducing reliance on centralized data aggregation, and addressing real-world challenges like non-IID data distributions and communication inefficiencies. The integration of these approaches has demonstrated remarkable performance improvements in real-world applications, such as autonomous systems, IoT devices, and mobile vision tasks, enabling robust scalability and adaptability across diverse hardware platforms.

Despite these advancements, challenges remain. Key hurdles include optimizing communication efficiency in FL, scaling KD to handle increasingly complex MobileNet architectures, and addressing privacy and security concerns in federated environments. Future research should focus on automated KD pipelines tailored to specific MobileNet variants and domain-specific optimizations to ensure sustained advancements. By addressing these challenges and leveraging the synergies between KD, FL, and MobileNet, this integration holds significant promise for driving innovations in efficient, privacy-preserving machine learning for real-world applications.

REFERENCES

[1] eCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[2] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, abs/1704.04861, 2017.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. Int. J. Comput. Vision, 115(3):211–252, December 2015.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[7] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. mobile networks for classification, detection and segmentation. CoRR, abs/1801.04381, 2018.

[8] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. CoRR, abs/1707.01083, 2017.

[9] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. CoRR, abs/1905.02244, 2019.

[10] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. CoRR, abs/1807.11626, 2018.

[11] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. CoRR, abs/1812.00332, 2018.

[12] Qin, Danfeng, et al. "MobileNetV4: Universal Models for the Mobile Ecosystem." European Conference on Computer Vision. Springer, Cham, 2025.

[13] Tang, Charles. "Monocular Cyclist Detection with Convolutional Neural Networks." arXiv preprint arXiv:2303.11223 (2023).

[14] Huang, Shan, Ye He, and Xiao-an Chen. "M-YOLO: a nighttime vehicle detection method combining mobilenet v2 and YOLO v3." Journal of Physics: Conference Series. Vol. 1883. No. 1. IOP Publishing, 2021.

[15] Kalgaonkar, Priyank, and Mohamed El-Sharkawy. "An Improved Lightweight Network Using Attentive Feature Aggregation for Object Detection in Autonomous Driving." Journal of Low Power Electronics and Applications 13.3 (2023): 49.

[16] Deng, Tianmin, and Yongjun Wu. "Simultaneous vehicle and lane detection via MobileNetV3 in car following scene." Plos one 17.3 (2022): e0264551.

[17] Chen L C, Papandreou G, Kokkinos I, et al. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2018; 40(4), 834–848. https://doi.org/10.1109/TPAMI.2017.2699184 PMID: 28463186

[18] Redmon J, Farhadi A. (2018) YOLOv3: An Incremental Improvement. arXiv e-prints.

[19] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and Efficient Object Detection. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10778–10787, June 2020. ISSN: 2575-7075.

[20] Berkin Akin, Suyog Gupta, Yun Long, Anton Spiridonov, Zhuo Wang, Marie White, Hao Xu, Ping Zhou, and Yanqi Zhou. Searching for efficient neural architectures for on-device ML on edge tpus. In IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022, New Orleans, LA, USA, June 19-20, 2022, pages 2666–2675. IEEE, 2022.

[21] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pages 11966–11976. IEEE, 2022.

[22] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[23] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V. Le. Can weight sharing outperform random architecture search? an investigation with tunas. In The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[24] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning, pages 6105–6114. PMLR, 2019.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[26] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255, June 2009.

[27] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V, volume 8693 of Lecture Notes in Computer Science, pages 740–755. Springer, 2014.

[28] Greg Corrado. Computer, respond to this email. http://googleresearch.blogspot.com/2015/11/computer-respond-to-this-email.html, November 2015.

[29] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in Federated Learning. In Foundations and Trends in Machine Learning, 2021.

[30] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B. Gibbons. The Non-IID data quagmire of decentralized machine learning. In ICML, 2020.

[31] Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In NAACL HLT, 2010.

[32] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. Parallel training of deep neural networks with natural gradient and parameter averaging. In ICLR Workshop Track, 2015.

[33] McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." Artificial intelligence and statistics. PMLR, 2017.

[34] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[35] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[36] Amir Moslemi, Anna Briskina, Zubeka Dang, Jason Li. A survey on knowledge distillation: Recent advancements, Machine Learning with Applications, Volume 18, 2024, 100605, ISSN 2666-8270. https://doi.org/10.1016/j.mlwa.2024.100605.

[37] Hinton, Geoffrey. "Distilling the Knowledge in a Neural Network." arXiv preprint arXiv:1503.02531 (2015).

[38] Lai, Fan, et al. "Oort: Efficient federated learning via guided participant selection." 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21). 2021.

[39] PySyft. https://github.com/OpenMined/PySyft.

[40] TensorFlow Federated. https://www.tensorflow.org/federated.

[41] Oort. https://github.com/SymbioticLab/Oort.

[42] Google Open Images Dataset. https://storage.googleapis.com/openimages/web/index.html

[43] Cui, Cheng, et al. "Beyond Self-Supervision: A Simple Yet Effective Network Distillation Alternative to Improve Backbones." arXiv preprint arXiv:2103.05959 (2021).

[44] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 558–567, 2019.

[45] David G Lowe. Object recognition from local scale-invariant features. In Proceedings of the seventh IEEE international conference on computer vision, volume 2, pages 1150–1157. Ieee, 1999.

[46] Beyer, Lucas, et al. "Knowledge distillation: A good teacher is patient and consistent." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.

[47] Maria-Elena Nilsback and Andrew Zisserman. Delving deeper into the whorl of flower segmentation. In British Machine Vision Conference (BMVC), 2007.

[48] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[49] Parneet Kaur, Karan Sikka, and Ajay Divakaran. Combining weakly and webly supervised learning for classifying food images. arXiv preprint arXiv:1712.08730, 2017.

[50] and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In Conference on Computer Vision and Pattern Recognition (CVPR), 2010.

[51] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision (IJCV), 2015.

[52] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818–2826, 2016.

[53] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pages 702–703, 2020.

[54] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412, 2017.

[55] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In Proceedings of the IEEE international conference on computer vision, pages 843–852, 2017.

[56] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10687–10698, 2020.

[57] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11), 2278–2324.

[58] "Deep Sparse Rectifier Neural Networks" By Xavier Glorot, Antoine Bordes, and Yoshua Bengio, 2011. Published in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS).

[59] "A Theory of the Learnable" By Leslie G. Valiant, 1984. Communications of the ACM, 27(11), 1134–1142. DOI:10.1145/1968.1972.

[60] "Deep Residual Learning for Image Recognition" By Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 2015. Published in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[61] "Learning Transferable Architectures for Scalable Image Recognition". By Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le, 2018. Published in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[62] "Feature Pyramid Networks for Object Detection" By Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, 2017. Published in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[63] Zhang, Jing, et al. "A survey on class imbalance in federated learning." arXiv preprint arXiv:2303.11673 (2023).

[64] An Introduction to Information Retrieval" (2008) By: Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze.

[65] Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2018). Federated Optimization in Heterogeneous Networks. Proceedings of Machine Learning and Systems (MLSys).

[66] Li, Q., Wen, Z., and He, B. (2019). Federated Learning: Challenges, Methods, and Future Directions. IEEE Signal Processing Magazine, 37(3), 50–60.

[67] Lin, Jianhua. "Divergence measures based on the Shannon entropy." IEEE Transactions on Information theory 37.1 (1991): 145-151.

[68] "Big Transfer (BiT): General Visual Representation Learning" (2019). Authors: Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, Neil Houlsby. Published in: Proceedings of the European Conference on Computer Vision (ECCV) 2020.

[69] Kullback, Solomon, and Richard A. Leibler. "On information and sufficiency." The annals of mathematical statistics 22.1 (1951): 79-86.

[70] "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era" by Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta, 2017.

[71] Loshchilov, I. "Decoupled weight decay regularization." arXiv preprint arXiv:1711.05101 (2017).

[72] J. R. K, C. Nigam, G. Kirubasri, S. Jayachitra, A. Aeron and D. Suganthi, "Real-Time Object Detection on Edge Devices Using Mobile Neural Networks," 2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), Bangalore, India, 2024, pp. 1-4, doi: 10.1109/IITCEE59897.2024.10467220.

[73] Singh, Tanpreet, Optimizing Neural Networks for Real-Time Object Detection Using Edge Computing and AI (October 01, 2024). Available at SSRN: https://ssrn.com/abstract=5018415 or http://dx.doi.org/10.2139/ssrn.5018415.

[74] Choi, Min-Kook, and Heechul Jung. "Development of fast refinement detectors on ai edge platforms." Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part IV. Springer International Publishing, 2021.

[75] Bechinia, Hadjer, Djamel Benmerzoug, and Nawres Khlifa. "Approach Based Lightweight Custom Convolutional Neural Network and Fine-Tuned MobileNet-V2 for ECG Arrhythmia Signals Classification." IEEE Access 12 (2024): 40827-40841.

[76] Rani, Manjot, and Munish Kumar. "MobileNet for human activity recognition in smart surveillance using transfer learning." Neural Computing and Applications (2024): 1-18.

[77] Dang, Thai-Viet. "Smart home management system with face recognition based on ArcFace model in deep convolutional neural network." Journal of Robotics and Control (JRC) 3.6 (2022): 754-761.

[78] Li, Y.; Huang, H.; Xie, Q.; Yao, L.; Chen, Q. Research on a Surface Defect Detection Algorithm Based on MobileNet-SSD. Appl. Sci. 2018, 8, 1678. https://doi.org/10.3390/app8091678.

[79] Aman Kumar. Deep Learning-based Defect Detection in Telecom Towers. July 21, 2022.

[80] Bolla, Bharath Kumar, Mohan Kingam, and Sabeesh Ethiraj. "Efficient deep learning methods for identification of defective casting products." International Conference on Cognition and Recongition. Cham: Springer Nature Switzerland, 2021.

[81] Rajagopal, Manoj Kumar, and Bala Murugan MS. "Artificial Intelligence based drone for early disease detection and precision pesticide management in cashew farming." arXiv preprint arXiv:2303.08556 (2023).

[82] Geethanjali, P., and M. Rajeshwari. "Advances in Ecological Surveillance: Real-Time Wildlife Detection using MobileNet-SSD V2 CNN." (2023).

[83] Dominguez-Morales, Juan P., et al. "Wildlife monitoring on the edge: A performance evaluation of embedded neural networks on microcontrollers for animal behavior classification." Sensors 21.9 (2021): 2975.

[84] Zualkernan, Imran, et al. "An IoT system using deep learning to classify camera trap images on the edge." Computers 11.1 (2022): 13.

[85] Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions." arXiv preprint arXiv:1710.05941 (2017).

[86] J. Zhang, J. Zhou, J. Guo and X. Sun, "Visual Object Detection for Privacy-Preserving Federated Learning," in IEEE Access, vol. 11, pp. 33324-33335, 2023, doi:10.1109/ACCESS.2023.3263533.

[87] Shenaj, Donald, Giulia Rizzoli, and Pietro Zanuttigh. "Federated learning in computer vision." IEEE Access (2023).

[88] Honmote, Harshal, et al. "Real time object detection and recognition using MobileNet-SSD with OpenCV." Int J Eng Res Technol 11.1 (2022).

[89] Y. Liu, J. Nie, X. Li, S. H. Ahmed, W. Y. B. Lim and C. Miao, "Federated Learning in the Sky: Aerial-Ground Air Quality Sensing Framework With UAV Swarms," in IEEE Internet of Things Journal, vol. 8, no. 12, pp. 9827-9837, 15 June15, 2021, doi: 10.1109/JIOT.2020.3021006.

[90] Liu, Shiya, and Yang Yi. "Knowledge Distillation between DNN and SNN for Intelligent Sensing Systems on Loihi Chip." 2023 24th International Symposium on Quality Electronic Design (ISQED). IEEE, 2023.

[91] Abimannan, S.; El-Alfy, E.-S.M.; Hussain, S.; Chang, Y.-S.; Shukla, S.; Satheesh, D.; Breslin, J.G. Towards Federated Learning and Multi-Access Edge Computing for Air Quality Monitoring: Literature Review and Assessment. Sustainability 2023, 15, 13951. https://doi.org/10.3390/su151813951.

[92] Han, Peng, et al. "FedDAN: Federated Learning with Distillation and Adapter Net Model Using in Intrusion Detection System." Proceedings of the 2024 9th International Conference on Cyber Security and Information Engineering. 2024.

[93] P. Dell'Acqua, M. Fazio, L. Carnevale and M. Villari, "Knowledge Distillation and Federated Learning for Data-Driven Monitoring of Electrical Vehicle Li-Battery," 2024 IEEE Symposium on Computers and Communications (ISCC), Paris, France, 2024, pp. 1-5, doi: 10.1109/ISCC61673.2024.10733629.

[94] Wu, C., Wu, F., Lyu, L. et al. Communication-efficient federated learning via knowledge distillation. Nat Commun 13, 2032 (2022). https://doi.org/10.1038/s41467-022-29763-x.

[95] Lidar, Velodyne. "What is lidar." Learn How Lidar Works," Velodyne Lidar,[online] (2022).

FIRST A. AUTHOR (M'76–SM'81–F'87) and all authors may include biographies. Biographies are often not included in conference-related papers. This author became a Member (M) of IEEE in 1976, a Senior Member (SM) in 1981, and a Fellow (F) in 1987. The first paragraph may contain a place and/or date of birth (list place, then date). Next, the author's educational background is listed. The degrees should be listed with type of degree in what field, which institution, city, state, and country, and year the degree was earned. The author's major field of study should be lower-cased.

The second paragraph uses the pronoun of the person (he or she) and not the author's last name. It lists military and work experience, including summer and fellowship jobs. Job titles are capitalized. The current job must have a location; previous positions may be listed without one. Information concerning previous publications may be included. Try not to list more than three books or published articles. The format for listing publishers of a book within the biography is: title of book (publisher name, year) similar to a reference. Current and previous research interests end the paragraph. The third paragraph begins with the author's title and last name (e.g., Dr. Smith, Prof. Jones, Mr. Kajor, Ms. Hunter). List any memberships in professional societies other than the IEEE. Finally, list any awards and work for IEEE committees and publications. If a photograph is provided, it should be of good quality, and professional-looking. Following are two examples of an author's biography.

SECOND B. AUTHOR was born in Greenwich Village, New York, NY, USA in 1977. He received the B.S. and M.S. degrees in aerospace engineering from the University of Virginia, Charlottesville, in 2001 and the Ph.D. degree in mechanical engineering from Drexel University, Philadelphia, PA, in 2008.

From 2001 to 2004, he was a Research Assistant with the Princeton Plasma Physics Laboratory. Since 2009, he has been an Assistant Professor with the Mechanical Engineering Department, Texas A&M University, College Station. He is the author of three books, more than 150 articles, and more than 70 inventions. His research interests include high-pressure and high-density nonthermal plasma discharge processes and applications, microscale plasma discharges, discharges in liquids, spectroscopic diagnostics, plasma propulsion, and innovation plasma applications. He is an Associate Editor of the journal Earth, Moon, Planets, and holds two patents.

Dr. Author was a recipient of the International Association of Geomagnetism and Aeronomy Young Scientist Award for Excellence in 2008, and the IEEE Electromagnetic Compatibility Society Best Symposium Paper Award in 2011.

THIRD C. AUTHOR, JR. (M'87) received the B.S. degree in mechanical engineering from National Chung Cheng University, Chiayi, Taiwan, in 2004 and the M.S. degree in mechanical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2006. He is currently pursuing the Ph.D. degree in mechanical engineering at Texas A&M University, College Station, TX, USA.

From 2008 to 2009, he was a Research Assistant with the Institute of Physics, Academia Sinica, Tapei, Taiwan. His research interest includes the development of surface processing and biological/medical treatment techniques using nonthermal atmospheric pressure plasmas, fundamental study of plasma sources, and fabrication of micro- or nanostructured surfaces.

Mr. Author's awards and honors include the Frew Fellowship (Australian Academy of Science), the I. I. Rabi Prize (APS), the European Frequency and Time Forum Award, the Carl Zeiss Research Award, the William F. Meggers Award and the Adolph Lomb Medal (OSA).