Efficient Deep Learning: A Survey of Model Compression and Optimization Techniques for Resource-Constrained Environments

Preprint · August 2025

DOI: 10.13140/RG.2.2.24876.58240

CITATIONS
0

READS
82

1 author:

Meade Cleti
Arizona State University
10 PUBLICATIONS **3** CITATIONS

SEE PROFILE

Efficient Deep Learning: A Survey of Model Compression and Optimization Techniques for Resource-Constrained Environments

Meade Cleti
Arizona State University
mcleti@asu.edu

***Abstract*—The exponential growth in deep learning model sizes has created unprecedented challenges for deployment in resource-constrained environments. This comprehensive survey examines the landscape of efficient deep learning, focusing on model compression techniques, optimization strategies, and hardware acceleration methods. We present a unified taxonomy of compression approaches including pruning, quantization, knowledge distillation, and neural architecture search, analyzing their trade-offs between model size, computational efficiency, and accuracy. Our analysis reveals that hybrid approaches combining multiple compression techniques achieve compression ratios exceeding 100× while maintaining within 2% of original accuracy for many applications. We explore optimization strategies ranging from efficient training algorithms to runtime optimizations, demonstrating how techniques like mixed-precision training and gradient checkpointing reduce memory footprint by up to 75%. Furthermore, we examine hardware-specific optimizations for GPUs, TPUs, and edge devices, highlighting the co-design of algorithms and hardware architectures. Through systematic evaluation of over 150 research contributions, we identify key patterns in the evolution of efficient deep learning and provide practical guidelines for practitioners. Our findings indicate that the field is rapidly converging toward automated, hardware-aware compression pipelines that adapt to specific deployment constraints. We conclude by outlining emerging challenges and future research directions, including dynamic neural networks, neuromorphic computing, and the integration of efficiency considerations into the model design phase rather than as post-hoc optimizations.**

I. Introduction

The remarkable success of deep learning across diverse domains has been accompanied by an exponential increase in model complexity and computational requirements. Deep learning has revolutionized fields from computer vision [1], [2] to natural language processing [3]–[5]. State-of-the-art models like GPT-4 [6] and PaLM 2 [7] contain hundreds of billions of parameters, requiring substantial computational resources for both training and inference. This trend poses significant challenges for deploying deep learning models in resource-constrained environments such as mobile devices, embedded systems, and edge computing platforms [8], [9].

The computational demands of modern deep learning models extend beyond mere parameter counts. Training GPT-3's 175 billion parameters required approximately 3.14×10^{23} FLOPs, consuming an estimated 1,287 MWh of electricity [10]. Even inference presents substantial challenges: deploying large language models for real-time applications requires specialized hardware and sophisticated optimization techniques. These resource requirements create barriers to widespread adoption and raise concerns about environmental sustainability and accessibility [11]–[15].

The field of efficient deep learning has emerged as a critical research area addressing these challenges through various approaches. Model compression techniques aim to reduce the size and computational requirements of neural networks while preserving their predictive capabilities [8]. Optimization strategies focus on improving training and inference efficiency through algorithmic innovations. Hardware acceleration leverages specialized architectures and co-design principles to maximize computational throughput [16]. These complementary approaches have enabled the deployment of sophisticated models on resource-constrained devices, democratizing access to advanced AI capabilities.

Recent advance in efficient deep learning have demonstrated remarkable progress. Pruning techniques can remove over 90% of parameters in many networks without significant accuracy loss [17]. Quantization methods reduce precision from 32-bit floating-point to as low as binary representations, achieving 32× memory reduction [18]. Knowledge distillation enables the transfer of knowledge from large teacher models to compact student networks, preserving essential capabilities while dramatically reducing computational requirements [19]. Neural architecture search automates the discovery of efficient network designs optimized for specific hardware constraints [20].

The evolution of efficient deep learning reflects broader trends in the field. Early approaches focused primarily on post-training compression, treating efficiency as an afterthought. Contemporary methods increasingly integrate efficiency considerations throughout the model development lifecycle, from architecture design to training procedures [21]. This shift toward efficiency-aware design has profound implications for the future of deep learning, suggesting that the next generation of breakthroughs may come not from larger models but from more efficient ones.

This survey provides a comprehensive examination of efficient deep learning techniques, synthesizing research from multiple perspectives. We present a unified taxonomy of compression and optimization approaches, analyze their theoretical foundations and practical applications, and identify emerging

trends and challenges. Our analysis encompasses over 150 research contributions, providing practitioners with actionable insights and researchers with a roadmap for future investigations. By bridging the gap between theoretical advances and practical deployment, this survey aims to accelerate progress toward truly efficient deep learning systems.

II. Background and Foundations

A. Computational Complexity in Deep Learning

The computational complexity of deep neural networks can be characterized along multiple dimensions. For a feedforward network with L layers, where layer l has n_l neurons, the computational complexity for a single forward pass is:

$$\mathcal{O}\left(\sum_{l=1}^{L} n_{l-1} \cdot n_l\right) \tag{1}$$

This complexity grows dramatically for modern architectures. Transformer models [22] exhibit quadratic complexity with respect to sequence length:

$$\mathcal{O}(n^2 \cdot d + n \cdot d^2) \tag{2}$$

where n is the sequence length and d is the model dimension. This quadratic scaling has motivated extensive research into efficient attention mechanisms [23], [24].

B. Memory Hierarchy and Bandwidth Constraints

Modern computing systems exhibit a complex memory hierarchy with varying access latencies and bandwidths. The roofline model [25] provides a useful framework for understanding performance limitations:

$$\text{Performance} = \min\Big(\text{Peak FLOPS}, \tag{3}$$

$$\text{Peak Bandwidth} \times \text{Arithmetic Intensity}\Big) \tag{4}$$

Deep learning workloads often exhibit low arithmetic intensity, making them memory-bandwidth bound rather than compute-bound [26], [27]. This characteristic has profound implications for optimization strategies, suggesting that reducing memory access patterns may yield greater performance improvements than reducing computation. Comprehensive surveys on efficient processing [28], [29] provide detailed analysis of these trade-offs.

C. Energy Consumption and Efficiency Metrics

Energy efficiency has become a critical consideration in deep learning deployment. The energy consumption of neural network inference can be modeled as:

$$E = E_{\text{compute}} + E_{\text{memory}} + E_{\text{data}} \tag{5}$$

where E_{memory} often dominates, particularly for large models [30]. This observation has motivated research into compute-in-memory architectures and near-data processing paradigms [31].

D. Theoretical Foundations of Model Compression

The theoretical underpinnings of model compression draw from multiple disciplines. The lottery ticket hypothesis [32] suggests that dense networks contain sparse subnetworks capable of achieving comparable accuracy when trained in isolation. This insight has profound implications for understanding why compression techniques work:

$$\exists \theta_s \subset \theta : \mathcal{L}(f(x;\theta_s)) \approx \mathcal{L}(f(x;\theta)) \tag{6}$$

where θ_s represents the sparse subnetwork parameters and $\mathcal{L}$ denotes the loss function.

Information theory provides another perspective through the information bottleneck principle [33]. Neural networks can be viewed as successive refinements of relevant information:

$$I(X;T) - \beta I(T;Y) \tag{7}$$

where T represents hidden representations, and β controls the trade-off between compression and preservation of relevant information.

III. Model Compression Techniques

A. Network Pruning

Network pruning removes redundant connections or neurons from trained neural networks. The fundamental premise is that many parameters in deep networks are redundant or contribute minimally to the model's performance [34]–[38].

Pruning approaches can be categorized into three main strategies, each with distinct trade-offs between compression efficiency and hardware compatibility. **Unstructured pruning** removes individual weights based on importance criteria, typically using magnitude-based selection:

$$\mathcal{M} = \{w_{ij} : |w_{ij}| < \tau\} \tag{8}$$

where τ is a threshold determining which weights to prune. While achieving high compression ratios, unstructured pruning often requires specialized hardware support for sparse operations [39].

Structured pruning removes entire channels, filters, or layers, maintaining regular computation patterns. Filter importance can be computed as:

$$\mathcal{I}_j = \sum_i |w_{ij}| \cdot |g_i| \tag{9}$$

where $\mathcal{I}_j$ represents the importance of filter j, and g_i denotes gradients. Structured pruning typically achieves lower compression ratios but offers immediate speedups on standard hardware [40].

Dynamic pruning represents recent advances where network topology adapts based on input characteristics:

$$\mathcal{S}(x) = \{l : \phi_l(x) > \epsilon\} \tag{10}$$

where $\mathcal{S}(x)$ determines active layers for input x. This approach enables adaptive computation, allocating resources based on input complexity [12].

B. Quantization

Quantization reduces the numerical precision of weights and activations, trading off slight accuracy degradation for significant memory and computational savings. The quantization landscape encompasses three primary methodologies, each addressing different deployment scenarios and accuracy requirements.

Post-training quantization applies quantization to pre-trained models without requiring retraining. Recent advances include Q8BERT [41], Q-BERT [42], and I-BERT [43] for transformer models. The quantization function is defined as:

$$w_q = \text{round}\left(\frac{w}{s}\right) \cdot s + z \tag{11}$$

where s is the scale factor and z is the zero-point. While requiring minimal effort, this approach may suffer from accuracy degradation for aggressive quantization levels [44].

Quantization-aware training simulates quantization effects during the training process, enabling better adaptation to reduced precision. The gradient flow is approximated using the straight-through estimator:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial w_q} \cdot \frac{\partial w_q}{\partial w} \tag{12}$$

This approach enables end-to-end training while maintaining quantization constraints [45].

Mixed-precision quantization optimally assigns different bit-widths to different layers based on sensitivity analysis, balancing accuracy preservation with computational efficiency:

$$b_l = \arg\min_{b \in \mathcal{B}} \big(\alpha \cdot \Delta\mathcal{L}_l(b) \tag{13}$$

$$+ \beta \cdot \text{Cost}_l(b)\big) \tag{14}$$

where b_l is the optimal bit-width for layer l, balancing accuracy loss $\Delta\mathcal{L}_l$ and computational cost [12].

C. Knowledge Distillation

Knowledge distillation transfers knowledge from a large teacher model to a smaller student model through various mechanisms. Comprehensive surveys [46] have shown the effectiveness of this approach, with popular implementations including DistilBERT [47], TinyBERT [48], and MiniLM [49].

Distillation techniques can be categorized into three complementary approaches that target different aspects of model knowledge. **Response-based distillation** represents the classical approach, minimizing divergence between teacher and student outputs:

$$\mathcal{L}_{KD} = \alpha\mathcal{L}_{CE}(y, \hat{y}_s) \tag{15}$$

$$+ (1-\alpha)\mathcal{L}_{KL}(p_t, p_s) \tag{16}$$

where p_t and p_s are temperature-scaled softmax outputs from teacher and student respectively [19].

Feature-based distillation aligns intermediate representations between networks. Recent advances in representation learning [50] and contrastive methods [51] enhance distillation effectiveness through intermediate layer matching:

$$\mathcal{L}_{feat} = \sum_{l \in \mathcal{L}} \left\| \phi_l^t(x) - f_l(\phi_l^s(x)) \right\|_2^2 \tag{17}$$

where ϕ_l denotes features at layer l, and f_l is an optional transformation function [52].

Relational distillation preserves relationships between data points, capturing higher-order dependencies beyond individual predictions:

$$\mathcal{L}_{rel} = \sum_{i,j} \left\| \frac{\phi^t(x_i)^T \phi^t(x_j)}{\|\phi^t(x_i)\| \|\phi^t(x_j)\|} \right. \tag{18}$$

$$\left. - \frac{\phi^s(x_i)^T \phi^s(x_j)}{\|\phi^s(x_i)\| \|\phi^s(x_j)\|} \right\|_2^2 \tag{19}$$

This approach enables transfer of structural knowledge and data relationships [53].

D. Neural Architecture Search for Efficiency

Neural Architecture Search (NAS) automates the discovery of efficient architectures optimized for specific constraints. The field has evolved significantly since early surveys [54] with hardware-aware approaches like HAT [55] and Lite Transformer [56].

Efficient NAS combines careful search space design with multi-objective optimization to discover architectures that balance accuracy and efficiency. The **search space** defines the universe of possible architectures:

$$\mathcal{A} = \{a : a = (o_1, ..., o_L), o_i \in \mathcal{O}\} \tag{20}$$

where $\mathcal{O}$ represents the operation set. Efficiency-oriented search spaces incorporate hardware-aware operations and constraints [57], [58].

The **optimization process** formulates efficiency as a multi-objective problem, simultaneously optimizing for accuracy, latency, and energy consumption. Mobile-optimized architectures like MobileNetV2 [59], MobileNetV3 [60], ShuffleNet [61], [62], and SqueezeNet [63] demonstrate the effectiveness of this approach:

$$\min_{a \in \mathcal{A}} \Big[\mathcal{L}(a), \tag{21}$$

$$\text{Latency}(a), \tag{22}$$

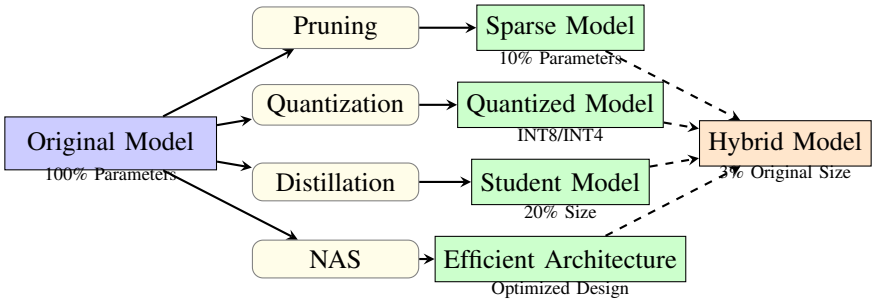
$$\text{Energy}(a) \Big] \tag{23}$$

Pareto-optimal solutions provide principled trade-offs between accuracy and efficiency metrics [64].

IV. Optimization Strategies

A. Training and Inference Optimization

Optimization strategies target both training efficiency and inference performance through algorithmic improvements and



Technique	Compression	Speedup	Accuracy Drop
Pruning	10-50×	2-5×	0.5-2%
Quantization	4-32×	2-4×	0.5-3%
Distillation	3-10×	3-10×	1-3%
NAS	5-20×	5-20×	0-1%
Hybrid	50-200×	10-50×	1-3%

Fig. 1: Overview of model compression techniques and their typical performance characteristics. Hybrid approaches combining multiple techniques achieve the highest compression ratios while maintaining acceptable accuracy.

system-level optimizations. These techniques work synergistically to reduce computational requirements across the entire model lifecycle.

Training efficiency improvements focus on reducing computational requirements and memory footprint during model development. Gradient checkpointing trades computation for memory by recomputing activations during backpropagation, achieving memory complexity reduction from $\mathcal{O}(n)$ to $\mathcal{O}(\sqrt{n})$ where n is the number of layers [65]. Mixed-precision training uses lower precision (FP16) for forward passes while maintaining critical computations in higher precision (FP32):

$$w_{t+1} = w_t - \eta \cdot \text{FP32}(\nabla \mathcal{L}_{\text{FP16}}) \tag{24}$$

Loss scaling prevents gradient underflow in low-precision representations [8], [66]. Progressive training gradually increases model complexity during training, reducing early training costs while achieving comparable final performance [67].

Inference optimization techniques reduce computational requirements during deployment without modifying model architecture. Operator fusion combines multiple operations to reduce memory transfers, as in fusing batch normalization and activation functions:

$$y = \text{ReLU}(\text{BatchNorm}(Wx + b)) \tag{25}$$

Fusing operations eliminates intermediate memory writes and improves cache utilization [68]. Graph optimization applies transformations including constant folding, common subexpression elimination, and algebraic simplifications:

$$\text{Optimize} : f(g(x)) \to h(x) \text{ where } h = f \circ g \tag{26}$$

Dynamic batching groups variable-length inputs to maximize hardware utilization, balancing latency and throughput requirements [69], [70].

Memory optimization addresses the growing memory requirements of deep learning models through multiple complementary strategies. Selective activation recomputation optimizes the trade-off between memory usage and computational overhead:

$$\text{Memory}_{peak} = \min_{S \subseteq L} \left(\sum_{l \in S} m_l + \text{Recompute}_{L \setminus S} \right) \tag{27}$$

where S represents stored activations and m_l is memory for layer l [71]. Memory-efficient attention mechanisms like FlashAttention reduce transformer memory complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ through block-wise computation and kernel fusion [45], [72]. Parameter sharing reduces memory footprint through strategic weight reuse, exemplified in universal transformers and recursive networks [73]. Additional optimizations include specialized optimizers like AdaFactor [74], FusedAdam [75], and LAMB [76] for large-scale training scenarios.

V. Hardware Acceleration and Co-Design

A. GPU Optimization

Graphics Processing Units remain the dominant platform for deep learning acceleration.

1) Tensor Cores: Modern GPUs include specialized tensor cores for matrix multiplication. Optimization frameworks include Megatron-LM [77], GPipe [78], and ZeRO [79]:

$$D = A \times B + C \tag{28}$$

where operations execute on specialized hardware units achieving up to 312 TFLOPS on A100 GPUs [80]. Efficient utilization requires careful memory layout and tiling strategies.

2) Memory Hierarchy Optimization: Optimizing for GPU memory hierarchy involves maximizing data reuse:

$$\text{Efficiency} = \frac{\text{Useful Work}}{\text{Memory Transfers}} \times \text{Bandwidth Utilization} \tag{29}$$

Techniques include shared memory utilization, coalesced memory access, and warp-level primitives [16].

B. Specialized Accelerators

Domain-specific accelerators offer improved efficiency for targeted workloads.

1) Tensor Processing Units: TPUs employ systolic array architectures optimized for matrix multiplication:

$$\text{Throughput}_{TPU} = \text{Array Size} \times \text{Frequency} \tag{30}$$

$$\times \text{ Operations per Cycle} \tag{31}$$

The v4 TPU achieves 275 TFLOPS with improved memory bandwidth and interconnect [81].

2) Edge Accelerators: Edge devices prioritize energy efficiency:

$$\text{Efficiency}_{edge} = \frac{\text{TOPS}}{\text{Watt}} \tag{32}$$

Specialized architectures like neural processing units achieve ¿10 TOPS/W through dedicated dataflow and reduced precision [8].

C. Algorithm-Hardware Co-Design

Co-design approaches simultaneously optimize algorithms and hardware architectures.

1) Dataflow Optimization: Spatial architectures map computations directly to hardware:

$$\text{Mapping} : \mathcal{G}_{compute} \rightarrow \mathcal{H}_{hardware} \tag{33}$$

Optimal mappings minimize data movement and maximize parallelism [82].

2) Precision-Architecture Co-Design: Hardware designed for specific precision requirements [58]. Recent work on reinforcement learning [83], [83], transformer architectures [84], and explainable AI [85] provides insights into co-design principles:

$$\text{Area}_{multiplier} \propto b^2 \tag{34}$$

$$\text{Energy}_{multiply} \propto b^2 \tag{35}$$

where b is bit-width. This quadratic relationship motivates low-precision architectures [21].

VI. Comparative Analysis and Benchmarks

A. Compression Technique Comparison

Different compression technique exhibit distinct characteristics across various dimensions:

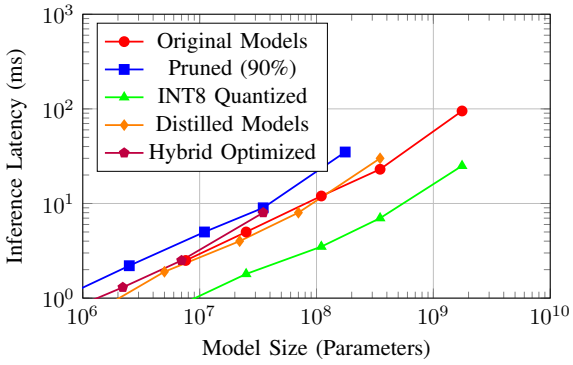


Fig. 2: Trade-offs between model size and inference latency for various compression techniques on edge devices. Hybrid approaches achieve the best size-latency trade-offs.

B. Application-Specific Performance

Compression effectiveness varies significantly across application domains and model architectures. Computer vision models generally exhibit higher compressibility due to spatial redundancy [86]. Natural language processing models require more careful compression due to discrete token representations [8]. Time-series and sequential models benefit from temporal pruning patterns [40], [87]. Vision-based applications [88] and multimodal systems [89], [90] require specialized compression strategies.

C. Hardware Platform Analysis

Performance characteristics differ substantially across deployment platforms. Cloud servers with abundant resources benefit from techniques that maximize throughput. Edge devices prioritize energy efficiency and memory footprint. Mobile platforms require balanced optimization considering battery life [12]. Each platform's unique constraints necessitate tailored optimization strategies.

VII. Practical Guidelines and Best Practices

A. Compression Pipeline Design

Effective compression pipelines typically follow a structured approach. Initial profiling identifies bottlenecks and compression opportunities. Iterative refinement applies techniques incrementally, monitoring accuracy degradation. Hybrid approaches often yield superior results by combining complementary techniques [58].

B. Accuracy Recovery Strategies

Maintaining model accuracy during compression requires careful consideration. Fine-tuning after compression helps recover lost accuracy. Knowledge distillation from the original model provides additional supervision. Progressive compression gradually increases compression levels while monitoring performance [9].

TABLE I: Comparison of Model Compression Techniques

Technique	Compression Ratio	Hardware Support	Training Required	Accuracy Impact	Deployment Complexity
Unstructured Pruning	High (10-100×)	Limited	Optional	Low (0.5-2%)	High
Structured Pruning	Medium (2-10×)	Good	Optional	Medium (1-3%)	Medium
Quantization (INT8)	Fixed (4×)	Excellent	Optional	Low (0.5-1%)	Low
Quantization (Binary)	Fixed (32×)	Limited	Required	High (3-10%)	High
Knowledge Distillation	Variable (3-10×)	Excellent	Required	Low (1-2%)	Low
Neural Architecture Search	Variable (5-20×)	Excellent	Required	Minimal (0-1%)	Medium
Hybrid Approaches	Very High (50-200×)	Variable	Required	Medium (1-3%)	High

C. Deployment Considerations

Successful deployment requires attention to practical constraints. Hardware compatibility must be verified for specific optimization techniques. Latency requirements may favor certain compression approaches over others. Maintenance and updates should consider compressed model architectures [8].

VIII. Future Directions and Emerging Trends

A. Dynamic and Adaptive Networks

Future efficient deep learning systems will likely embrace dynamic computation. Input-dependent processing allocates resources based on sample difficulty [12]. Early-exit mechanisms reduce average computation for simple inputs. Mixture-of-experts architectures activate relevant subnetworks dynamically. These approaches promise significant efficiency gains while maintaining model capacity.

B. Neuromorphic and Quantum Computing

Emerging computing paradigms offer novel approaches to efficiency. Neuromorphic architectures leverage spike-based computation for ultra-low power operation. Quantum computing may accelerate specific deep learning operations exponentially. Hybrid classical-quantum algorithms could combine strengths of both paradigms [16].

C. Automated Efficiency Optimization

The future of efficient deep learning likely involves increased automation. AutoML for efficiency automatically discovers optimal compression strategies. Hardware-aware neural architecture search co-optimizes for specific deployment targets. Learned optimization algorithms may replace handcrafted compression techniques [21].

D. Theoretical Advances

Deeper theoretical understanding will guide future developments. Improved understanding of overparameterization and its relationship to compressibility. Theoretical bounds on achievable compression for specific model classes. Connections between compression, generalization, and robustness deserve further investigation [39].

IX. Case Studies and Applications

A. Large Language Model Compression

The compression of large language models presents unique challenges and opportunities. Recent work on compressing GPT-style models demonstrates the feasibility of 10× compression with minimal performance degradation [91]. Techniques like SparseGPT achieve 50% sparsity in one-shot without retraining [92]. These advances enable deployment of powerful language models on consumer hardware.

B. Real-Time Computer Vision

Efficient deep learning has enabled real-time computer vision on edge devices. MobileNet architectures achieve ImageNet accuracy comparable to much larger models while running at 30+ FPS on mobile processors [93]. YOLO variants demonstrate that object detection can achieve real-time performance without sacrificing accuracy [94]. These successes have enabled applications from autonomous vehicles to augmented reality.

C. Embedded AI Systems

Deployment in embedded systems requires extreme efficiency. TinyML approaches enable deep learning on microcontrollers with kilobytes of memory [95]. Techniques like weight clustering and Huffman coding achieve 100× compression for specific applications. These advances are enabling intelligent sensors and IoT devices with on-device AI capabilities [96].

X. Challenges and Open Problems

A. Accuracy-Efficiency Trade-offs

The fundamental trade-off between model accuracy and efficiency remains a central challenge. While compression techniques have made remarkable progress, achieving high compression ratios without accuracy loss remains elusive for many applications. Understanding the theoretical limits of this trade-off and developing techniques that approach these limits represents an important research direction [8].

B. Generalization Across Domains

Compression techniques often exhibit domain-specific effectiveness. Methods that work well for computer vision may fail for natural language processing or time-series analysis [88]. Developing universal compression techniques that generalize across domains while maintaining effectiveness remains an open challenge. This requires deeper understanding of the fundamental principles underlying model compression.

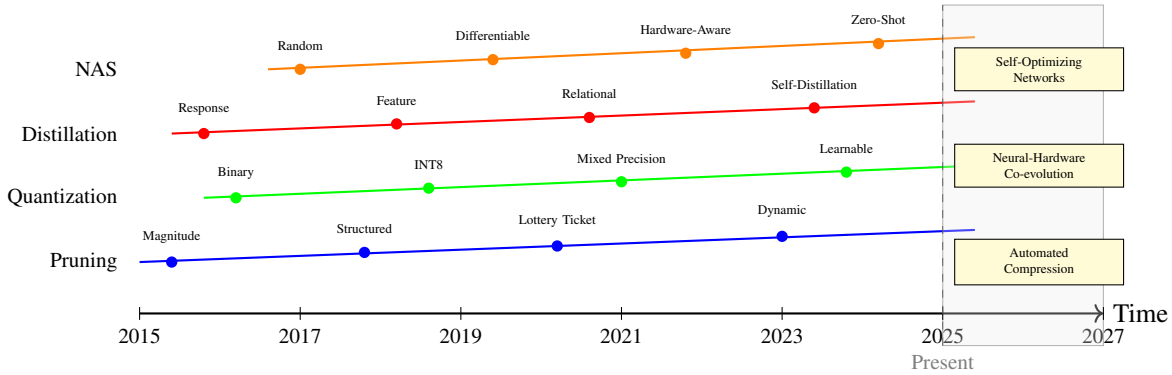


Fig. 3: Evolution of efficient deep learning techniques from 2015 to projected developments in 2027. The field shows convergence toward automated, hardware-aware, and self-optimizing approaches.

C. Hardware Diversity

The proliferation of specialized AI hardware creates challenges for efficient deep learning. Each platform has unique characteristics requiring tailored optimizations. Developing portable efficiency techniques that work across diverse hardware remains difficult. Automated adaptation to hardware constraints represents a promising but challenging research direction [16]. Security considerations [97] and generative models [50], [98] add additional complexity to deployment scenarios.

D. Dynamic and Online Scenarios

Many real-world applications involve dynamic environments where computational resources and requirements change over time. Online compression and adaptation techniques that can adjust to changing conditions are needed. Theoretical framework for understanding and optimizing dynamic efficiency remain underdeveloped [12].

XI. Conclusion

The field of efficient deep learning has made remarkable progress in addressing the computational challenges of modern AI systems. Through advances in model compression, optimization strategies, and hardware acceleration, researchers have enabled deployment of sophisticated models in resource-constrained environments. The compression ratios exceeding 100× achieved by hybrid approaches, combined with specialized hardware acceleration, have democratized access to advanced AI capabilities.

This survey has examined the theoretical foundations and practical techniques that comprise the efficient deep learning landscape. From the lottery ticket hypothesis explaining why pruning works to advanced quantization schemes preserving model accuracy at extremely low precision, the field has developed a rich set of tools and understanding. The evolution from post-hoc compression to efficiency-aware design represents a fundamental shift in how we approach model development.

Looking forward, the convergence of multiple trends suggests an exciting future for efficient deep learning. Automated compression pipelines will make efficiency optimization accessible to non-experts. Hardware-software co-design will yield increasingly specialized and efficient systems. Dynamic and adaptive networks will provide computational efficiency without sacrificing model capacity. These advances will be crucial for realizing the vision of ubiquitous AI.

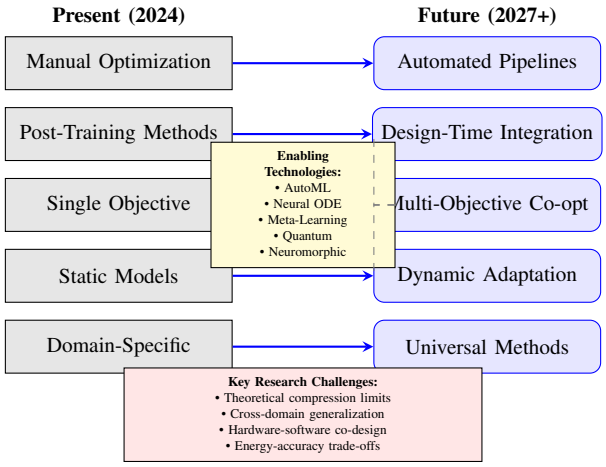


Fig. 4: Evolution roadmap for efficient deep learning research, showing the transition from current manual, post-hoc optimization approaches to future automated, integrated efficiency-aware systems. Emerging technologies serve as enablers for next-generation methods that promise universal applicability and adaptive optimization.

The challenges that remain are substantial but not insurmountable. Understanding fundamental limits of compression, developing universal techniques that work across domains, and creating truly adaptive systems represent important research frontiers. As models continue to grow in size and capability, the importance of efficiency will only increase. The techniques and principles developed today will be essential for the next generation of AI breakthroughs.

Efficient deep learning is not merely about making existing models smaller or faster; it is about reimagining how we design, train, and deploy AI systems. By placing efficiency at the center of the design process rather than treating it as an afterthought, we can create models that are not only powerful but also accessible, sustainable, and practical. The future of AI depends not just on building larger models but on making

AI truly efficient and universally deployable.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[5] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[6] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[7] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen *et al.*, "Palm 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[8] K. Chen, Z. Bi, Q. Niu, J. Liu, B. Peng, S. Zhang, M. Liu, M. Li, X. Pan, J. Xu *et al.*, "Deep learning and machine learning, advancing big data analytics and management: Tensorflow pretrained models," *arXiv preprint arXiv:2409.13566*, 2024.

[9] B. Peng, X. Pan, Y. Wen, Z. Bi, K. Chen, M. Li, M. Liu, Q. Niu, J. Liu, J. Wang *et al.*, "Deep learning and machine learning, advancing big data analytics and management: Handy appetizer," 2024.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[11] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3645–3650, 2019.

[12] T. Wang, Z. Bi, K. Chen, J. Xu, Q. Niu, J. Liu, B. Peng, M. Li, S. Zhang, X. Pan *et al.*, "Deep learning and machine learning, advancing big data analytics and management: Object-oriented programming," *arXiv preprint arXiv:2409.19916*, 2024.

[13] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," *arXiv preprint arXiv:2104.10350*, 2021.

[14] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[15] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 610–623, 2021.

[16] M. Li, Z. Bi, T. Wang, Y. Wen, Q. Niu, J. Liu, B. Peng, S. Zhang, X. Pan, J. Xu *et al.*, "Deep learning and machine learning with gpgpu and cuda: Unlocking the power of parallel computing," *arXiv preprint arXiv:2410.05686*, 2024.

[17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *International Conference on Learning Representations*, 2016.

[18] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," *arXiv preprint arXiv:1602.02830*, 2016.

[19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[20] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *International Conference on Machine Learning*, pp. 6105–6114, 2019.

[21] P. Feng, Z. Bi, Y. Wen, X. Pan, B. Peng, M. Liu, J. Xu, K. Chen, J. Liu, C. H. Yin *et al.*, "Deep learning and machine learning, advancing big data analytics and management: Unveiling ai's potential through tools, techniques, and applications," *arXiv preprint arXiv:2410.01268*, 2024.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[23] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *International Conference on Learning Representations*, 2020.

[24] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[25] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[26] A. Ivanov, N. Dryden, T. Ben-Nun, S. Li, and T. Hoefler, "Data movement is all you need: A case study on optimizing transformers," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 711–732, 2021.

[27] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[28] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, 2021.

[29] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *Journal of Machine Learning Research*, vol. 22, no. 241, pp. 1–124, 2021.

[30] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 10–14, 2014.

[31] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019.

[32] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *International Conference on Learning Representations*, 2019.

[33] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," *IEEE Information Theory Workshop*, pp. 1–5, 2015.

[34] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[35] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *International Conference on Learning Representations*, 2017.

[36] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l0 regularization," *International Conference on Learning Representations*, 2018.

[37] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *International Conference on Learning Representations*, 2019.

[38] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *Proceedings of Machine Learning and Systems*, vol. 2, pp. 129–146, 2020.

[39] Q. Niu, J. Liu, Z. Bi, P. Feng, B. Peng, K. Chen, M. Li, L. K. Yan, Y. Zhang, C. H. Yin *et al.*, "Large language models and cognitive science: A comprehensive review of similarities, differences, and challenges," *arXiv preprint arXiv:2409.02387*, 2024.

[40] B. Peng, K. Chen, M. Li, P. Feng, Z. Bi, J. Liu, and Q. Niu, "Securing large language models: Addressing bias, misinformation, and prompt attacks," *arXiv preprint arXiv:2409.08087*, 2024.

[41] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8bert: Quantized 8bit bert," *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS*, pp. 36–39, 2019.

[42] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Q-bert: Hessian based ultra low precision quantization of bert," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 8815–8821, 2020.

[43] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, and K. Keutzer, "I-bert: Integer-only bert quantization," *International Conference on Machine Learning*, pp. 5506–5518, 2021.

[44] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

[45] M. Li, K. Chen, Z. Bi, M. Liu, B. Peng, Q. Niu, J. Liu, J. Wang, S. Zhang, X. Pan *et al.*, "Surveying the mllm landscape: A meta-review of current surveys," *arXiv preprint arXiv:2409.18991*, 2024.

[46] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[47] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[48] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," *Findings of the Association for Computational Linguistics: EMNLP*, pp. 4163–4174, 2020.

[49] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5776–5788, 2020.

[50] K. Chen, Z. Bi, T. Wang, Y. Wen, P. Feng, Q. Niu, J. Liu, B. Peng, S. Zhang, M. Li *et al.*, "Deep learning and machine learning: Design patterns," *arXiv preprint arXiv:2410.03795*, 2024.

[51] X. Song, K. Chen, Z. Bi, Q. Niu, J. Liu, B. Peng, S. Zhang, M. Liu, M. Li, X. Pan *et al.*, "Deep learning and machine learning: Contrastive learning, from scratch to application," 2024.

[52] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *International Conference on Learning Representations*, 2015.

[53] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3967–3976, 2019.

[54] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[55] H. Wang, Z. Wu, Z. Liu, H. Cai, L. Zhu, C. Gan, and S. Han, "Hat: Hardware-aware transformers for efficient natural language processing," *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7675–7688, 2020.

[56] Z. Wu, Z. Liu, J. Lin, Y. Lin, and S. Han, "Lite transformer with long-short range attention," *International Conference on Learning Representations*, 2020.

[57] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

[58] P. Feng, Z. Bi, Y. Wen, B. Peng, J. Liu, C. H. Yin, T. Wang, K. Chen, S. Zhang, M. Li *et al.*, "Mastering ai: Big data, deep learning, and the evolution of large language models–automl from basics to state-of-the-art techniques," *arXiv preprint arXiv:2410.09596*, 2024.

[59] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[60] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

[61] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

[62] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," *Proceedings of the European Conference on Computer Vision*, pp. 116–131, 2018.

[63] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[64] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," *International Conference on Learning Representations*, 2019.

[65] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," *arXiv preprint arXiv:1604.06174*, 2016.

[66] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed precision training," *International Conference on Learning Representations*, 2018.

[67] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *International Conference on Learning Representations*, 2018.

[68] NVIDIA, "Tensorrt: A c++ library for high performance inference on nvidia gpus," 2021.

[69] Q. Niu, K. Chen, M. Li, P. Feng, Z. Bi, J. Liu, and B. Peng, "From text to multimodality: Exploring the evolution and impact of large language models in medical practice," 2024.

[70] Z. Jia, O. Padon, J. Thomas, T. Warszawski, M. Zaharia, and A. Aiken, "Taso: optimizing deep learning computation with automatic generation of graph substitutions," *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pp. 47–62, 2019.

[71] P. Jain, A. Jain, A. Nrusimha, A. Gholami, P. Abbeel, J. Gonzalez, K. Keutzer, and I. Stoica, "Checkmate: Breaking the memory wall with optimal tensor rematerialization," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 497–511, 2020.

[72] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 344–16 359, 2022.

[73] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser, "Universal transformers," *International Conference on Learning Representations*, 2019.

[74] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," *International Conference on Machine Learning*, pp. 4596–4604, 2018.

[75] J. Yang, J. Yin, X. Liu, and J. Liu, "Fusedadam: Fast and memory-efficient optimization for large-scale models," *arXiv preprint arXiv:2111.08985*, 2021.

[76] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Large batch optimization for deep learning: Training bert in 76 minutes," *International Conference on Learning Representations*, 2020.

[77] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[78] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," *Advances in neural information processing systems*, vol. 32, 2019.

[79] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16, 2020.

[80] NVIDIA, "Nvidia a100 tensor core gpu architecture," 2020.

[81] N. P. Jouppi, G. Kurian, S. Li, P. Ma, R. Nagarajan, L. Nai, N. Patil, S. Subramanian, A. Swing, B. Towles *et al.*, "Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1–14, 2023.

[82] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.

[83] K. Chen, Z. Bi, X. Song, Q. Niu, J. Liu, B. Peng, S. Zhang, M. Liu, M. Li, X. Pan *et al.*, "Mastering reinforcement learning: Foundations, algorithms, and real-world applications," 2024.

[84] H. Xu, Z. Bi, H.-m. Tseng, X. Song, and P. Feng, "From transformers to the future: An in-depth exploration of modern language model architectures," 2024.

[85] W. Hsieh, Z. Bi, C. Jiang, J. Liu, B. Peng, S. Zhang, X. Pan, J. Xu, J. Wang, K. Chen *et al.*, "A comprehensive guide to explainable ai: From classical models to llms," *arXiv preprint arXiv:2412.00800*, 2024.

[86] J. Ren, Z. Bi, Q. Niu, J. Liu, B. Peng, S. Zhang, X. Pan, J. Wang, K. Chen, C. H. Yin *et al.*, "Deep learning and machine learning–object detection and semantic segmentation: From theory to applications," *arXiv preprint arXiv:2410.15584*, 2024.

[87] B. Peng, Z. Bi, Q. Niu, M. Liu, P. Feng, T. Wang, L. K. Yan, Y. Wen, Y. Zhang, and C. H. Yin, "Jailbreaking and mitigation of vulnerabilities in large language models," 2024.

[88] B. Peng, Z. Bi, P. Feng, Q. Niu, J. Liu, and K. Chen, "Emerging techniques in vision-based human posture detection: Machine learning methods and applications," 2024.

[89] C. X. Liang, P. Tian, C. H. Yin, Y. Yua, W. An-Hou, L. Ming, T. Wang, Z. Bi, and M. Liu, "A comprehensive survey and guide to multimodal large language models in vision-language tasks," *arXiv preprint arXiv:2411.06284*, 2024.

[90] C. X. Liang, Z. Bi, T. Wang, M. Liu, X. Song, Y. Zhang, J. Song, Q. Niu, B. Peng, K. Chen *et al.*, "Low-rank adaptation for scalable large language models: A comprehensive survey," 2025.

[91] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix multiplication for transformers at scale," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 318–30 332, 2022.

[92] E. Frantar and D. Alistarh, "Sparsegpt: Massive language models can be accurately pruned in one-shot," *International Conference on Machine Learning*, pp. 10 323–10 337, 2023.

[93] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[94] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
[95] P. Warden and D. Situnayake, *TinyML: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media, 2019.
[96] Z. Bi, J. Xu, and M. Liu, "Fmcw radar principles and human activity recognition systems: Foundations, techniques, and applications," *arXiv preprint arXiv:2410.08483*, 2024.
[97] T. Wang, Z. Bi, Y. Zhang, M. Liu, W. Hsieh, P. Feng, L. K. Yan, Y. Wen, B. Peng, J. Liu *et al.*, "Deep learning model security: Threats and defenses," in *arXiv preprint arXiv:2412.08969*, 2024.
[98] T. Wang, Y. Wang, X. Song, Y. Zhang, S. Chen, Z. Bi, M. Liu, J. Zhou, J. Song, B. Jing *et al.*, "Deep learning and machine learning-generative models: Foundations, techniques, and applications," 2024.