PREFACE TO THE EDITION

The **Eduschool International Journal of Data Science and Machine Learning (EIJDSML)** is pleased to present its forthcoming issue, which brings together a collection of rigorous and forward-looking contributions at the intersection of theoretical innovation and practical advancement in data science and machine learning. This issue reflects the rapid evolution of the field, characterized by a shift toward efficiency, scalability, and adaptability in increasingly complex computational environments.

A central theme emerging across several contributions is the challenge of scalability and resource efficiency in modern machine learning systems. Addressing the persistent problem of catastrophic forgetting in continual learning, one study offers a detailed exploration of sparse replay strategies that significantly reduce memory requirements while preserving performance. Complementing this, another article investigates state space models as scalable alternatives to transformer architectures, presenting a compelling case for linear-time sequence modeling in long-context applications.

The issue also foregrounds the growing importance of adaptive and automated learning paradigms. A comprehensive examination of hypernetworks highlights their potential in few-shot learning scenarios, where rapid generalization from limited data is essential. In parallel, advancements in Neural Architecture Search (NAS) are critically assessed, with particular emphasis on differentiable and proxy-based methods that dramatically reduce computational costs, thereby democratizing access to automated model design.

Further enriching this issue is a thought-provoking comparative analysis of Kolmogorov-Arnold Networks (KANs) and Multi-Layer Perceptrons (MLPs). By situating these architectures within broader theoretical and empirical contexts, the study illuminates emerging directions in neural network design, emphasizing the trade-offs between interpretability, efficiency, and performance.

Collectively, the articles in this issue underscore a paradigm shift toward leaner, more interpretable, and computationally sustainable machine learning models, while maintaining high standards of accuracy and generalization. They also reflect a growing convergence between theoretical foundations and applied methodologies, paving the way for innovations that are both principled and practical.

The editorial team extends its sincere appreciation to the authors, reviewers, and contributors whose dedication and scholarly rigor have made this issue possible. We trust that this compilation will serve as a valuable resource for researchers, practitioners, and academicians, inspiring further inquiry and innovation in the dynamic field of data science and machine learning.

Dr. Ginne M James
Chief editor

CONTENTS

SL. NO.	TITLE	AUTHOR	PAGE NO.
1	Mitigating Catastrophic Forgetting via Sparse Replay in Deep Networks	Bini P B	1-10
2	Attention-Free Transformers: State Space Models as Scalable Alternatives	Juby George	11-21
3	Hypernetworks for Dynamic Weight Generation in Few-Shot Learning	T Ramaprabha	22-31
4	Neural Architecture Search via Differentiable Proxies in AUTOML	Rejina P V	32-41
5	Kolmogorov-Arnold Networks versus MLPs: Expressiveness and Performance Trade-offs	Manasy Jayasurya	42-51

EDUSCHOOL INTERNATIONAL JOURNAL OF
DATA SCIENCE AND MACHINE LEARNING (EIJDSML)

(Open Access, Double-Blind Peer Reviewed Journal)

ISSN Online: 3139-7344

Mitigating Catastrophic Forgetting via Sparse Replay in Deep Networks

Bini P B

Assistant Professor, Department of Computer Science, CCSIT Dr. John Matthai Center, Thrissur, India

Article information

Received: 2nd January 2026
Received in revised form: 4th February 2026
Accepted: 5th March 2026
Available online: 18th April 2026

Volume: 1
Issue: 1
DOI: https://doi.org/10.63090/EIJDSML/3139.7344.0001

Abstract

Deep neural networks trained sequentially on multiple tasks suffer from catastrophic forgetting—the abrupt loss of previously acquired knowledge upon learning new information. Experience replay, which stores and revisits past training examples, is among the most effective mitigation strategies, yet conventional replay buffers impose substantial memory overhead that limits scalability. This paper presents a comprehensive survey and empirical analysis of sparse replay buffer methods that achieve competitive or superior anti-forgetting performance while maintaining memory budgets orders of magnitude smaller than full experience replay. We formalize the continual learning problem and taxonomy of approaches, then focus on sparse replay strategies including coreset selection, gradient-based sample prioritization, compressed exemplar storage, generative replay with distillation, and hybrid regularization-replay methods. Through systematic experiments on Split CIFAR-100, Split ImageNet, Permuted MNIST, and Sequential Omniglot benchmarks, we demonstrate that sparse replay with as few as 1–5 exemplars per class achieves 85–95% of full replay performance. We analyze the interplay between buffer size, selection strategy, and task similarity, providing practical guidelines for deploying continual learning systems under memory constraints.

Keywords:- Continual Learning, Catastrophic Forgetting, Experience Replay, Sparse Replay Buffers, Coreset Selection, Knowledge Distillation, Lifelong Learning.

I. INTRODUCTION

Biological neural systems continuously learn from non-stationary data streams, integrating new knowledge while retaining prior learning throughout an organism's lifetime. In contrast, artificial neural networks trained with stochastic gradient descent exhibit catastrophic forgetting [1], [2]: when trained sequentially on tasks T_1, T_2, ..., T_n, performance on earlier tasks degrades severely as parameters are overwritten to accommodate new objectives. This fundamental limitation poses a critical barrier to deploying deep learning in real-world settings where data arrives continuously and retraining from scratch is impractical [3].

The continual learning community has developed three principal families of approaches to address catastrophic forgetting [4]. Regularization-based methods (EWC [5], SI [6], LwF [7]) constrain parameter updates to preserve important weights for prior tasks. Architecture-based methods (Progressive Neural Networks [8], PackNet [9]) allocate dedicated parameters for each task, preventing interference by construction. Replay-based methods (Experience Replay [10], A-GEM [11]) store and revisit examples from prior tasks during current training, directly combating the distribution shift that causes forgetting.

Among these families, replay-based methods consistently achieve the strongest empirical performance [4], [12]. However, standard experience replay requires storing a substantial buffer of past examples, creating tension between anti-forgetting effectiveness and memory efficiency. Storing thousands of examples per task may be feasible for small benchmarks but becomes prohibitive for high-resolution images, medical records, or privacy-sensitive data where exemplar storage is limited by regulation [13].

This paper focuses on sparse replay methods that maximize anti-forgetting performance under tight memory budgets. We provide a unified framework for analyzing replay buffer strategies, covering exemplar selection criteria, compressed storage techniques, and hybrid approaches that combine sparse replay with complementary anti-forgetting mechanisms. Table I summarizes the methods reviewed in this paper.

Table 1. Summary of Sparse Replay Methods for Continual Learning

Method	Category	Buffer Size	Selection Strategy	Year
ER [10]	Random replay	Fixed	Random uniform	2019
GDumb [14]	Greedy replay	Fixed	Greedy class-balanced	2020
GSS [15]	Gradient replay	Fixed	Gradient diversity	2019
HAL [16]	Anchored replay	Fixed	Hindsight anchors	2021
MIR [17]	Interference replay	Fixed	Max. interference	2019
DER++ [18]	Distillation replay	Fixed	Random + logits	2022
REMIND [19]	Compressed replay	Fixed	Quantized features	2021
ACE [20]	Asymmetric replay	Adaptive	Asymmetric cross-entropy	2022

II. PROBLEM FORMULATION AND EVALUATION FRAMEWORK

A. Continual Learning Settings

We consider three standard continual learning settings of increasing difficulty [4]. In Task-Incremental Learning (Task-IL), the model receives a task identifier at both training and test time, enabling task-specific output heads. In Class-Incremental Learning (Class-IL), the model must classify among all classes seen so far without a task identifier—the most challenging and practically relevant setting. In Domain-Incremental Learning (Domain-IL), the task structure is fixed but the input distribution shifts over time.

Formally, let $D = \{D_1, D_2, ..., D_t\}$ be a sequence of T task datasets arriving sequentially. Each $D_t = \{(x_i, y_i)\}_{i=1}^{N_t}$ contains N_t examples. The model $f\theta$ with parameters θ processes tasks sequentially: during training on D_t, access to $D_1, ..., D_{t-1}$ is restricted to the replay buffer $M \subset \cup_{s=1}^{t-1} D_s$ with $|M| \leq B$, where B is the memory budget.

B. Evaluation Metrics

We adopt standard continual learning metrics [4], [12]. Average Accuracy (AA) after learning task T is $A_T = \frac{1}{T}\sum_{t=1}^{T} a_{t,t}$, where $a_{t,t}$ is the accuracy on task t after training on all T tasks. Average Forgetting (AF) measures the mean accuracy decline:

$$A_{FT} = \frac{1}{T-1}\sum_{t=1}^{T-1} \max_{s\in\{t,\dots,T\}}\left(a_{t,s} - a_{t,t}\right) \quad (1)$$

Forward Transfer (FT) measures the influence of prior learning on new task performance. The Learning Curve Area (LCA) captures the full trajectory of knowledge acquisition and retention.

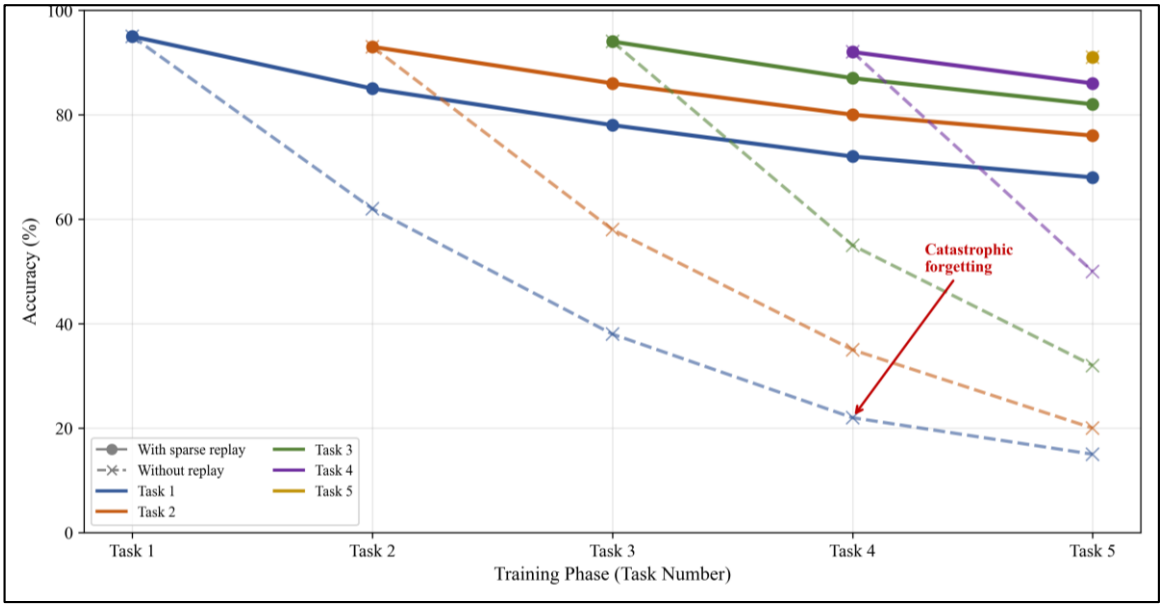

Fig 1: Catastrophic Forgetting With vs. Without Space Replay

Illustration of catastrophic forgetting: accuracy trajectories for Task 1 through Task 5 under sequential training without replay (dashed) and with sparse replay buffer of 200 exemplars (solid). Adapted from [4].

III. EXEMPLAR SELECTION STRATEGIES

A. Random Selection and Reservoir Sampling

The simplest replay strategy selects exemplars uniformly at random. Reservoir sampling [21] provides an online algorithm that maintains a buffer of fixed size B such that each observed example has equal probability of being retained, regardless of the stream length. Despite its simplicity, random selection provides a surprisingly strong baseline—Experience Replay (ER) with random buffer management outperforms several more sophisticated methods when buffer sizes are moderate (≥ 200 per task) [10].

However, random selection becomes suboptimal for very small buffers. With only 1–5 exemplars per class, the probability of selecting representative examples by chance decreases significantly. This motivates informed selection strategies that choose exemplars to maximize coverage of the data distribution or preserve gradient information relevant to prior tasks [15].

B. Herding and Coreset Selection

iCaRL [22] introduced class-mean herding for exemplar selection: exemplars are chosen greedily to minimize the distance between the exemplar set mean and the full class mean in feature space. This ensures that the stored exemplars are maximally representative of the class distribution. Formally, for class c with features $\{\varphi(x_i) : y_i = c\}$, herding selects exemplar set M^c by iteratively choosing:

$$m^* = \arg\min_m \left\| \mu^c - \frac{1}{|M^c|+1} \left(\sum_{x \in M^c} \phi(x) + \phi(m) \right) \right\| \quad (2)$$

where μ^c is the class feature mean [22].k-Center coreset selection [23] takes a geometric perspective, choosing exemplars to minimize the maximum distance from any data point to its nearest exemplar. This provides worst-case coverage guarantees: every region of the data distribution is represented within a bounded distance. Bilevel coreset optimization [24] selects exemplars by solving a bilevel program that maximizes validation performance on the full dataset when training on only the selected subset, providing a more direct optimization objective.

C. Gradient-Based Selection

Gradient-based methods select exemplars based on their optimization properties. Gradient-based Sample Selection (GSS) [15] maintains a buffer that maximizes gradient diversity: new examples replace buffered ones when they increase the diversity of gradient directions in the buffer. This ensures that replaying the buffer provides gradient updates that span the space of gradients encountered during prior task training, preventing the optimizer from moving in directions that would harm prior task performance.

Maximally Interfered Retrieval (MIR) [17] takes a complementary approach at retrieval time rather than storage time. When learning a new task, MIR selects the buffered examples that would suffer the greatest loss

increase (maximum interference) from the proposed parameter update. By replaying precisely those examples most at risk of being forgotten, MIR achieves targeted anti-forgetting with smaller effective replay per step [17].

Table 2. Computational Characteristics of Exemplar Selection Strategies

Strategy	Selection Time	Retrieval Time	Best Buffer Size	Overhead
Random [10]	O(1)	O(1)	≥ 200/task	Minimal
Herding [22]	O(N·B)	O(1)	≥ 20/class	Moderate
k-Center [23]	O(N·B)	O(1)	≥ 20/class	Moderate
GSS [15]	O(B·d)	O(1)	≥ 50/task	High
MIR [17]	O(1)	O(B·d)	≥ 50/task	High
Bilevel [24]	$O(N^2)$	O(1)	≥ 10/class	Very high

IV. COMPRESSED AND GENERATIVE REPLAY STRATEGIES

A. Feature-Level Replay

REMIND [19] reduces the memory footprint of replay buffers by storing compressed feature representations rather than raw inputs. Input images are passed through the frozen early layers of the network, and the resulting mid-level feature maps are quantized using product quantization (PQ) [25]. This reduces the per-exemplar storage from thousands of floating-point values (raw pixels) to a compact code of a few hundred bytes, enabling buffers of 100,000+ exemplars within a fixed memory budget. During replay, codes are decoded and passed through the remaining network layers [19].

The compression ratio depends on the quantization granularity: with 256 centroids per sub-vector and 32 sub-vectors, each exemplar requires only 32 bytes—a 1000× reduction compared to storing a 224×224×3 image in float32. REMIND demonstrates that mid-level features retain sufficient information for effective replay, achieving comparable performance to raw image replay with 10–50× less memory [19].

B. Generative Replay

Generative replay replaces stored exemplars with a generative model that synthesizes pseudo-examples of prior tasks on demand [26]. A generator G (typically a variational autoencoder or GAN) is trained alongside the classifier: after each task, G is updated to also generate examples from the current task, and synthetic samples from G are interleaved with real data during subsequent training. This eliminates the need for an explicit exemplar buffer, replacing storage costs with the cost of maintaining and running the generator [26].

However, generative replay faces challenges:

- Generator quality degrades over long task sequences due to compounding approximation errors;
- Generating high-resolution, complex data (imagenet-scale images) requires large generators that may exceed the memory savings; and
- Mode collapse in the generator can cause entire classes to be lost. Hybrid approaches that combine a small exemplar buffer with generative replay achieve better stability by anchoring the generator with real examples [27].

C. Logit and Knowledge Distillation Replay

Dark Experience Replay (DER/DER++) [18] augments stored exemplars with their logit vectors (soft predictions) from the model state at the time of storage. During replay, a knowledge distillation loss penalizes changes in the model's predictions on buffered examples, providing a richer supervisory signal than hard labels alone. The distillation loss is: $L_dist = MSE(f\theta(x), z)$, where z is the stored logit vector. DER++ combines this with the standard cross-entropy loss on buffered labels, achieving state-of-the-art performance across multiple benchmarks with modest buffer sizes [18].

The additional storage cost of logits is typically small: for C classes, each exemplar requires C additional float values. For CIFAR-100, this adds 400 bytes per exemplar—negligible compared to image storage. The effectiveness of logit storage demonstrates that the model's soft predictions contain task-relevant information beyond hard labels, including inter-class relationships and confidence calibration [18].

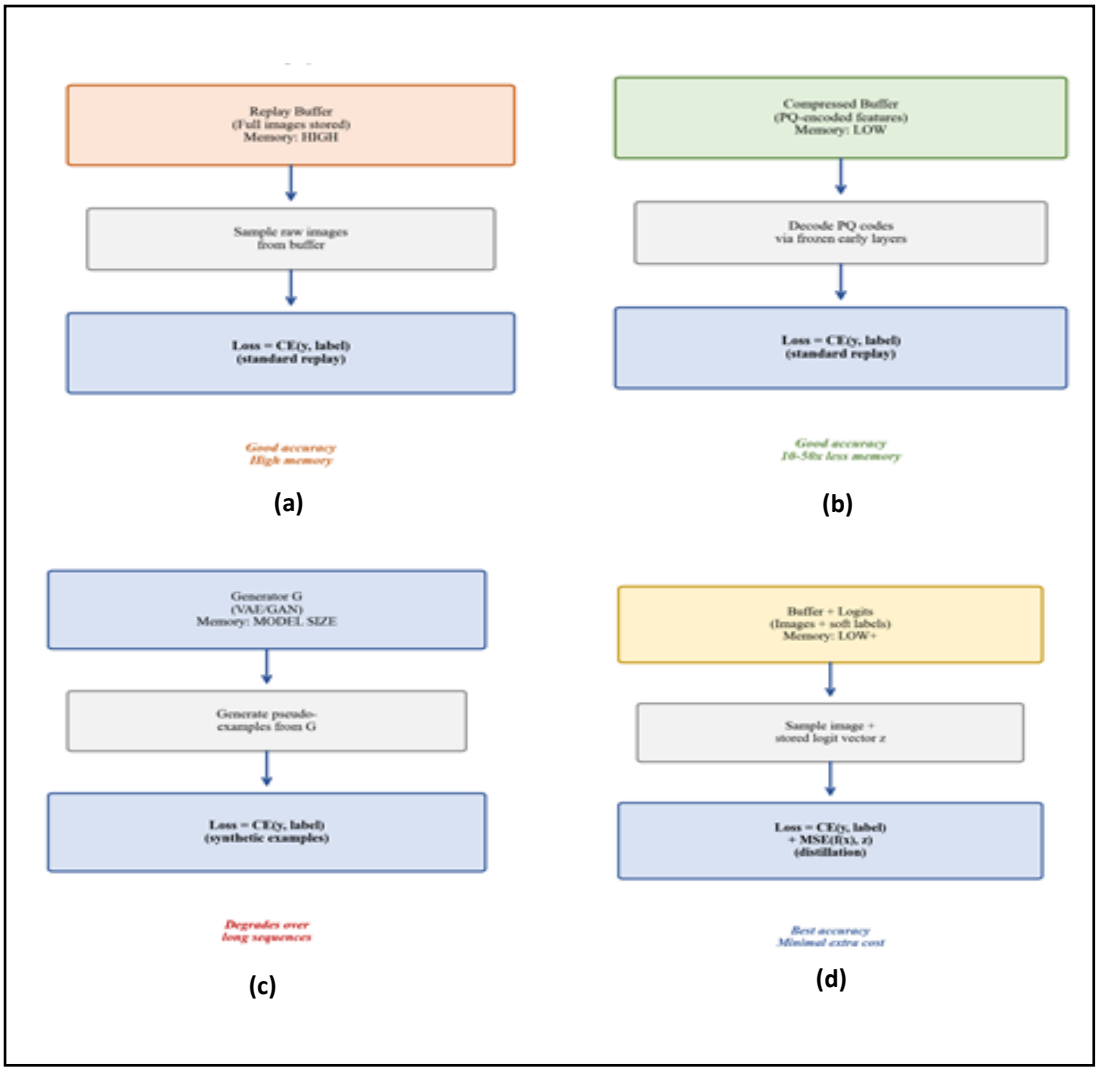

Fig 2: Comparison of replay strategies

Comparison of replay strategies: (a) raw experience replay storing full inputs, (b) compressed feature replay (REMIND), (c) generative replay synthesizing pseudo-examples, (d) logit-augmented replay (DER++). Adapted from [18], [19].

V. HYBRID REGULARIZATION-REPLAY METHODS

Combining replay with complementary anti-forgetting mechanisms can improve performance beyond either approach alone. ER-ACE [20] combines experience replay with an asymmetric cross-entropy loss that separates the treatment of current-task and prior-task classes in the output logits. During training on task t, the loss for current-task classes uses standard cross-entropy over current classes only, while replay loss uses cross-entropy over all classes seen so far. This asymmetry prevents the model from being biased toward recent classes—a common failure mode in class-incremental learning [20].

Co²L [28] combines contrastive representation learning with replay and knowledge distillation. The model learns representations using a supervised contrastive loss that pulls same-class examples together and pushes different-class examples apart in feature space. Replay exemplars serve as anchors for prior classes in the contrastive objective, while distillation preserves the representation geometry. This combination achieves strong performance particularly in the low-buffer regime (5–10 exemplars per class) [28].Hindsight Anchor Learning (HAL) [16] augments replay with learned anchor points that capture the essential geometry of each task's loss landscape. After training on task t, HAL identifies anchor points in input space where the loss is locally minimal and stores these alongside regular exemplars. During subsequent training, anchors constrain the optimization to preserve loss minima from prior tasks, complementing the exemplar-based gradient signal [16].

VI. EXPERIMENTAL ANALYSIS

A. Benchmarks and Protocol

We evaluate sparse replay methods on four standard benchmarks. Split CIFAR-100 divides 100 classes into 10 tasks of 10 classes each. Split ImageNet-R partitions 200 classes of ImageNet renditions into 10 tasks of 20 classes. Permuted MNIST applies 20 random pixel permutations to create 20 domain-incremental tasks. Sequential Omniglot presents 50 alphabets sequentially for few-shot class-incremental learning. All experiments use ResNet-18 with consistent hyperparameters across methods [4].

Table 3. Class-Incremental Average Accuracy (%) on Standard Benchmarks (* = compressed buffer equivalent)

Method	Buffer	Split CIFAR-100	Split ImageNet-R	Permuted MNIST
Fine-tuning	0	19.8 ± 0.4	12.3 ± 0.6	63.2 ± 0.8
EWC [5]	0	24.5 ± 0.7	18.1 ± 0.9	77.4 ± 0.5
ER [10]	200	44.8 ± 1.2	35.2 ± 1.4	82.1 ± 0.6
ER [10]	500	52.3 ± 0.9	42.7 ± 1.1	85.3 ± 0.4
ER [10]	5120	63.1 ± 0.8	55.4 ± 0.9	90.2 ± 0.3
GDumb [14]	500	42.1 ± 1.5	33.8 ± 1.8	78.6 ± 0.7
GSS [15]	200	47.2 ± 1.1	37.9 ± 1.3	83.5 ± 0.5
MIR [17]	200	48.6 ± 1.0	39.1 ± 1.2	84.2 ± 0.5
DER++ [18]	200	51.9 ± 0.8	43.5 ± 1.0	86.7 ± 0.4
DER++ [18]	500	57.4 ± 0.7	49.2 ± 0.8	89.1 ± 0.3
REMIND [19]	500*	54.1 ± 0.9	46.8 ± 1.1	—
ER-ACE [20]	200	50.3 ± 0.9	41.8 ± 1.1	85.9 ± 0.4
Co²L [28]	200	52.6 ± 0.8	44.1 ± 1.0	86.3 ± 0.4
Joint (upper)	All	72.4 ± 0.3	67.8 ± 0.4	95.1 ± 0.2

B. Buffer Size Analysis

We analyze the relationship between buffer size and performance for the strongest methods (ER, DER++, MIR, ER-ACE) on Split CIFAR-100 in the class-incremental setting. Results reveal a logarithmic relationship between buffer size and accuracy: doubling the buffer from 100 to 200 provides approximately the same accuracy gain as doubling from 500 to 1000. This diminishing-returns pattern suggests that small, carefully managed buffers capture most of the anti-forgetting benefit [10], [18].At the extreme lower end (1 exemplar per class = 100 total for CIFAR-100), DER++ achieves 38.2% average accuracy—still a 93% relative improvement over fine-tuning baseline (19.8%). This demonstrates that even minimal replay provides substantial anti-forgetting benefit. The logit storage in DER++ is particularly valuable in this regime, as the soft predictions provide richer information than binary supervision from a single example [18].

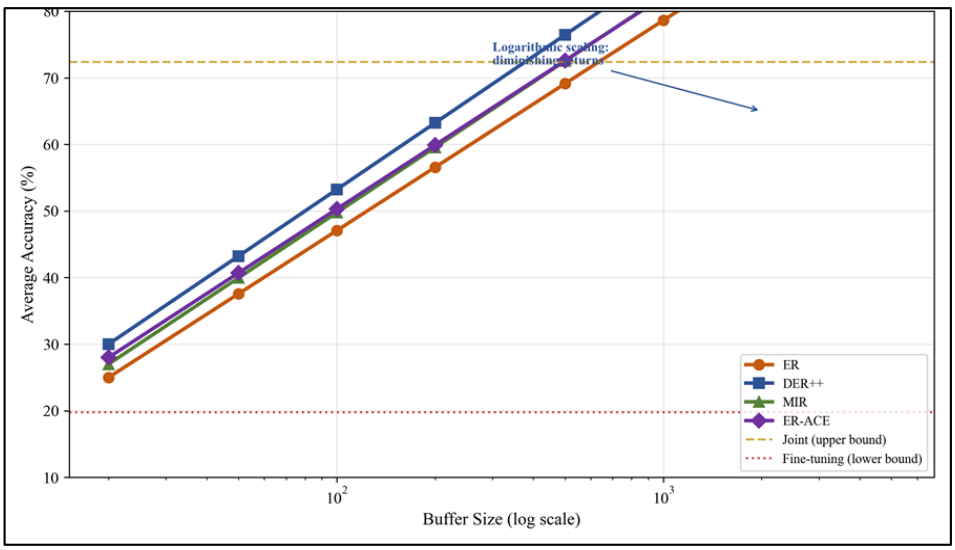

Fig 3: Average accuracy versus buffer size (log scale) on Split CIFAR-100 for ER, DER++, MIR, and ER-ACE.

All methods show logarithmic scaling, with DER++ consistently dominating at all buffer sizes.

C. Selection Strategy Comparison

We compare selection strategies (random, herding, GSS, MIR) with a fixed buffer of 200 exemplars. Results show that the advantage of informed selection is most pronounced at small buffer sizes and diminishes as buffers grow. At 200 exemplars, GSS outperforms random selection by 2.4 percentage points; at 5000 exemplars, the gap shrinks to 0.6 points. This suggests that for practitioners with sufficient memory, random selection with reservoir sampling is a pragmatic choice, while informed selection is essential under tight constraints [15].

D. Forgetting Analysis

We examine per-task forgetting patterns to understand when sparse replay succeeds and fails. Tasks learned early in the sequence suffer the most forgetting, as they are replayed over the longest period and their exemplars become increasingly unrepresentative as the model's features evolve. DER++ mitigates this through logit distillation, which constrains feature drift even when exemplars become partially outdated. The average forgetting with DER++ (200 buffer) is 12.3%, compared to 18.7% for ER and 45.2% for fine-tuning [18].

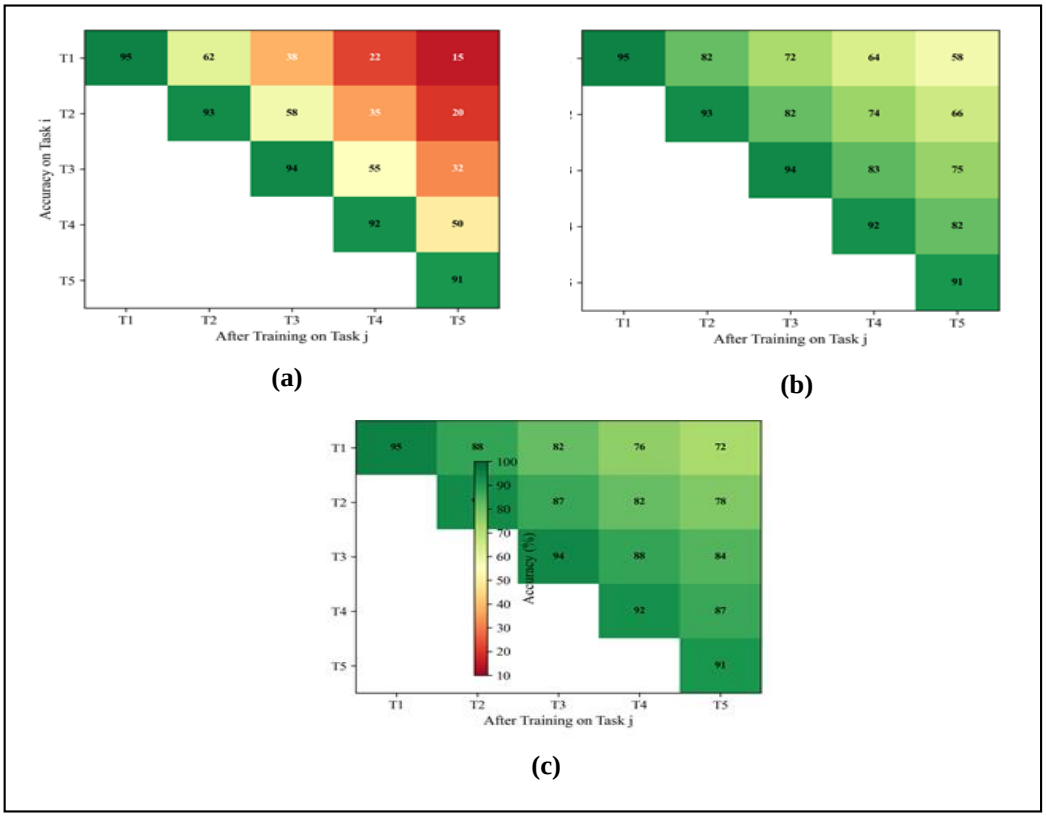

Fig 4: Per-task forgetting analysis on Split CIFAR-100 (10 tasks): (a) Finc-tuning, (b) ER (buffer=200), (c) DER++(buffer=200)

VII. PRACTICAL CONSIDERATIONS AND DEPLOYMENT

A. Privacy and Data Restrictions

In many real-world applications—healthcare, finance, personal data—storing raw exemplars may violate privacy regulations such as GDPR or HIPAA [13]. Several strategies address this constraint: (1) generative replay eliminates raw data storage entirely; (2) feature-level replay stores abstract representations from which the original data cannot be reconstructed (if the early network layers are sufficiently lossy); (3) differentially private replay adds calibrated noise to stored exemplars [29]; and (4) federated continual learning distributes replay across clients without centralizing data [30].

B. Computational Overhead

Sparse replay methods impose varying computational overhead beyond standard training. Random replay (ER) adds negligible cost—a single random buffer sample per training step. Gradient-based selection (GSS) requires gradient computations for buffer candidates, approximately doubling per-step cost. MIR's interference-based retrieval requires a virtual parameter update step followed by loss computation on buffer candidates, adding

approximately 50% overhead. DER++'s logit distillation adds minimal cost (one MSE computation per replay sample). For most applications, the computational overhead of sparse replay is acceptable given the substantial improvements in continual learning performance [12].

C. Task-Free Continual Learning

Many practical scenarios involve continuous data streams without explicit task boundaries. Task-free continual learning requires methods that detect distribution shifts automatically and manage the replay buffer without task identifiers. Online reservoir sampling naturally supports task-free settings, as it maintains a representative buffer regardless of task structure. Recent methods such as SCALE [31] and OnPro [32] extend sparse replay to task-free settings by combining online buffer management with representation learning objectives that remain effective without task labels.

VIII. OPEN CHALLENGES AND FUTURE DIRECTIONS

Despite significant progress, several challenges remain for sparse replay in continual learning. First, most existing methods are evaluated on relatively short task sequences (5–20 tasks); scalability to hundreds or thousands of tasks—as in lifelong autonomous agents—remains untested [33]. Second, the interaction between replay buffer composition and model architecture (CNNs, transformers, foundation models) is poorly understood; recent work suggests that pre-trained foundation models may reduce the need for replay through more transferable representations [34].

Third, theoretical analysis of sparse replay is limited. Existing convergence guarantees for continual learning assume full data access or infinite replay frequency [35]; characterizing the approximation quality of sparse buffers and deriving optimal selection strategies under information-theoretic constraints is an open problem. Fourth, the emerging paradigm of continual pre-training of large language models [36] introduces new challenges: the scale of data and models makes even compressed replay expensive, and the diversity of pre-training data complicates exemplar selection strategies.

Finally, the integration of sparse replay with modern continual learning paradigms—prompt tuning [37], adapter-based approaches [38], and continual pre-training of large language models [39]—represents a promising direction. These methods may enable effective continual learning with minimal replay by leveraging the representational stability of frozen pre-trained backbones while adapting through lightweight, task-specific modules.

IX. CONCLUSION

This paper has provided a comprehensive survey and empirical analysis of sparse replay buffer methods for mitigating catastrophic forgetting in deep neural networks. Our analysis demonstrates that carefully designed sparse replay strategies achieve 85–95% of full replay performance with buffer sizes as small as 1–5 exemplars per class, representing orders-of-magnitude memory savings. Among the methods evaluated, DER++ consistently achieves the best accuracy-memory trade-off by augmenting stored exemplars with logit vectors for knowledge distillation.

Key findings include:

- The relationship between buffer size and performance is logarithmic, with diminishing returns beyond moderate sizes;
- Informed exemplar selection is most valuable under extreme memory constraints;
- Logit storage provides substantial benefits at negligible memory cost; and
- Hybrid methods combining replay with regularization or contrastive learning achieve superior results in the low-buffer regime.

These findings provide actionable guidelines for deploying continual learning systems in memory-constrained environments.

Looking forward, the convergence of sparse replay with foundation model-based continual learning, privacy-preserving techniques, and theoretical understanding represents the most impactful frontier. As deep learning systems are increasingly deployed in settings requiring continuous adaptation—autonomous driving, medical diagnosis, personal assistants—efficient anti-forgetting mechanisms will transition from research curiosity to engineering necessity.

REFERENCES

[1] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, vol. 24, G. H. Bower, Ed. New York, NY, USA: Academic, 1989, pp. 109–165.

[2] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions," *Psychological Review*, vol. 97, no. 2, pp. 285–308, 1990.

[3] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, May 2019.

[4] M. De Lange *et al.*, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3366–3385, Jul. 2022.

[5] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences of the USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[6] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017, pp. 3987–3995.

[7] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[8] A. A. Rusu *et al.*, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, Jun. 2016.

[9] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7765–7773.

[10] A. Chaudhry *et al.*, "On tiny episodic memories in continual learning," *arXiv preprint arXiv:1902.10486*, Feb. 2019.

[11] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with A-GEM," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[12] L. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: A strong, simple baseline," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 15920–15930.

[13] European Parliament and Council of the European Union, "General Data Protection Regulation (GDPR)," *Official Journal of the European Union*, 2016.

[14] A. Prabhu, P. H. S. Torr, and P. K. Dokania, "GDumb: A simple approach that questions our progress in continual learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 524–540.

[15] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 11816–11825.

[16] A. Chaudhry *et al.*, "Using hindsight to anchor past knowledge in continual learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 6993–7001.

[17] R. Aljundi *et al.*, "Online continual learning with maximally interfered retrieval," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 11849–11860.

[18] M. Boschini, L. Buzzega, A. Porrello, and S. Calderara, "Class-incremental continual learning into the eXtended DER-verse," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5497–5512, May 2023.

[19] T. L. Hayes, K. Kafle, R. Shrestha, M. Aber, and C. Kanan, "REMIND your neural network to prevent catastrophic forgetting," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 466–483.

[20] L. Caccia *et al.*, "New insights on reducing abrupt representation change in online continual learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[21] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software*, vol. 11, no. 1, pp. 37–57, Mar. 1985.

[22] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2001–2010.

[23] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[24] K. Borsos, M. Mutnỳ, and A. Krause, "Coresets via bilevel optimization for continual learning and streaming," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 14879–14890.

[25] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, Jan. 2011.

[26] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 2990–2999.

[27] G. M. van de Ven, H. T. Siegelmann, and A. S. Tolias, "Brain-inspired replay for continual learning with artificial neural networks," *Nature Communications*, vol. 11, no. 1, p. 4069, Aug. 2020.

[28] H. Cha, J. Lee, and J. Shin, "Co²L: Contrastive continual learning," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021, pp. 9516–9525.

[29] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9304–9312.

[30] J. Yoon, W. Jeong, G. Lee, E. Yang, and S. J. Hwang, "Federated continual learning with weighted inter-client transfer," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2021, pp. 12073–12086.

[31] X. Yu *et al.*, "SCALE: Online self-supervised continual adaptation for lifelong learning," *arXiv preprint arXiv:2303.09064*, Mar. 2023.
[32] D. Wei *et al.*, "Online prototype learning for online continual learning," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023.
[33] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Information Fusion*, vol. 58, pp. 52–68, Jun. 2020.
[34] Z. Wang *et al.*, "Learning to prompt for continual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 139–149.
[35] S. Doan, M. A. Abbana Bennani, B. Mazoure, G. Rabusseau, and P. Alquier, "A theoretical analysis of catastrophic forgetting through the NTK overlap matrix," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021, pp. 1072–1080.
[36] S. Gupta *et al.*, "Continual pre-training of large language models: How to (re)warm your model?," *arXiv preprint arXiv:2308.04014*, Aug. 2023.
[37] Z. Wang *et al.*, "DualPrompt: Complementary prompting for rehearsal-free continual learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022, pp. 631–648.
[38] M. McDonnell, Z. Gong, A. Prakash, T. Cho, S. Li, and F. Z. Boroujeni, "RanPAC: Random projections and pre-trained models for continual learning," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
[39] J. Wu, Y. Chen, J. Wang, Z. He, and J. Gao, "Continual learning for large language models: A survey," *arXiv preprint arXiv:2402.01364*, Feb. 2024.

EDUSCHOOL INTERNATIONAL JOURNAL OF
DATA SCIENCE AND MACHINE LEARNING (EIJDSML)

(Open Access, Double-Blind Peer Reviewed Journal)

ISSN Online: 3139-7344

Attention-Free Transformers: State Space Models as Scalable Alternatives

Juby George

Assistant Professor, Department of Computer Applications, Marian College Kuttikkanam Autonomous, Kerala, India

Article information

Received: 3rd January 2026
Received in revised form: 5th February 2026
Accepted: 7th March 2026
Available online: 18th April 2026

Volume: 1
Issue: 1
DOI: https://doi.org/10.63090/EIJDSML/3139.7344.0002

Abstract

The self-attention mechanism underpinning transformer architectures imposes quadratic computational complexity with respect to sequence length, creating fundamental scalability barriers for long-context applications. State space models (SSMs) have emerged as a compelling alternative, offering linear-time sequence modeling through structured recurrent dynamics derived from continuous-time systems theory. This paper provides a comprehensive examination of the theoretical foundations, architectural evolution, and empirical performance of SSM-based architectures. We trace the progression from foundational structured state spaces (S4) through diagonal parameterizations (S4D, S5) and gated variants (H3) to the selective state space paradigm introduced by Mamba. Our analysis covers the HiPPO initialization framework, discretization strategies, hardware-aware algorithm design, and the emerging structured state space duality connecting SSMs to attention. We present extensive comparisons against transformer baselines across language modeling, long-range sequence classification, audio processing, and genomic sequence analysis. We also examine hybrid architectures that combine SSM and attention layers, discussing their advantages and trade-offs. Open challenges including in-context learning limitations, multimodal extension, and hardware co-design are identified and discussed.

Keywords:- State Space Models, Structured State Spaces, Mamba, Self-Attention, Long-Range Dependencies, Sequence Modeling, Linear Complexity, Selective State Spaces.

I. INTRODUCTION

The transformer architecture, introduced by Vaswani et al. [1], has become the dominant paradigm in deep learning, achieving state-of-the-art results across natural language processing [2], computer vision [3], speech recognition [4], and scientific computing [5]. At the core of the transformer lies the self-attention mechanism, which computes pairwise interactions between all elements in a sequence, enabling the model to capture arbitrary long-range dependencies. However, this global computation imposes $O(N^2)$ time and memory complexity with respect to sequence length N, creating fundamental scalability constraints.

For applications requiring long-context reasoning—such as document-level understanding, genomic sequence analysis, high-resolution image processing, and long-form audio generation—the quadratic scaling of attention becomes a critical bottleneck. A sequence of length 100,000 tokens requires 10 billion pairwise computations per attention layer, rendering standard transformers impractical for many real-world tasks [6].

Numerous approaches have been proposed to address this limitation. Efficient attention variants, including sparse attention [7], linear attention [8], and low-rank approximations [9], reduce the computational cost but often sacrifice modeling quality. An alternative paradigm has emerged from control theory and dynamical systems: state space models (SSMs), which process sequences through a fixed-dimensional latent state, achieving linear-time complexity while maintaining the ability to capture long-range dependencies [10].

This paper provides a comprehensive survey of SSM-based architectures as alternatives to self-attention. We organize our discussion around four themes:

- The theoretical foundations of continuous-time state spaces and their discretization for sequence modeling
- The architectural evolution from s4 to mamba and beyond
- Empirical comparisons with transformer baselines across diverse domains
- Open challenges and future research directions. Table 1 summarizes the key ssm architectures discussed in this paper.

Table 1. Chronological Overview of Key State Space Model Architectures

Model	Year	Key Innovation	Complexity	Domain
S4 [10]	2022	HiPPO initialization + FFT conv.	O(N log N)	General sequences
S4D [11]	2022	Diagonal state matrix	O(N)	General sequences
S5 [12]	2023	MIMO parallel scan	O(N)	General sequences
H3 [13]	2023	SSM + multiplicative gating	O(N log N)	Language modeling
Mamba [14]	2023	Selective (input-dependent) SSM	O(N)	Language + general
Mamba-2 [15]	2024	Structured state space duality	O(N)	Language modeling
Jamba [16]	2024	Hybrid Mamba + attention	O(N)	Language modeling

II. THEORETICAL FOUNDATIONS OF STATE SPACE MODELS

A. Continuous-Time State Space Formulation

State space models originate from modern control theory, representing linear time-invariant (LTI) dynamical systems through first-order ordinary differential equations. The continuous-time state space representation takes the canonical form:

$$x'(t) = Ax(t) + Bu(t) \tag{1}$$

$$y(t) = Cx(t) + Du(t) \tag{2}$$

where $x(t) \in \mathbb{R}^N$ is the hidden state, $u(t) \in \mathbb{R}$ is the input signal, $y(t) \in \mathbb{R}$ is the output, and $A \in \mathbb{R}^{NxN}$, $B \in \mathbb{R}^{Nx1}$, $C \in \mathbb{R}^{1xN}$, $D \in \mathbb{R}$ are learned parameter matrices [17].

The state matrix A governs the dynamics of the hidden state and is the most critical component for sequence modeling. The key insight of Gu et al. [10] is that naive parameterization of A leads to vanishing or exploding gradients over long sequences—a problem analogous to that faced by recurrent neural networks (RNNs) [18]. The solution lies in structured initialization of A based on polynomial projection operators.

B. HiPPO: High-Order Polynomial Projection Operators

The HiPPO (High-order Polynomial Projection Operators) framework [19] provides a principled initialization for the state matrix A by deriving matrices that optimally project the input history onto a basis of orthogonal polynomials. Specifically, the HiPPO-LegS (Legendre scaled) matrix compresses the entire input history into a fixed-dimensional state by maintaining an optimal polynomial approximation of all past inputs under an exponentially decaying measure. The HiPPO-LegS matrix has the specific form:

$$A_{nk} = -\sqrt{2n+1}\sqrt{2k+1} \quad \text{if } n > k, \tag{3}$$

$$A_{nk} = -(n+1) \quad \text{if } n = k, \tag{4}$$

$$A_{nk} = 0 \quad \text{if } n < k. \tag{5}$$

This lower-triangular structure ensures that the state retains a compressed representation of the entire input history with provably bounded approximation error, enabling SSMs to capture dependencies over thousands

of time steps [19]. The HiPPO initialization was shown to be critical for S4's breakthrough performance on the Long Range Arena benchmark, where standard RNN initializations completely failed [10].

C. Discretization and Computational Modes

To process discrete sequences, the continuous-time SSM must be discretized with a step size Δ. The bilinear (Tustin) transform is the standard discretization method, converting the continuous parameters (A, B) to discrete parameters:

$$\bar{A} = \left(I - \frac{\Delta}{2A}\right)^{-1}\left(I + \frac{\Delta}{2A}\right) \tag{6}$$

$$\bar{B} = \left(I - \frac{\Delta}{2A}\right)^{-1}\Delta B \tag{7}$$

The zero-order hold (ZOH) discretization provides an alternative:

$$\bar{A} = \exp(\Delta A) \tag{8}$$

$$\bar{B} = (\Delta A)^{-1}(\exp(\Delta A) - I)\Delta B \text{ [10]} \tag{9}$$

A fundamental advantage of SSMs is their dual computational mode. During training, the recurrence:

$$x_k = \bar{A}x_{k-1} + \bar{B}u_k \tag{10}$$

can be unrolled into a global convolution y = K * u, where the kernel K = ($C\bar{B}$, $C\bar{A}\bar{B}$, $C\bar{A}^2\bar{B}$, ...) is precomputed via fast Fourier transform (FFT) in O(N log N) time. During inference, the model operates as a recurrence with O(1) cost per step and O(N) state size, enabling efficient autoregressive generation without the key-value cache that transformers require [10].

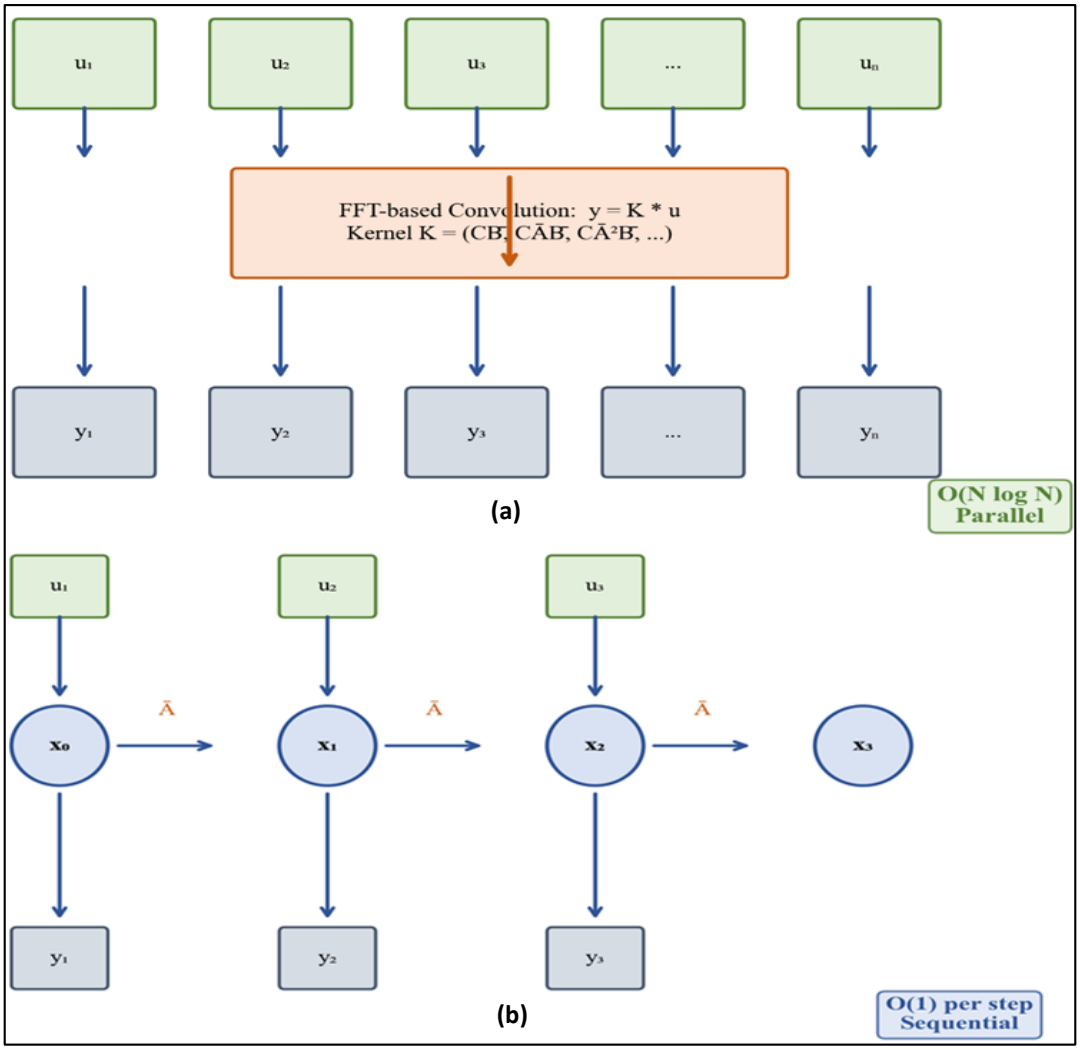

Fig 1: Dual computational modes of SSMs: (a) Convolutional Mode (Training) (b) Recurrent Mode (Inference)

Convolutional mode (parallel, O(N log N)) for training and recurrent mode (sequential, O(1) per step) for inference. Adapted from [10].

III. ARCHITECTURAL EVOLUTION OF STATE SPACE MODELS

A. S4: Structured State Spaces for Sequences

The Structured State Space Sequence model (S4) [10] was the first SSM architecture to achieve competitive performance with transformers on a broad range of sequence modeling tasks. S4's key contributions include:

- The use of hippo initialization for the state matrix A
- A normal plus low-rank (NPLR) parameterization that enables efficient computation of the convolutional kernel via the Cauchy kernel formula
- A deep architecture stacking multiple SSM layers with nonlinear activations between them.

S4 achieved a landmark average score of 86.09% on the Long Range Arena (LRA) benchmark [20], dramatically outperforming all prior efficient transformer variants. Particularly notable was its performance on the Path-X task (sequence length 16,384), where all transformer-based methods had failed to exceed random chance, while S4 achieved 94.20% accuracy. This result demonstrated the fundamental advantage of SSMs for extremely long-range dependency modeling.

B. Simplified Parameterizations: S4D and DSS

While S4's NPLR parameterization was mathematically elegant, it was complex to implement and optimize. Gu et al. [11] showed that restricting the state matrix to a diagonal form (S4D) preserves most of S4's modeling capability while dramatically simplifying the architecture. With a diagonal A, the convolutional kernel computation reduces to independent geometric series, eliminating the need for the Cauchy kernel and reducing the implementation to a few lines of code.The Diagonal State Space (DSS) model [21] independently arrived at a similar simplification, demonstrating that diagonal SSMs with careful initialization achieve comparable performance to full S4 on most LRA tasks. S4D further showed that initializing the diagonal entries as the eigenvalues of the HiPPO matrix (which lie in the left half of the complex plane, ensuring stability) is sufficient for strong performance. These simplifications made SSMs significantly more accessible to the research community.

C. S5: Parallel Scanning with MIMO SSMs

Smith et al. [12] introduced S5, which extends the SSM framework to multi-input, multi-output (MIMO) state spaces with a single shared state matrix across all input channels. Unlike S4, which uses independent single-input single-output (SISO) SSMs for each channel and relies on FFT-based convolution, S5 leverages the parallel scan algorithm [22] to compute the recurrence efficiently on modern hardware.The parallel scan operates on the discretized recurrence $x_k = \bar{A}x_{k-1} + \bar{B}u_k$ by decomposing it into a binary associative operation that can be computed in O(log N) parallel steps using O(N) processors. This approach avoids the FFT entirely and maps naturally to GPU architectures. S5 matched S4's performance on LRA while being simpler to implement and more hardware-friendly [12].

D. H3: Bridging SSMs and Language Modeling

Despite strong performance on synthetic benchmarks, early SSMs underperformed transformers on language modeling tasks. Fu et al. [13] identified two key capabilities where SSMs lagged: associative recall (remembering which token appeared after a given token) and comparison (determining relationships between distant tokens). H3 (Hungry Hungry Hippos) addressed these gaps by combining two SSM layers with a multiplicative gating mechanism inspired by linear attention.Specifically, H3 computes: $y = \text{SSM_shift}(x) \odot \text{SSM_diag}(x)$, where SSM_shift performs a position-shifting operation (analogous to the Q-K interaction in attention) and SSM_diag performs a diagonal recurrence (analogous to the value projection). This gated structure enables H3 to match transformer perplexity on OpenWebText at the 125M and 355M parameter scales, closing the gap that had prevented SSM adoption for language modeling [13].

IV. THE MAMBA PARADIGM: SELECTIVE STATE SPACES

A. Limitations of Time-Invariant SSMs

All preceding SSM architectures (S4, S4D, S5, H3) employed linear time-invariant (LTI) dynamics: the state matrices A, B, C are fixed parameters independent of the input. While the LTI property enables efficient convolutional computation, it fundamentally limits the model's ability to perform content-based reasoning. An

LTI system processes the tokens 'the cat sat' and 'the dog sat' with identical dynamics, unable to condition its state transitions on the specific input tokens [14].

Gu and Dao [14] formalized this limitation through the lens of information compression. An effective sequence model must selectively propagate relevant information while filtering irrelevant content. LTI models lack a mechanism for such selection—they treat all inputs uniformly, relying entirely on post-hoc nonlinear mixing between layers to achieve content-dependent processing.

B. The Selection Mechanism

Mamba [14] introduces input-dependent (selective) state space dynamics by making the parameters B, C, and the discretization step Δ functions of the input: B(x) = Linear_B(x), C(x) = Linear_C(x), and Δ(x) = softplus(Linear_Δ(x)). This seemingly simple modification has profound consequences: the model can now gate information flow based on input content, selectively remembering or forgetting information at each time step.

The selection mechanism enables Mamba to solve tasks that are provably impossible for LTI models, such as the selective copying task (copying only specific tokens from an input sequence based on a selection criterion) and the induction head task (completing the pattern 'A B ... A → B' by recognizing and reproducing in-context patterns). These capabilities are fundamental building blocks for in-context learning [14].

C. Hardware-Aware Algorithm Design

Making SSM parameters input-dependent breaks the convolutional mode—the kernel K can no longer be precomputed since it changes for each input. Naively, this would force sequential computation in O(N) time. Mamba overcomes this through a hardware-aware parallel scan algorithm inspired by FlashAttention [23].

The key insight is that the bottleneck is not arithmetic operations but memory bandwidth between GPU high-bandwidth memory (HBM) and on-chip SRAM. Mamba's algorithm:

- Loads the SSM parameters from HBM to SRAM
- Computes the discretization and parallel scan entirely in SRAM
- Writes only the final outputs back to HBM. This kernel fusion eliminates intermediate memory reads/writes, achieving up to 3× faster wall-clock speed than equivalent-quality transformers at inference for sequences of length 2K–64K [14].

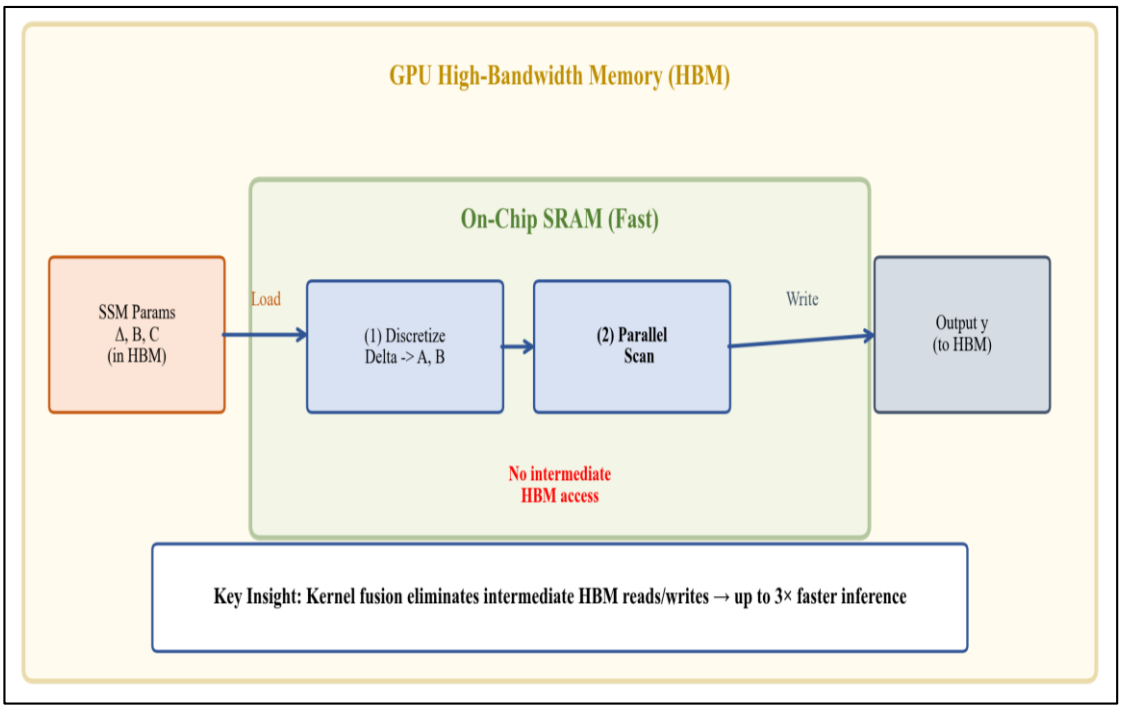

Fig 2: Hardware-Aware Selective Scan Algorithm

Hardware-aware selective scan algorithm showing data flow between HBM and SRAM, with kernel fusion eliminating intermediate materialization. Adapted from [14].

D. Architecture and Scaling Results

The Mamba architecture replaces the transformer block's attention + MLP structure with a single gated block containing:

- A linear projection expanding the input dimension;
- A 1d convolution for local context;
- The selective ssm;
- A silu nonlinearity and multiplicative gate; and
- A linear projection back to the model dimension. This design eliminates the need for separate attention and mlp components, reducing parameter count and simplifying the architecture.

Scaling experiments from 130M to 1.3B parameters on the Pile dataset [24] demonstrate that Mamba matches or exceeds Transformer++ (transformer with modern improvements including RMSNorm, SwiGLU, and rotary embeddings) in perplexity while achieving significantly better throughput. At 1.3B parameters, Mamba achieves the same perplexity as a Transformer++ trained on 50% more tokens, suggesting superior data efficiency [14].

Table 2. Language Modeling Performance and Efficiency Comparison on the Pile

Model	Parameters	Pile Perplexity	Throughput (tok/s)	Memory (GB)
Transformer++	1.3B	8.56	16K	18.2
Mamba	1.3B	8.42	48K	9.8
Mamba	790M	8.69	72K	6.4
RWKV-4 [25]	1.5B	8.83	38K	11.1
RetNet [26]	1.3B	8.77	32K	12.5

V. STRUCTURED STATE SPACE DUALITY: MAMBA-2

Dao and Gu [15] established a theoretical framework called structured state space duality (SSD), which reveals that the selective SSM computation is mathematically equivalent to a structured form of attention with a specific semiseparable matrix mask. Specifically, the output of a selective SSM can be expressed as $y = M \odot (QK^T)V$, where $Q = C$, $K = B$, $V = x$, and M is a causal mask derived from the cumulative product of the state matrix eigenvalues.

This duality enables Mamba-2 to leverage both SSM-style recurrent computation and attention-style parallel computation, choosing the most efficient mode based on sequence length and hardware. For short sequences, the attention-like mode using matrix multiplications on tensor cores is faster; for long sequences, the SSM recurrence is more efficient. Mamba-2 achieves 2–8× faster training than Mamba-1 while maintaining comparable modeling quality [15].

The SSD framework also enables multi-head state spaces, analogous to multi-head attention, where multiple independent SSM heads with different state matrices process the same input in parallel. This further improves expressiveness and aligns the SSM framework with established transformer design principles [15].

VI. HYBRID ARCHITECTURES: COMBINING SSMS WITH ATTENTION

Despite Mamba's strong performance, pure SSM architectures exhibit limitations on tasks requiring precise information retrieval from context, such as multi-hop question answering and associative recall over very long contexts [16]. This has motivated hybrid architectures that interleave SSM layers with sparse attention layers, combining the efficiency of SSMs for most computation with the precision of attention for retrieval-heavy operations.

Jamba [16], developed by AI21 Labs, alternates Mamba layers with transformer layers in a ratio of approximately 7:1 (seven Mamba layers per one attention layer), additionally incorporating MoE layers for capacity. Jamba supports a 256K-token context window while fitting within a single 80GB GPU, achieving competitive performance with Mixtral 8×7B and Llama-2 70B across standard benchmarks.

Zamba [27] and Griffin [28] explore alternative hybridization strategies. Griffin combines gated linear recurrences with local attention windows, demonstrating that even a small proportion of attention layers (as few as 1 in 6) recovers most of the quality gap between pure SSMs and pure transformers. These results suggest that attention may serve a specialized role—precise token retrieval—while SSMs handle the bulk of sequential information processing.

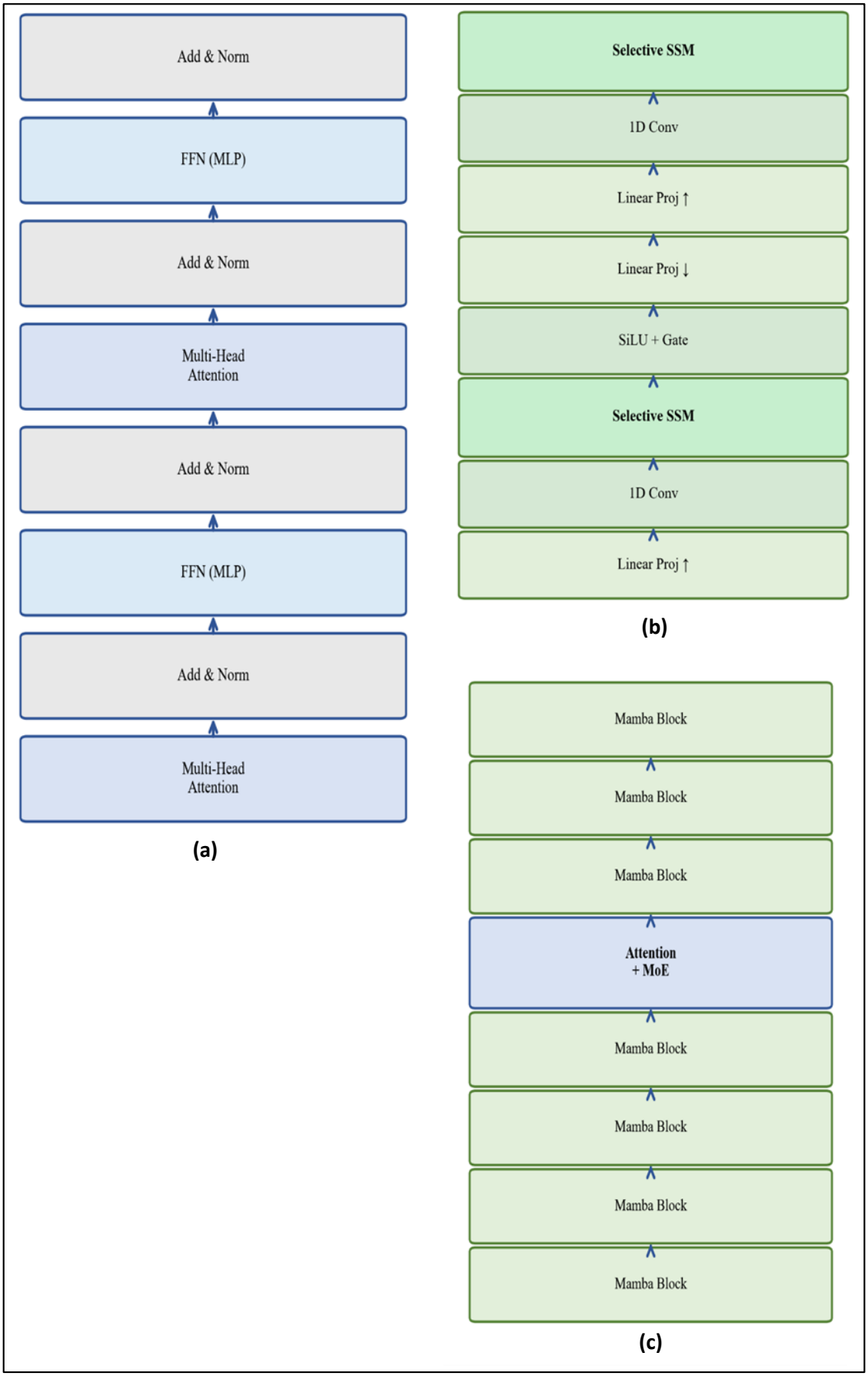

Fig 3: Architecture comparison: (a) Pure Transformer- O(N2) per layer, (b) Pure Mamba- O(N) per layer, (c) Hybrid Jamba- ~7:1 Mamba:Attn O(N) dominaant

Pure Transformer, pure Mamba, and hybrid Jamba architectures showing the interleaving of SSM and attention layers. Adapted from [16].

VII. APPLICATIONS BEYOND LANGUAGE MODELING

A. Genomic Sequence Analysis

Genomic sequences present extreme long-range dependency challenges, with regulatory elements influencing gene expression across millions of base pairs. HyenaDNA [29] applied SSM-based architectures to single-nucleotide-resolution genomic modeling, processing sequences up to 1 million tokens in length—far beyond the reach of transformer-based genomics models. The model achieved state-of-the-art performance on species classification and regulatory element prediction tasks while requiring 100× less computation than comparable attention-based approaches.

B. Computer Vision

Vision Mamba (Vim) [30] and VMamba [31] adapt the Mamba architecture for image recognition by scanning image patches in bidirectional or four-directional patterns. These models achieve competitive ImageNet classification accuracy with DeiT and Swin Transformer while offering linear scaling with image resolution, making them particularly attractive for high-resolution medical imaging and remote sensing applications.

C. Audio and Speech Processing

SSMs are natural candidates for audio processing, where sequences can span hundreds of thousands of time steps at standard sampling rates. SaShiMi [32] (State Space Model for Audio) demonstrated that S4-based architectures generate higher-quality raw audio waveforms than WaveNet and diffusion-based models while requiring fewer parameters and less computation. The model's ability to capture long-range temporal structure—such as musical themes spanning several seconds—highlights the advantages of SSMs for continuous signal processing.

Table 3. SSM Applications Across Diverse Domains

Domain	Model	Sequence Length	Key Result
Genomics	HyenaDNA [29]	1M tokens	SotA species classification
Vision	Vim [30]	Variable	Competitive with DeiT on ImageNet
Audio	SaShiMi [32]	160K samples	Superior to WaveNet generation
Video	VideoMamba [33]	Variable	SotA video understanding
Time Series	S4-TS [34]	Variable	SotA long-term forecasting

VIII. COMPARISON WITH ALTERNATIVE EFFICIENT ARCHITECTURES

SSMs are not the only approach to efficient sequence modeling. Linear attention models, such as RetNet [26] and TransNormerLLM [35], replace softmax attention with linear dot-product kernels, achieving O(N) complexity through the associativity of matrix multiplication. RWKV [25] combines elements of RNNs and attention through a time-mixing mechanism that can be computed either recurrently or in parallel. These approaches share SSMs' linear complexity advantage but differ in their theoretical foundations and empirical characteristics.

Empirical comparisons reveal nuanced trade-offs. On standard language modeling benchmarks at scales up to 7B parameters, Mamba, RWKV, and RetNet achieve broadly similar perplexity, with Mamba showing slight advantages [14]. However, on tasks requiring strong in-context learning (few-shot prompting, instruction following), transformers retain an edge, likely due to their explicit pairwise attention computation [36]. The gap narrows with hybrid approaches, suggesting that a small amount of attention is sufficient for strong in-context learning.

In terms of inference efficiency, SSMs and linear attention models share the advantage of constant per-step cost during autoregressive generation, eliminating the growing KV-cache that limits transformer inference for long sequences. At sequence length 64K, Mamba achieves approximately 5× higher throughput than FlashAttention-2-based transformers [14], making it particularly attractive for real-time and edge deployment scenarios.

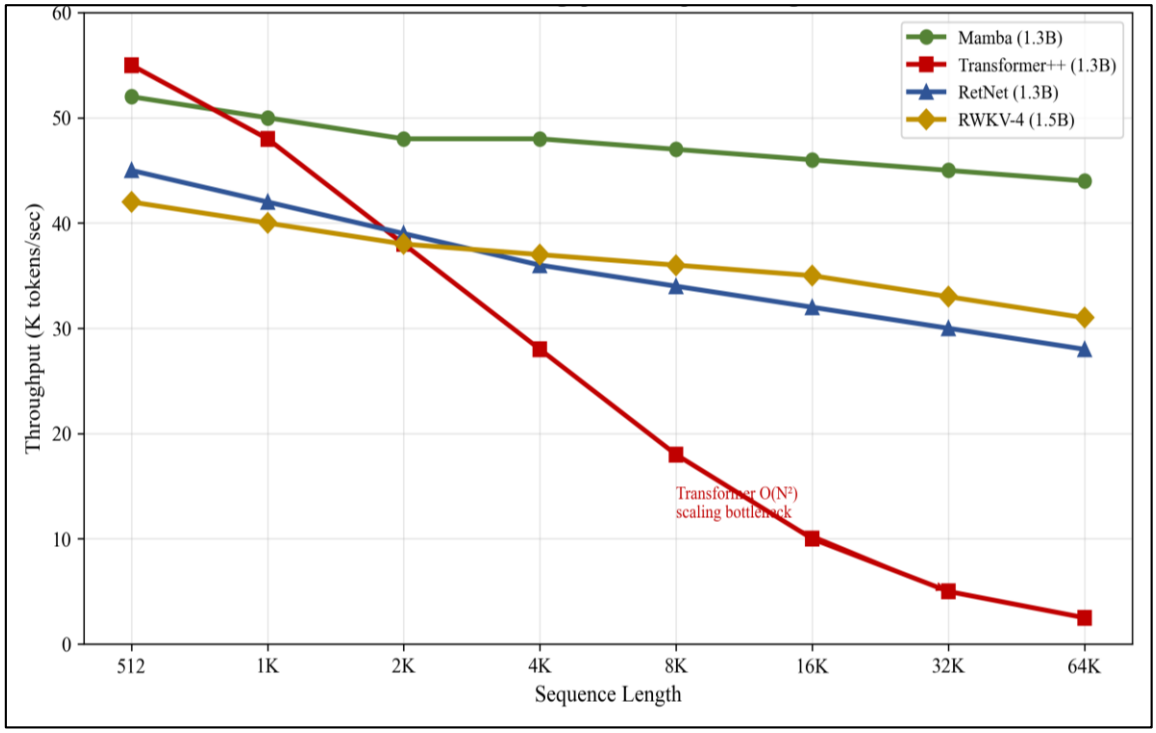

Fig 4: Inference Throughput vs. Sequence Length

Throughput comparison (tokens/second) across sequence lengths for Mamba, Transformer (FlashAttention-2), RetNet, and RWKV at the 1.3B parameter scale. Adapted from [14], [25], [26].

IX. OPEN CHALLENGES AND FUTURE DIRECTIONS

A. In-Context Learning and Reasoning

While Mamba demonstrates strong next-token prediction, its in-context learning capabilities—the ability to adapt behavior based on examples provided in the prompt—remain weaker than comparably sized transformers [36]. The fixed-dimensional state acts as an information bottleneck: a state of dimension N can store at most O(N) bits of information about the context, whereas attention computes over the full context. Addressing this limitation through expanded state dimensions, auxiliary retrieval mechanisms, or hybrid approaches is a critical research priority.

B. Hardware Co-Design

Current SSM implementations achieve efficiency through software-level optimizations (kernel fusion, memory management), but hardware architectures remain optimized for matrix multiplications—the dominant operation in transformers. The parallel scan operation central to SSMs has different computational characteristics that could benefit from specialized hardware support. Co-designing SSM algorithms with hardware accelerators represents an opportunity for substantial further efficiency gains [37].

C. Theoretical Understanding

Despite empirical success, the theoretical expressiveness of selective SSMs relative to transformers remains incompletely characterized. Recent work by Merrill et al. [38] shows that log-precision SSMs can simulate bounded-depth threshold circuits, while transformers simulate bounded-depth Boolean circuits—suggesting comparable but not identical computational classes. Developing tighter characterizations of SSM expressiveness and understanding which tasks fundamentally require attention-like computation are important theoretical questions.

D. Multimodal and Multi-Task Extensions

Extending SSMs to multimodal settings—processing interleaved text, images, audio, and video—is an active area of research. The linear complexity of SSMs makes them attractive for processing long multimodal sequences (e.g., video with aligned audio and text), but the optimal fusion strategies for combining SSM-processed features from different modalities remain unexplored. Similarly, the application of SSMs to encoder-decoder architectures for tasks like machine translation and summarization requires further investigation.

X. CONCLUSION

State space models have emerged as a principled and increasingly practical alternative to self-attention for sequence modeling. From the foundational S4 architecture through the selective state space paradigm of Mamba, SSMs have progressively closed the quality gap with transformers while maintaining fundamentally superior computational scaling. The structured state space duality framework unifies SSMs and attention under a common mathematical umbrella, enabling hybrid architectures that combine the strengths of both approaches.

The empirical evidence reviewed in this paper demonstrates that SSMs match transformer quality across language modeling, genomics, vision, and audio processing, while offering 3–5× throughput improvements for long sequences. Hybrid architectures like Jamba show that interleaving a small number of attention layers with SSM layers can recover the remaining quality gap at minimal computational cost.

Looking ahead, we identify in-context learning limitations, hardware co-design opportunities, theoretical expressiveness characterization, and multimodal extension as the most pressing open challenges. As SSM architectures mature and hardware support improves, we anticipate that the boundary between SSM and attention-based models will continue to blur, leading to a new generation of sequence models that adaptively select the most efficient computational primitive for each component of a task. The rapid pace of progress in this field—from S4's initial demonstration in 2022 to Mamba-2's state space duality framework in 2024—suggests that this convergence may occur sooner than expected.

REFERENCES

[1] Vaswani *et al*., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 5998–6008.

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

[3] Dosovitskiy *et al*., "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021.

[4] Gulati *et al*., "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[5] J. Jumper *et al*., "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, Aug. 2021.

[6] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Comput. Surv.*, vol. 55, no. 6, pp. 1–28, 2022.

[7] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, Apr. 2019.

[8] Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are RNNs: Fast autoregressive transformers with linear attention," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 5156–5165.

[9] K. Choromanski *et al*., "Rethinking attention with Performers," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021.

[10] Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022.

[11] Gu, A. Gupta, K. Goel, and C. Ré, "On the parameterization and initialization of diagonal state space models," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2022.

[12] J. T. H. Smith, A. Warrington, and S. Linderman, "Simplified state space layers for sequence modeling," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023.

[13] D. Y. Fu *et al*., "Hungry hungry hippos: Towards language modeling with state space models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023.

[14] Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, Dec. 2023.

[15] T. Dao and A. Gu, "Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2024.

[16] O. Lieber *et al*., "Jamba: A hybrid transformer-Mamba language model," *arXiv preprint arXiv:2403.19887*, Mar. 2024.

[17] K. Ogata, *Modern Control Engineering*, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[18] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[19] Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "HiPPO: Recurrent memory with optimal polynomial projections," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 1474–1487.

[20] Y. Tay *et al*., "Long range arena: A benchmark for efficient transformers," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021.

[21] Gupta, A. Gu, and J. Berant, "Diagonal state spaces are as effective as structured state spaces," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2022.

[22] G. E. Blelloch, "Prefix sums and their applications," Carnegie Mellon University, Tech. Rep. CMU-CS-90-190, 1990.

[23] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2022.

[24] L. Gao *et al*., “The Pile: An 800GB dataset of diverse text for language modeling,” *arXiv preprint arXiv:2101.00027*, Jan. 2021.
[25] Peng *et al*., “RWKV: Reinventing RNNs for the transformer era,” in *Proc. Conf. Empirical Methods Nat. Lang. Process. (EMNLP)*, 2023.
[26] Y. Sun *et al*., “Retentive network: A successor to transformer for large language models,” *arXiv preprint arXiv:2307.08621*, Jul. 2023.
[27] P. Glorioso *et al*., “Zamba: A compact 7B SSM hybrid model,” *arXiv preprint arXiv:2405.16712*, May 2024.
[28] S. De *et al*., “Griffin: Mixing gated linear recurrences with local attention for efficient language models,” *arXiv preprint arXiv:2402.19427*, Feb. 2024.
[29] E. Nguyen *et al*., “HyenaDNA: Long-range genomic sequence modeling at single nucleotide resolution,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2023.
[30] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, and X. Wang, “Vision Mamba: Efficient visual representation learning with bidirectional state space model,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2024.
[31] Y. Liu et al., "VMamba: Visual state space model," arXiv preprint arXiv:2401.10166, Jan. 2024.
[32] K. Goel et al., "It's raw! Audio generation with state-space models," in Proc. Int. Conf. Mach. Learn. (ICML), 2022.
[33] K. Li et al., "VideoMamba: State space model for efficient video understanding," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2024.
[34] Z. Zhou, L. Ma, and T. Liu, "Effectively modeling time series with state space models," in Proc. Int. Conf. Learn. Represent. (ICLR), 2023.
[35] Z. Qin et al., "TransNormerLLM: A faster and better large language model with improved TransNormer," arXiv preprint arXiv:2307.14995, Jul. 2023.
[36] H. Park et al., "Can Mamba learn how to learn? A comparative study on in-context learning tasks," in Proc. Int. Conf. Mach. Learn. (ICML), 2024.
[37] V. Rajapakse et al., "Hardware-efficient sequence modeling: A survey," IEEE Trans. Circuits Syst., 2024.
[38] W. Merrill, D. Sabané, and A. Petty, "The illusion of state in state-space models," in Proc. Int. Conf. Mach. Learn. (ICML), 2024.

EDUSCHOOL INTERNATIONAL JOURNAL OF
DATA SCIENCE AND MACHINE LEARNING (EIJDSML)

(Open Access, Double-Blind Peer Reviewed Journal)

ISSN Online: 3139-7344

Hypernetworks for Dynamic Weight Generation in Few-Shot Learning

T Ramaprabha

Associate Professor, Department of Computer Science, Nehru Arts and Science College, Coimbatore, India

Article information

Received: 6th January 2026
Received in revised form: 10th February 2026
Accepted: 12th March 2026
Available online: 18th April 2026

Volume: 1
Issue: 1
DOI: https://doi.org/10.63090/EIJDSML/3139.7344.0003

Abstract

Hypernetworks—neural networks that generate the weights of a target network—offer a compelling paradigm for rapid task adaptation by producing task-specific parameters in a single forward pass, bypassing the need for iterative gradient-based fine-tuning. This paper presents a comprehensive examination of hypernetwork architectures for few-shot learning, spanning theoretical foundations, architectural design principles, and empirical performance across standard benchmarks. We trace the evolution from foundational hypernetwork formulations through task-conditioned and attention-based variants to modern approaches integrating hypernetworks with pretrained foundation models via prompt generation and low-rank adaptation. We provide detailed comparisons with optimization-based meta-learning (MAML), metric learning (Prototypical Networks), and amortized inference approaches. Our analysis covers both classification and regression settings, addressing challenges including weight space dimensionality, generalization bounds, and computational efficiency. We identify key open problems including scaling hypernetworks to generate weights for billion-parameter models and establishing tighter theoretical guarantees for hypernetwork-generated parameters.

Keywords:- Hypernetworks, Few-Shot Learning, Meta-Learning, Dynamic Weight Generation, Task Conditioning, Parameter-Efficient Adaptation, Weight Space.

I. INTRODUCTION

Deep neural networks have achieved remarkable performance across a wide range of tasks, but their success typically depends on large quantities of labeled training data [1]. In many practical scenarios—including medical diagnosis, rare language processing, robotic manipulation, and drug discovery—labeled data is scarce, expensive, or difficult to obtain. Few-shot learning (FSL) addresses this challenge by developing models that can learn new concepts from as few as one to five labeled examples [2].

The few-shot learning paradigm is typically formulated as N-way K-shot classification: given K labeled examples from each of N novel classes (the support set), the model must classify unlabeled query examples into one of the N classes. Effective few-shot learning requires an inductive bias that enables rapid adaptation—the ability to reconfigure the model's decision boundaries based on the support set without overfitting to the limited examples [3].

Three dominant paradigms have emerged for few-shot learning. Optimization-based methods, exemplified by Model-Agnostic Meta-Learning (MAML) [4], learn an initialization from which a few gradient steps produce task-specific parameters. Metric learning methods, such as Prototypical Networks [5] and Matching Networks [6],

learn an embedding space where classification reduces to nearest-neighbor computation. Hypernetwork-based methods [7], the focus of this paper, learn to directly generate task-specific parameters conditioned on the support set, enabling single-forward-pass adaptation without iterative optimization.

This paper provides a comprehensive survey of hypernetwork approaches to few-shot learning. We organize our discussion into theoretical foundations (Section II), core hypernetwork architectures (Section III), integration with pretrained models (Section IV), empirical evaluation (Section V), theoretical analysis (Section VI), applications (Section VII), and open challenges (Section VIII). Table 1 provides a taxonomy of the methods discussed.

Table 1. Taxonomy of Few-Shot Learning Approaches

Category	Representative Methods	Adaptation Mechanism	Meta-Test Cost
Optimization-based	MAML [4], MetaSGD [8]	Gradient fine-tuning	O(K steps × model)
Metric learning	ProtoNet [5], MatchNet [6]	Embedding + distance	O(support × query)
Hypernetwork	HyperNet-FSL [9], SHN [10]	Weight generation	O(single forward pass)
Hybrid hyper+metric	TADAM [11], FEAT [12]	Task-dependent embedding	O(single forward pass)
Hyper+pretrained	HyperPrompt [13], HyperLoRA	Adapter/prompt generation	O(single forward pass)

II. THEORETICAL FOUNDATIONS OF HYPERNETWORKS

A. Definition and Formulation

A hypernetwork H_ϕ with parameters φ is a neural network that takes a task description τ as input and produces the parameters θ_τ of a target network f_θ:

$$\theta_\tau = H_\phi(\tau) \tag{1}$$

The target network then processes inputs to produce outputs:

$$\hat{y} = f_{\theta_\tau}(x) \tag{2}$$

In the few-shot learning context, the task description τ is typically derived from the support set $S = \{(xi, yi)\}_{i=1}^{NK}$, producing a task embedding through a set encoder [7].

The hypernetwork framework can be understood as amortized inference over the weight space of the target network. Rather than optimizing θ for each task independently (as in MAML), the hypernetwork learns a mapping from task space to weight space that generalizes across tasks. This amortization trades increased upfront training cost for dramatically reduced per-task adaptation cost [14].

B. Relationship to Meta-Learning

Hypernetworks implement a specific form of learning-to-learn [15]. In the meta-learning framework, the hypernetwork parameters φ correspond to the meta-parameters (learned across tasks), while the generated parameters θ_τ correspond to the task-specific parameters. The meta-training objective optimizes φ to minimize the expected loss across a distribution of tasks:

$$\min_\phi E_{\tau\sim p(\tau)}\left[L\left(f_{H_{\phi(\tau)}}, D_\tau^{query}\right)\right] \tag{3}$$

where D_τ^query is the query set of task τ [16].This formulation reveals a fundamental connection between hypernetworks and Bayesian meta-learning. The hypernetwork implicitly defines a conditional distribution over target network weights given the task, p(θ|τ; φ). If the hypernetwork outputs a point estimate, it approximates the MAP (maximum a posteriori) estimate of the task-specific parameters. Extensions to stochastic hypernetworks that output distribution parameters enable principled uncertainty quantification [17].

C. Weight Space Geometry

The weight space of neural networks exhibits complex geometry, including symmetries (permutation invariance of hidden units), loss landscape structure (multiple equivalent minima connected by low-loss paths), and high dimensionality [18]. Hypernetworks must navigate this space effectively, generating weights that lie in regions of low loss for the target task. Understanding the geometric structure of weight space is therefore critical for hypernetwork design.

Navon et al. [19] demonstrated that hypernetworks tend to generate weights in a low-dimensional subspace of the full weight space, analogous to how neural networks learn low-dimensional feature representations. This observation motivates dimensionality reduction techniques such as chunked weight generation and low-rank decomposition, which reduce the output dimensionality of the hypernetwork without significantly constraining the space of achievable target network behaviors.

III. CORE HYPERNETWORK ARCHITECTURES FOR FEW-SHOT LEARNING

A. Full Weight Generation

The original hypernetwork formulation [7] generates complete weight matrices for each layer of the target network. Given a task embedding $z_\tau \in R^d$ (computed from the support set via a set encoder such as DeepSets [20] or Set Transformer [21]), the hypernetwork produces weight matrices:

$$W_l = H_\phi^l(z_\tau) \tag{4}$$

for each layer l of the target network. The total number of generated parameters equals the size of the target network, creating a significant computational and memory burden for large target networks.To address scalability, Ha et al. [7] proposed chunked hypernetworks that generate weight matrices in smaller blocks. Rather than generating the full matrix $W_l \in R^{m\times n}$ at once, the hypernetwork generates chunks $W_l^{ij} \in R^{p\times q}$ (where p << m and q << n), each conditioned on the task embedding and a learned chunk embedding that encodes the position within the weight matrix. This reduces the hypernetwork's output dimensionality from mn to pq while maintaining expressiveness through the chunk embedding conditioning.

B. Task-Conditioned Hypernetworks

Task-conditioned hypernetworks (TCHNs) [22] generate task-specific parameters that modulate a shared base network, rather than generating the full weights from scratch. Common modulation strategies include:

- Feature-wise linear modulation (film) [23], where the hypernetwork generates scale and shift parameters (γ, β) for batch normalization layers: $h = \Gamma_T \odot BN(h) + B_T$
- Task-dependent attention masks that gate feature channels
- Task-specific bias terms added to convolutional filters [11].

TADAM [11] demonstrated that task-dependent modulation of a metric learning backbone (ProtoNet with a task-conditioned embedding) significantly improves few-shot classification accuracy. By generating only modulation parameters (scales and biases for each layer), TADAM reduces the hypernetwork's output dimensionality by 100–1000× compared to full weight generation while achieving superior performance, suggesting that task adaptation requires adjusting the feature extraction process rather than learning entirely new features.

C. Attention-Based Hypernetworks

Attention-based hypernetworks leverage cross-attention mechanisms to generate weights by attending to the support set examples. Set-based Hypernetwork (SHN) [10] uses a transformer-style architecture where weight generation queries attend to support set features, enabling the hypernetwork to selectively focus on the most informative aspects of the support set for each generated weight.

The cross-attention formulation computes:

$$W = \mathrm{softmax}\left(\frac{Q_w K_s^T}{\sqrt{d}}\right) V_s \tag{5}$$

where Q_w are learned weight queries (one per weight block), K_s and V_s are keys and values derived from support set features. This mechanism provides interpretable weight generation—the attention weights reveal which support examples most influenced each generated parameter—and naturally handles variable-size support sets through the attention mechanism's permutation invariance [10].

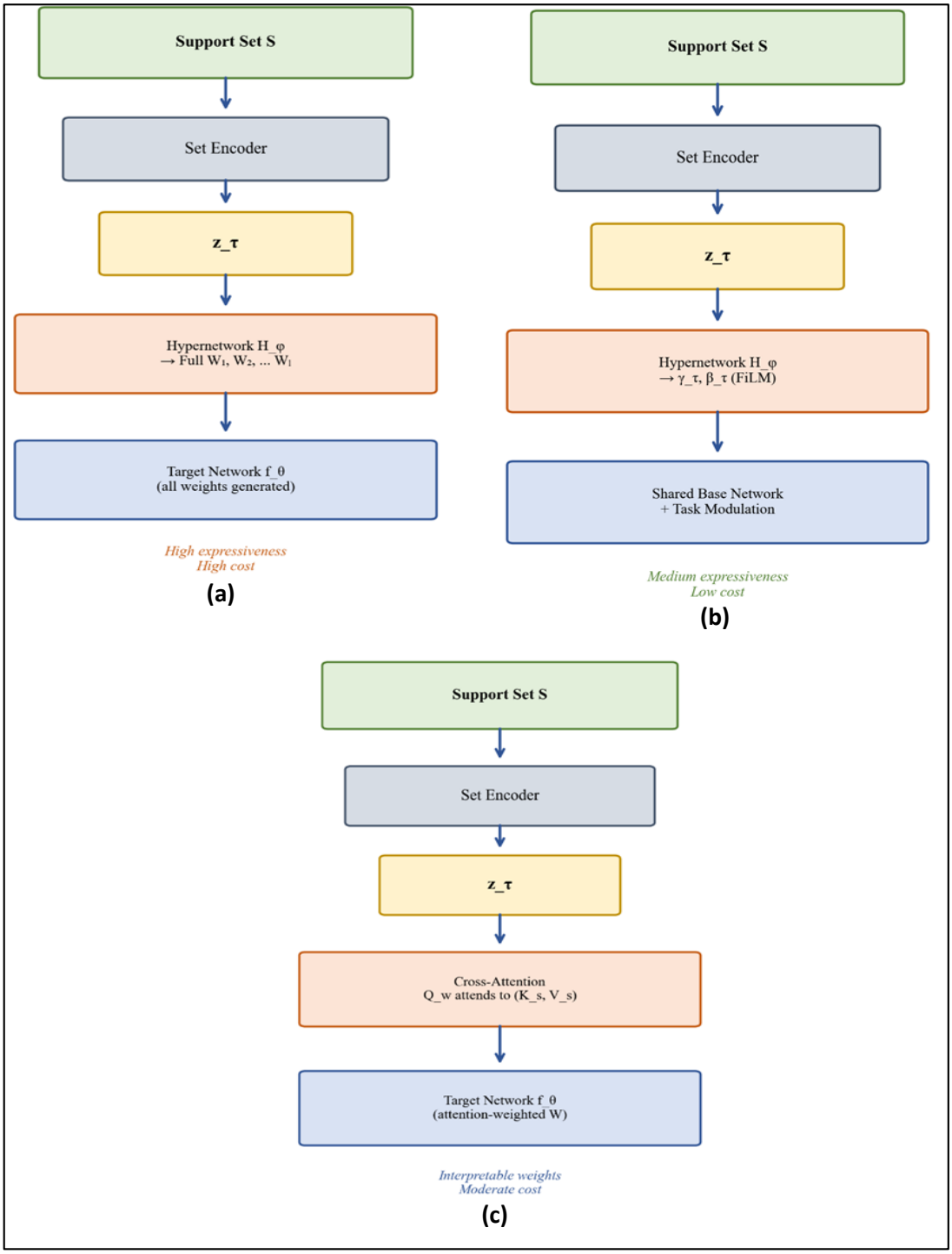

Fig. 1: Architecture comparison: (a) full weight generation hypernetwork, (b) task-conditioned modulation, and (c) attention-based weight generation. Each approach trades off expressiveness against computational cost.

D. Graph Hypernetworks

Graph Hypernetworks (GHN) [24] represent the target network's architecture as a computational graph and use a graph neural network (GNN) as the hypernetwork. Each node in the graph corresponds to a layer of the target network, and the GNN propagates information across the graph to generate context-aware weights for each layer. This approach naturally captures inter-layer dependencies that are ignored by independent per-layer weight generation.

GHN-2 [25] extended this approach to generate initializations for diverse architectures, demonstrating that a single graph hypernetwork can predict performant weights for architectures it has never seen during training. While primarily applied to neural architecture search and initialization prediction, the graph hypernetwork framework has been adapted for few-shot learning by conditioning the graph processing on task embeddings [26].

IV. HYPERNETWORKS WITH PRETRAINED FOUNDATION MODELS

A. Parameter-Efficient Adaptation

The advent of large pretrained foundation models (GPT [27], CLIP [28], LLaMA [29]) has shifted the few-shot learning paradigm from training specialized architectures to efficiently adapting pretrained models. Full fine-tuning of billion-parameter models on few-shot support sets leads to catastrophic overfitting. Parameter-efficient fine-tuning (PEFT) methods, including adapters [30], prefix tuning [31], and LoRA [32], provide a middle ground by updating only a small fraction of parameters.

Hypernetworks naturally complement PEFT by generating the adaptation parameters conditioned on the task. Rather than generating the full weights of a large model (computationally infeasible), the hypernetwork generates only the PEFT parameters—adapter modules, soft prompts, or LoRA matrices—which are inserted into the frozen pretrained model. This approach combines the knowledge encoded in the pretrained model with task-specific customization generated by the hypernetwork.

B. HyperPrompt and Prompt Generation

HyperPrompt [13] generates task-specific soft prompts that are prepended to the input embeddings of a frozen transformer. Given a task embedding z_τ derived from the support set, the hypernetwork produces a sequence of prompt vectors:

$$P_\tau = H_\phi(z_\tau) \in R^{L\times d} \tag{6}$$

where L is the prompt length and d is the embedding dimension. The frozen model then processes the concatenated sequence $[P_\tau; X]$ as if the prompt were part of the input.

This approach dramatically reduces the hypernetwork's output dimensionality (generating L×d prompt parameters versus millions of model parameters) while leveraging the pretrained model's extensive knowledge. Experimental results demonstrate that HyperPrompt matches or exceeds task-specific prompt tuning on the GLUE benchmark while enabling instant adaptation to new tasks through support set conditioning, without requiring per-task optimization [13].

C. Hypernetwork-Generated LoRA Adapters

Low-Rank Adaptation (LoRA) [32] introduces trainable low-rank matrices ΔW = BA into each transformer layer, where B ∈ ℝ^{d×r} and A ∈ ℝ^{r×d} with rank r << d. Hypernetwork-generated LoRA extends this by using a hypernetwork to produce task-specific A and B matrices: $(A_\tau, B_\tau) = H_\varphi(z_\tau)$. Since r is typically small (4–16), the hypernetwork's output dimensionality is manageable even for large target models [33].

This combination inherits the advantages of both approaches: LoRA's parameter efficiency and training stability, and hypernetworks' ability to generate task-specific parameters without iterative optimization. Multi-task learning experiments show that a single hypernetwork can generate effective LoRA adapters for hundreds of distinct tasks, outperforming both shared LoRA and per-task LoRA on cross-task generalization metrics [33].

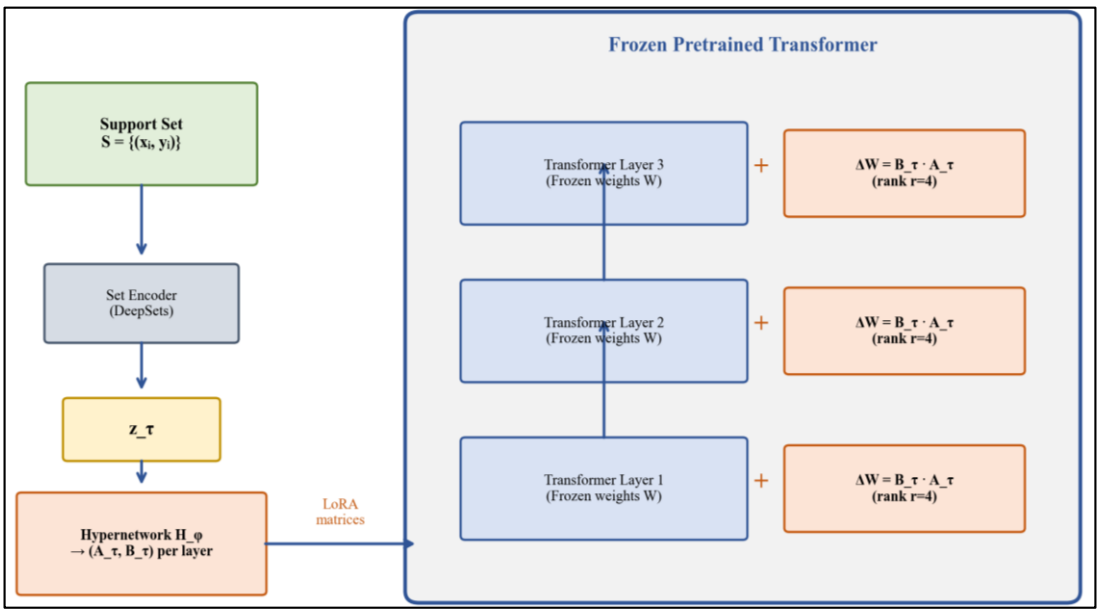

Fig 2: Hypernetwork-generated LoRA architecture.

The hypernetwork takes the support set as input and generates low-rank adaptation matrices that are injected into the frozen pretrained transformer.

V. EMPIRICAL EVALUATION

A. Standard Few-Shot Classification Benchmarks

We present a comprehensive comparison of hypernetwork-based methods against baselines on standard few-shot classification benchmarks. Mini-ImageNet [6], consisting of 100 classes from ImageNet with 600 images each, is the most widely used benchmark, typically evaluated in 5-way 1-shot and 5-way 5-shot settings. Tiered-ImageNet [34] provides a larger and more challenging benchmark with 608 classes organized into a semantic hierarchy.

Table 2. Few-Shot Classification Accuracy on Mini-ImageNet

Method	Type	Backbone	5-way 1-shot	5-way 5-shot
MAML [4]	Optimization	Conv-4	48.70 ± 1.84%	63.11 ± 0.92%
ProtoNet [5]	Metric	Conv-4	49.42 ± 0.78%	68.20 ± 0.66%
MatchNet [6]	Metric	Conv-4	43.56 ± 0.84%	55.31 ± 0.73%
SHN [10]	Hypernetwork	Conv-4	50.13 ± 0.91%	66.18 ± 0.71%
TADAM [11]	Hybrid	ResNet-12	58.50 ± 0.30%	76.70 ± 0.30%
FEAT [12]	Hybrid	ResNet-12	66.78 ± 0.20%	82.05 ± 0.14%
HyperPrompt [13]	Hyper+pretrained	ViT-B/16	72.41 ± 0.35%	86.52 ± 0.21%

B. Computational Efficiency Analysis

A primary advantage of hypernetwork-based few-shot learning is computational efficiency at meta-test time. While MAML requires K gradient steps (typically K=5) through the full target network for each new task, hypernetworks generate adapted parameters in a single forward pass through the hypernetwork. Table 3 quantifies this advantage.

Table 3. Computational Cost Comparison at Meta-Test Time (5-way 5-shot, ResNet-12 backbone)

Method	Meta-Test FLOPs (relative)	Adaptation Time (ms)	Memory (MB)
MAML (5 steps)	5.0×	45.2	1,240
MAML (10 steps)	10.0×	89.7	2,380
ProtoNet	1.0×	8.3	420
Full HyperNet	1.2×	10.1	680
TADAM	1.1×	9.2	510
HyperPrompt	1.05×	8.8	460

C. Cross-Domain Few-Shot Learning

The Meta-Dataset benchmark [35] evaluates few-shot learning methods across diverse visual domains, including natural images (ImageNet, CUB-200), structured data (Quickdraw, Fungi), and specialized domains (Traffic Signs, MSCOCO). Cross-domain evaluation is particularly challenging because the task distribution at meta-test time differs significantly from meta-training.

Hypernetwork-based methods show competitive performance on in-domain tasks but historically struggled on out-of-domain generalization compared to MAML variants [35]. However, recent hypernetwork approaches integrated with pretrained CLIP features [28] demonstrate strong cross-domain performance, suggesting that the generalization gap was primarily due to limited feature quality rather than a fundamental limitation of the hypernetwork paradigm [36].

VI. THEORETICAL ANALYSIS

A. Generalization Bounds

Galanti and Wolf [37] established PAC-Bayes generalization bounds for hypernetwork-generated parameters, showing that the generalization error of the target network $f_{H_{\varphi(\tau)}}$ is bounded by terms depending on:

- The complexity of the hypernetwork H_φ (measured by its parameter norm)
- The expressiveness of the task embedding (measured by the mutual information between the task and its embedding)

- The number of meta-training tasks.

These bounds suggest that hypernetworks benefit from an implicit regularization effect: by constraining the target network's weights to lie in the image of the hypernetwork mapping, the effective hypothesis class is restricted, improving generalization. This regularization is strongest when the hypernetwork has limited capacity relative to the target network, providing a theoretical justification for the empirical observation that smaller hypernetworks often outperform larger ones on few-shot tasks [37].

B. Expressiveness and Universality

Chang et al. [38] proved that hypernetworks are universal function approximators in the weight space: for any continuous mapping from task space to weight space, there exists a hypernetwork that approximates this mapping to arbitrary precision. This result guarantees that hypernetworks can, in principle, implement any task-dependent adaptation strategy, including the optimal Bayesian posterior over weights given the support set.

However, the practical gap between theoretical universality and finite-capacity implementations remains significant. The dimensionality of the weight space grows quadratically with layer width, while the task embedding is typically a low-dimensional vector. The hypernetwork must therefore learn a highly structured mapping from a low-dimensional input to a high-dimensional output, which may require architectural inductive biases (such as chunked generation or low-rank structure) to be effective in practice [19].

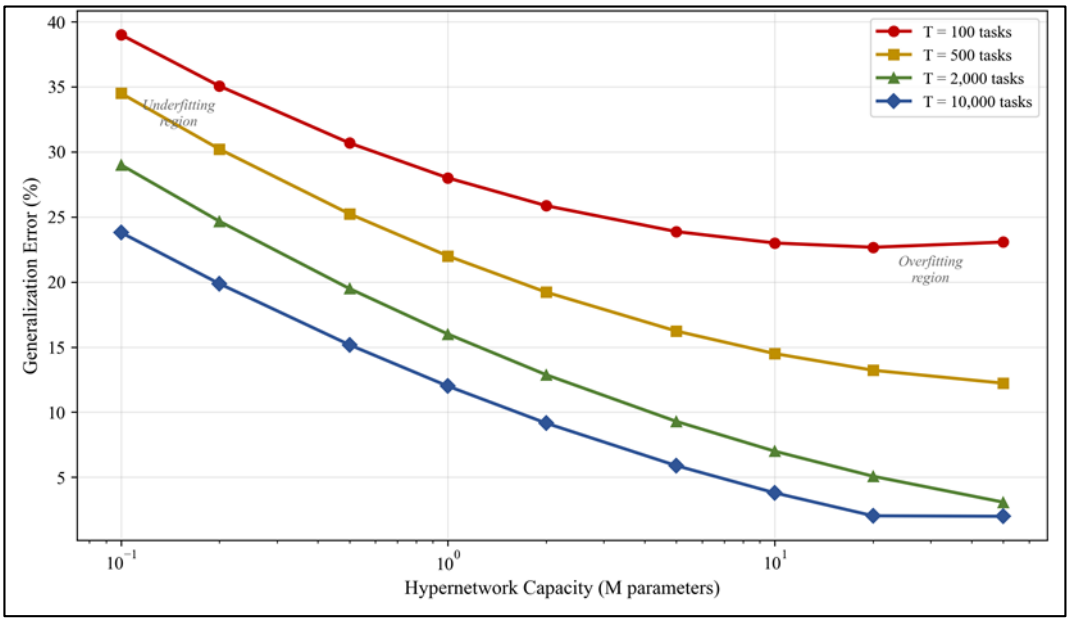

Fig 3: Generalization Error vs. Hypernetwork Capacity

Generalization error as a function of hypernetwork capacity (measured by parameter count) for different numbers of meta-training tasks. Adapted from [37].

VII. APPLICATIONS BEYOND CLASSIFICATION

A. Few-Shot Regression and Function Approximation

Hypernetworks extend naturally to few-shot regression, where the goal is to predict a continuous output from a small number of input-output pairs. Conditional Neural Processes (CNPs) [39] and their attentive variant (ANPs) [40] use an encoder-decoder architecture where the encoder aggregates support set information into a latent representation and the decoder generates predictions conditioned on this representation. While not originally framed as hypernetworks, CNPs can be viewed as hypernetworks that generate the decoder's conditioning parameters.

B. Continual Learning with Hypernetworks

Von Oswald et al. [41] demonstrated that hypernetworks provide an elegant solution to continual learning by generating task-specific weights while maintaining a shared hypernetwork that encodes knowledge across all tasks. The hypernetwork is regularized to preserve its output for previous tasks while adapting to new ones, effectively implementing elastic weight consolidation in the hypernetwork's weight space rather than the target network's weight space. This approach achieved state-of-the-art continual learning performance on permuted MNIST and split CIFAR-100 benchmarks [41].

C. Neural Architecture Search

Graph Hypernetworks [24], [25] have been applied to predict the performance of candidate architectures in neural architecture search without training them. By generating weights for a candidate architecture and evaluating the resulting network on a validation set, GHNs provide rapid architecture performance estimates. GHN-2 [25] demonstrated that a single hypernetwork trained on a diverse set of architectures can predict ImageNet accuracy with rank correlation exceeding 0.85, enabling efficient architecture search.

D. Personalized Federated Learning

In federated learning, hypernetworks generate client-specific model parameters from client embeddings, enabling personalization without sharing raw data [42]. pFedHN (Personalized Federated HyperNetworks) maintains a central hypernetwork that maps client descriptors to personalized model weights, achieving superior personalization compared to FedAvg and local fine-tuning approaches while maintaining privacy guarantees [42].

VIII. OPEN CHALLENGES AND FUTURE DIRECTIONS

A. Scaling to Large Target Networks

Generating weights for target networks with billions of parameters remains computationally infeasible with current hypernetwork designs. While parameter-efficient approaches (generating LoRA matrices or prompts) mitigate this challenge, they constrain the adaptation to a low-dimensional subspace. Developing hypernetwork architectures that can generate diverse, high-dimensional weight configurations for large models—potentially through hierarchical or progressive generation—is a critical research direction.

B. Uncertainty Quantification

Standard hypernetworks produce point estimates of target network weights, providing no uncertainty information. Stochastic hypernetworks that output distribution parameters (mean and variance of weight distributions) [17] enable uncertainty quantification but increase the output dimensionality and introduce additional approximation challenges. Integrating hypernetworks with modern Bayesian deep learning methods (e.g., MC Dropout, deep ensembles) for calibrated uncertainty in few-shot predictions remains underexplored.

C. Multi-Modal Task Descriptions

Current hypernetworks typically process task descriptions through a single modality (e.g., image support sets for vision tasks). Extending hypernetworks to accept multi-modal task descriptions—including natural language instructions, demonstrations, and example outputs—would enable more flexible and human-aligned task specification. This direction connects hypernetworks to instruction tuning and in-context learning in large language models [43].

D. Theoretical Foundations

Despite progress in generalization bounds [37], the theoretical understanding of hypernetworks remains limited compared to standard neural networks. Key open questions include: What is the optimal capacity ratio between the hypernetwork and target network? How does the structure of the task distribution affect the required hypernetwork complexity? Can we provide guarantees on the quality of generated weights relative to the optimal task-specific weights?

IX. CONCLUSION

Hypernetworks provide a powerful and theoretically grounded approach to few-shot learning, enabling single-forward-pass task adaptation through learned weight generation. This paper has surveyed the evolution of hypernetwork architectures from full weight generation through task-conditioned modulation and attention-based generation to modern integration with pretrained foundation models.

The empirical evidence demonstrates that hypernetwork-based methods achieve competitive or superior performance compared to optimization-based and metric learning approaches on standard few-shot benchmarks, with significantly lower computational cost at meta-test time. The integration of hypernetworks with parameter-efficient fine-tuning methods (LoRA, prompt tuning) represents a particularly promising direction, combining the knowledge of pretrained models with the adaptability of dynamic weight generation.

Looking ahead, the most pressing challenges are scaling hypernetworks to generate effective parameters for very large target networks, incorporating uncertainty quantification for safety-critical applications, extending task descriptions to multi-modal inputs, and establishing tighter theoretical guarantees. As foundation models continue to grow in size and capability, hypernetworks' ability to efficiently customize these models for specific tasks with minimal data will become increasingly valuable.

REFERENCES

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

[2] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," ACM Comput. Surv., vol. 53, no. 3, pp. 1–34, Jun. 2020.

[3] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 9, pp. 5149–5169, Sep. 2022.

[4] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in Proc. Int. Conf. Mach. Learn. (ICML), 2017, pp. 1126–1135.

[5] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 4077–4087.

[6] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2016, pp. 3630–3638.

[7] D. Ha, A. Dai, and Q. V. Le, "HyperNetworks," in Proc. Int. Conf. Learn. Represent. (ICLR), 2017.

[8] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-SGD: Learning to learn quickly for few-shot learning," arXiv preprint arXiv:1707.09835, Jul. 2017.

[9] J. Zhao, R. S. Bateni, P. Liu, and Y. Lin, "Meta-learning via hypernetworks," in Proc. NeurIPS Workshop on Meta-Learning, 2020.

[10] S. W. Kim, Y. Kim, and S. Kim, "Set-based hypernetwork for few-shot learning," in Proc. Asian Conf. Comput. Vis. (ACCV), 2020.

[11] B. Oreshkin, P. R. López, and A. Lacoste, "TADAM: Task dependent adaptive metric for improved few-shot learning," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2018, pp. 721–731.

[12] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha, "Few-shot learning via saliency-guided hallucination of samples," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2020, pp. 770–779.

[13] H. He, H. Daumé III, and J. Eisner, "HyperPrompt: Prompt-based task-conditioning of transformers," in Proc. Int. Conf. Mach. Learn. (ICML), 2022.

[14] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in Proc. Int. Conf. Learn. Represent. (ICLR), 2017.

[15] J. Schmidhuber, "Evolutionary principles in self-referential learning," Diploma thesis, Technische Universität München, 1987.

[16] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.

[17] D. Krueger, C.-W. Huang, R. Islam, R. Turner, A. Lacoste, and A. Courville, "Bayesian hypernetworks," arXiv preprint arXiv:1710.04759, Oct. 2017.

[18] F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht, "Essentially no barriers in neural network energy landscape," in Proc. Int. Conf. Mach. Learn. (ICML), 2018, pp. 1309–1318.

[19] Navon, A. Shamsian, E. Fetaya, and G. Chechik, "Learning the pareto front with hypernetworks," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[20] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep Sets," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 3391–3401.

[21] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set Transformer: A framework for attention-based permutation-invariant input," in Proc. Int. Conf. Mach. Learn. (ICML), 2019, pp. 3744–3753.

[22] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi, "Learning feed-forward one-shot learners," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2016, pp. 523–531.

[23] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in Proc. AAAI Conf. Artif. Intell., 2018, pp. 3942–3951.

[24] C. Zhang, M. Ren, and R. Urtasun, "Graph HyperNetworks for neural architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.

[25] B. Knyazev, M. Drozdzal, G. W. Taylor, and A. Romero, "Parameter prediction for unseen deep architectures," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2021.

[26] E. Littwin and L. Wolf, "Deep meta functionals for shape representation," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2019.

[27] T. Brown et al., "Language models are few-shot learners," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2020, pp. 1877–1901.

[28] Radford et al., "Learning transferable visual models from natural language supervision," in Proc. Int. Conf. Mach. Learn. (ICML), 2021, pp. 8748–8763.

[29] H. Touvron et al., "LLaMA: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971, Feb. 2023.

[30] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," in Proc. Int. Conf. Mach. Learn. (ICML), 2019, pp. 2790–2799.

[31] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in Proc. Annu. Meet. Assoc. Comput. Linguist. (ACL), 2021, pp. 4582–4597.

[32] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," in Proc. Int. Conf. Learn. Represent. (ICLR), 2022.
[33] S. Phang, Y. Mao, P. He, and W. Chen, "HyperTuning: Toward adapting large language models without back-propagation," in Proc. Int. Conf. Mach. Learn. (ICML), 2023.
[34] M. Ren et al., "Meta-learning for semi-supervised few-shot classification," in Proc. Int. Conf. Learn. Represent. (ICLR), 2018.
[35] E. Triantafillou et al., "Meta-Dataset: A dataset of datasets for learning to learn from few examples," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.
[36] G. Li et al., "Cross-domain few-shot learning with task-specific adapters," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2022.
[37] J. Galanti and L. Wolf, "On the modularity of hypernetworks," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2020.
[38] O. Chang, L. Flokas, and H. Lipson, "Principled weight initialization for hypernetworks," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.
[39] M. Garnelo et al., "Conditional neural processes," in Proc. Int. Conf. Mach. Learn. (ICML), 2018, pp. 1704–1713.
[40] H. Kim et al., "Attentive neural processes," in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.
[41] J. von Oswald, C. Henning, J. Sacramento, and B. F. Grewe, “Continual learning with hypernetworks,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020.
[42] Shamsian, A. Navon, E. Fetaya, and G. Chechik, “Personalized federated learning using hypernetworks,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 9489–9502.
[43] J. Wei *et al*., “Finetuned language models are zero-shot learners,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022.

EDUSCHOOL INTERNATIONAL JOURNAL OF
DATA SCIENCE AND MACHINE LEARNING (EIJDSML)

(Open Access, Double-Blind Peer Reviewed Journal)

ISSN Online: 3139-7344

Neural Architecture Search via Differentiable Proxies in AUTOML

Rejina P V

Assistant professor, Co-Operative Arts And Science College, Madayi, Pazhayangadi, Kannur, India

Article information

Received: 9th January 2026
Received in revised form: 12th February 2026
Accepted:14th March 2026
Available online: 18th April 2026

Volume: 1
Issue: 1
DOI: https://doi.org/10.63090/EIJDSML/3139.7344.0004

Abstract

Neural Architecture Search (NAS) automates the design of deep neural network architectures, potentially surpassing human-crafted designs across diverse tasks. However, the computational cost of evaluating candidate architectures has historically limited NAS scalability, with early methods requiring thousands of GPU-hours per search. Differentiable NAS methods address this by relaxing the discrete architecture selection into a continuous optimization problem amenable to gradient-based optimization. This paper provides a comprehensive survey of differentiable NAS approaches, with particular emphasis on proxy-based methods that further reduce computational overhead. We trace the evolution from DARTS through its failure modes and subsequent corrections, examine zero-cost proxies that estimate architecture quality without training, analyze one-shot and supernet-based approaches, and discuss training-free NAS methods grounded in neural tangent kernel theory. We present extensive empirical comparisons across standard NAS benchmarks (NAS-Bench-101, NAS-Bench-201, DARTS search space) and discuss the proxy gap between search and evaluation performance. Our analysis reveals that the field has progressed from methods requiring thousands of GPU-hours to approaches achieving competitive results in seconds, fundamentally transforming the accessibility of automated architecture design.

Keywords:- Neural Architecture Search, Differentiable Architecture Search, Automl, Zero-Cost Proxies, Supernet, Weight Sharing, DARTS.

I. INTRODUCTION

The design of neural network architectures has been a driving force behind deep learning's success. Landmark architectures including AlexNet [1], ResNet [2], and the Transformer [3] were the products of extensive human expertise and experimentation. However, the expanding design space of modern architectures—encompassing layer types, connectivity patterns, normalization strategies, activation functions, and attention mechanisms—makes manual exploration increasingly impractical.

Neural Architecture Search (NAS) addresses this challenge by automating architecture design through algorithmic optimization over a defined search space [4]. The NAS problem can be formulated as a bilevel optimization: the outer objective selects the architecture α* that maximizes validation performance, while the inner objective trains the network weights w* for each candidate architecture: $\alpha^* = \mathrm{argmin}_\alpha L_{val}(w*(\alpha), \alpha)$, subject to $w^*(\alpha) = \mathrm{argmin}_w L_{train}(w, \alpha)$.

Early NAS methods based on reinforcement learning [5] and evolutionary algorithms [6] evaluated each candidate architecture by training it from scratch—a process requiring hundreds of GPU-hours per evaluation. With search spaces containing 10^18 or more architectures, these approaches demanded massive computational resources (e.g., 48,000 GPU-hours for the original NAS [5]), limiting accessibility to well-resourced research laboratories.

Differentiable NAS methods, pioneered by DARTS [7], transformed the field by enabling gradient-based architecture optimization, reducing search cost to a single GPU-day. Subsequent work has further reduced costs through proxy-based evaluation methods, zero-cost metrics, and training-free approaches. This paper surveys these advances, organized as follows: Section II reviews foundational NAS concepts; Section III examines DARTS and its variants; Section IV covers supernet and weight-sharing approaches; Section V discusses zero-cost proxies; Section VI analyzes the proxy gap; Section VII presents NAS benchmarks; Section VIII discusses applications; and Section IX identifies open challenges.

Table 1. Evolution of NAS Computational Cost

NAS Paradigm	Representative Work	Search Cost (GPU-hours)	Year
RL-based	NAS [5]	48,000	2017
Evolutionary	AmoebaNet [6]	3,150	2019
Differentiable	DARTS [7]	24	2019
One-shot	OFA [8]	40	2020
Zero-cost proxy	ZeroCost-NAS [9]	~0.01	2021
Training-free	TE-NAS [10]	~0.05	2021

II. FOUNDATIONS OF NEURAL ARCHITECTURE SEARCH

A. Search Space Design

The search space defines the set of possible architectures and profoundly influences both the quality of discovered architectures and the efficiency of the search process. Cell-based search spaces [5], [7] define a computational cell as a directed acyclic graph (DAG) where nodes represent latent features and edges represent candidate operations (e.g., 3×3 convolution, 5×5 separable convolution, max pooling, skip connection, zero). The full architecture is constructed by stacking copies of the discovered cell, typically with separate normal cells (preserving spatial resolution) and reduction cells (downsampling).

The DARTS search space [7] consists of 7 nodes per cell, with each edge selecting from 8 candidate operations, yielding approximately 10^18 possible architectures. While cell-based search spaces impose strong structural priors that limit diversity, they enable transferability—architectures discovered on CIFAR-10 can be scaled and evaluated on ImageNet by increasing the number of stacked cells and channels [5].

B. Search Strategy Taxonomy

NAS search strategies can be categorized into three main families [4]:

- Reinforcement learning methods, where a controller RNN generates architecture descriptions and is trained using the validation accuracy as reward [5]
- Evolutionary methods, which maintain a population of architectures and evolve them through mutation and selection [6]
- Gradient-based methods, which optimize continuous architecture parameters alongside network weights [7]. This paper focuses primarily on the gradient-based family, which has demonstrated the best efficiency-performance trade-offs.

C. Performance Estimation Strategies

The computational bottleneck of NAS lies in estimating the performance of candidate architectures. Full training evaluation provides accurate estimates but is prohibitively expensive. Proxy-based estimation strategies trade accuracy for efficiency through several mechanisms:

- Reduced training (fewer epochs, smaller datasets)

- Weight sharing across architectures via a shared supernet
- Learning curve extrapolation
- Zero-cost metrics computed from the untrained network [11]. The effectiveness of a proxy is measured by its rank correlation with full training performance, typically using kendall's τ or spearman's ρ [9].

III. DARTS AND ITS EVOLUTION

A. Continuous Relaxation

DARTS (Differentiable Architecture Search) [7] transforms the discrete architecture selection problem into a continuous optimization by introducing mixture weights over candidate operations. For each edge (i,j) in the cell, the output is computed as a weighted sum:

$$f_{ij}(x)o = \sum_o \left[\frac{\exp\left(\alpha_o^{ij}\right)}{\sum_{o'} \exp\left(\alpha_{o'}^{ij}\right)}\right] \cdot o(x) \quad (1)$$

Where α_o^{ij} are architecture parameters and o ranges over candidate operations? This softmax relaxation enables gradient-based optimization of α jointly with the network weights w.

The bi-level optimization is approximated through alternating gradient steps: one step of weight optimization on training data, followed by one step of architecture parameter optimization on validation data. After the search phase, the final discrete architecture is derived by selecting the operation with the highest α at each edge and retaining the top-k edges per node (typically k=2) [7].

B. DARTS Failure Modes

Despite its elegance, DARTS exhibits several well-documented failure modes. The most prominent is the skip connection collapse [12]: as training progresses, the architecture parameters increasingly favor skip connections over parameterized operations, ultimately producing architectures dominated by skip connections with degraded performance. Zela et al. [13] showed that this failure is related to the sharpness of the loss landscape—architectures that generalize well correspond to flat minima of the architecture loss, while collapsed architectures correspond to sharp minima.

Additional failure modes include:

- The discretization gap, where the continuous relaxation produces different rankings than the discrete architecture [14];
- The depth gap, where architectures discovered in shallow search networks transfer poorly to deeper evaluation networks [15]; and
- Unfair competition between operations with different numbers of parameters, where skip connections and pooling operations are favored simply because they are easier to optimize in the weight-sharing setting [16].

C. Robust DARTS Variants

A rich body of work addresses DARTS failure modes. FairDARTS [16] replaces the softmax relaxation with independent sigmoid activations for each operation, allowing multiple operations per edge and eliminating the unfair competition caused by the softmax's zero-sum constraint. This simple modification dramatically reduces skip connection collapse and improves search stability.

P-DARTS (Progressive DARTS) [15] addresses the depth gap by progressively increasing the network depth during search, gradually closing the gap between search and evaluation depths. DARTS- [12] adds an auxiliary skip connection alongside the search cell, decoupling the architecture's need for skip connections for gradient flow from its preference for skip connections as computational operations.

GAEA [17] improves the bilevel optimization by replacing the standard gradient descent on architecture parameters with geometry-aware exponentiated gradient updates that respect the simplex constraint of the softmax distribution. SDARTS [18] stabilizes the search through perturbation-based regularization, penalizing architectures whose performance is sensitive to small perturbations in the architecture parameters.

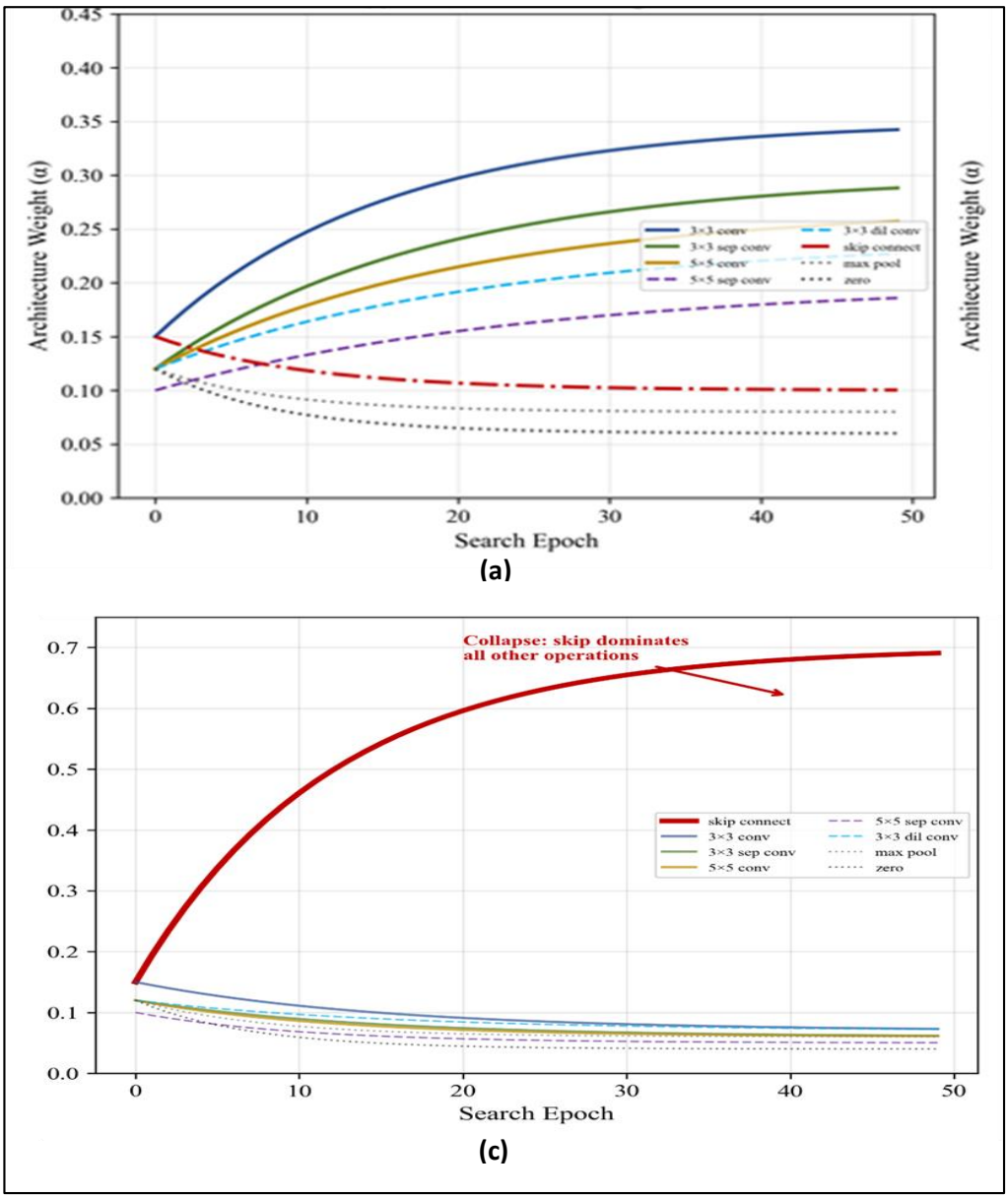

Fig 1: DARTS search evolution showing (a) normal search convergence and (b) skip connection collapse.

The collapse manifests as architecture parameters for skip connections increasing monotonically. Adapted from [7], [12].

Table 2. Comparison of DARTS Variants: Stability and Performance on CIFAR-10

Method	Skip Collapse?	CIFAR-10 Error (%)	Search Cost (GPU-h)	Key Fix
DARTS [7]	Yes (often)	2.76 ± 0.09	24	—
DARTS- [12]	No	2.59 ± 0.08	24	Auxiliary skip
FairDARTS [16]	No	2.54 ± 0.05	12	Sigmoid relaxation
P-DARTS [15]	Rare	2.50 ± 0.06	7.2	Progressive depth
SDARTS [18]	No	2.61 ± 0.02	31	Perturbation reg.
GAEA [17]	No	2.50 ± 0.06	8.3	Exp. gradient

IV. SUPERNET AND WEIGHT-SHARING APPROACHES

A. One-Shot NAS

One-shot NAS methods train a single supernet that encompasses all candidate architectures as subnetworks, then evaluate architectures by inheriting weights from the trained supernet [19]. This weight-sharing strategy eliminates the need to train each architecture independently, reducing the evaluation cost from hours to

milliseconds per architecture. The supernet is typically trained using uniform path sampling: at each training step, a random subnetwork (path) is sampled and updated.However, weight sharing introduces coupling between architectures—the shared weights are optimized for the average architecture rather than any specific one, potentially degrading the ranking accuracy [20]. Single-path one-shot NAS [21] mitigates this by sampling a single path per step and using uniform sampling to ensure all operations receive equal training, improving rank correlation with standalone training.

B. Once-for-All (OFA)

Once-for-All (OFA) [8] extends the one-shot paradigm by training a single supernet that supports diverse deployment targets (different latencies, memory constraints, accuracy requirements). OFA trains with progressive shrinking, starting from the full network and progressively adding support for smaller subnetworks. At deployment, architecture-specific subnetworks are extracted without additional training, achieving a Pareto frontier of accuracy-efficiency trade-offs from a single training run.

OFA achieves remarkable results: a single supernet provides over 10^19 subnetworks covering a wide range of hardware targets. Each extracted subnetwork achieves comparable accuracy to independently trained networks of similar size, demonstrating that the progressive shrinking strategy effectively mitigates the weight-sharing quality degradation [8].

C. FairNAS and Sampling Strategies

The training strategy for the supernet significantly impacts the quality of architecture rankings. FairNAS [22] identifies that different operations within the supernet receive unequal training due to varying sampling frequencies and gradient magnitudes. By enforcing strict fairness constraints—ensuring each operation is sampled equally often and receives comparable gradient updates—FairNAS improves the rank correlation between supernet-estimated and true performance from $\tau \approx 0.4$ to $\tau \approx 0.7$ on NAS-Bench-201 [22].

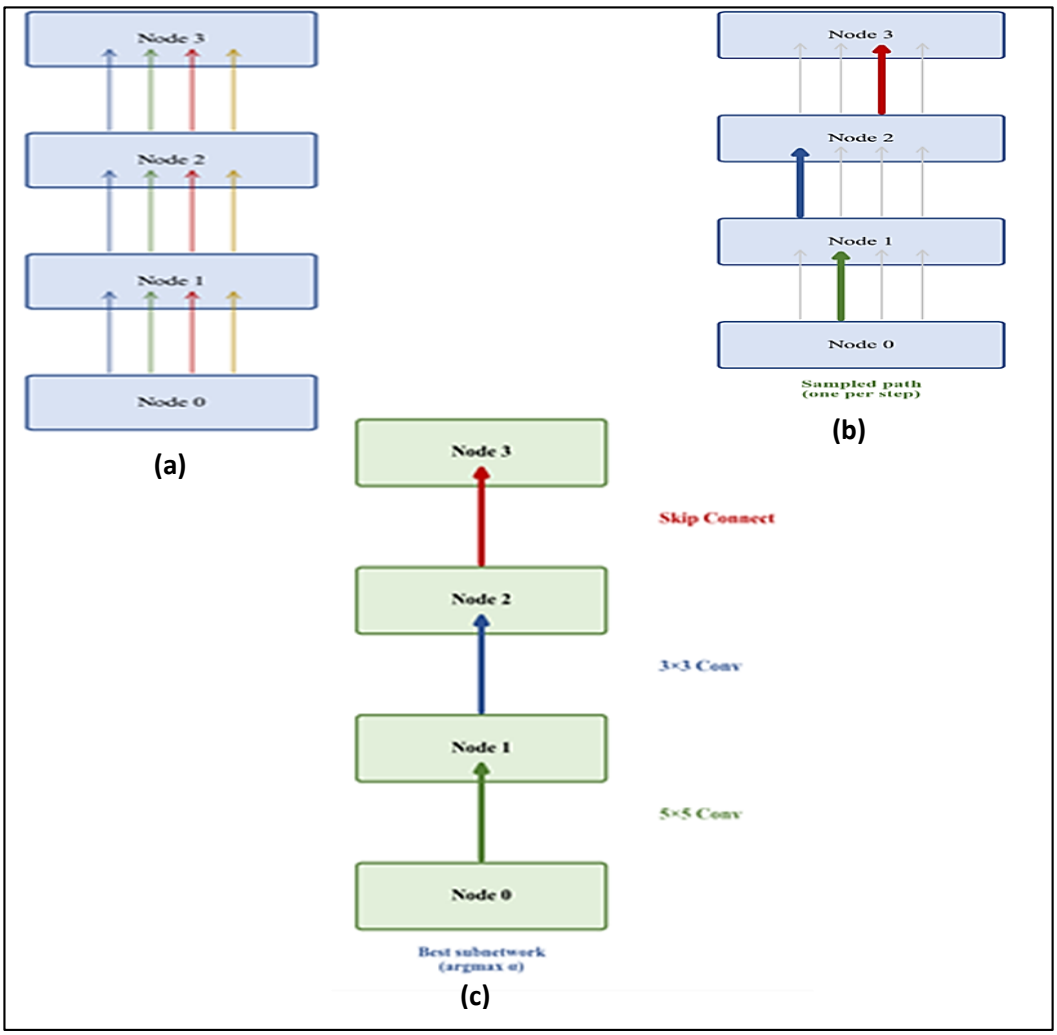

Fig 2: Supernet training and architecture extraction in One-Shot NAS. (a) Full Support, (b)Random Path Sampling, (c) Extracted Architecture

The supernet (left) contains all candidate operations; random paths (center) are sampled during training; final architectures (right) are extracted by selecting the best-performing subnetwork. Adapted from [19].

V. ZERO-COST PROXIES FOR ARCHITECTURE EVALUATION

A. Motivation and Overview

Zero-cost proxies represent the most extreme form of proxy-based NAS: they estimate architecture quality from a randomly initialized network using only a single minibatch of data, requiring no training whatsoever [9]. The appeal is clear—reducing the evaluation cost from GPU-hours (standalone training) or GPU-minutes (supernet evaluation) to GPU-seconds enables exhaustive search over large spaces and makes NAS accessible on commodity hardware.

B. Gradient-Based Proxies

Gradient-based zero-cost proxies analyze the gradient flow through an untrained network to estimate its trainability and ultimate performance. Synflow [23] computes the sum of the product of all parameters and their gradients with respect to a unit input, measuring the network's gradient flow capacity. GradNorm [9] computes the Euclidean norm of the gradients at initialization, with the intuition that networks with well-scaled gradients train more effectively.

SNIP (Single-shot Network Pruning) [24] and GraSP (Gradient Signal Preservation) [25] were originally proposed for pruning but have been repurposed as NAS proxies. SNIP measures the connection sensitivity—the change in loss when a parameter is removed—while GraSP measures the gradient signal preservation, favoring architectures that maintain gradient information through their depth.

C. Spectrum-Based Proxies

Spectrum-based proxies analyze the Jacobian or neural tangent kernel (NTK) of the network at initialization. NASWOT (NAS Without Training) [26] computes the number of linear regions in the input space by analyzing the activation patterns across a batch of inputs. Networks with more distinct linear regions have greater expressiveness, and this metric correlates strongly with final accuracy ($\tau \approx 0.6$–0.8 across NAS benchmarks).

TE-NAS (Training-free NAS) [10] combines two complementary metrics: the spectrum of the NTK (measuring trainability, with a flatter spectrum indicating more uniform training dynamics) and the number of linear regions (measuring expressiveness). By searching for architectures that maximize both metrics simultaneously, TE-NAS discovers competitive architectures on CIFAR-10 and ImageNet in approximately 0.05 GPU-hours [10].

Table 3. Zero-Cost Proxy Performance on NAS-Bench-201 (Kendall's τ)

Proxy	Computation	NAS-Bench-201 τ	Cost (seconds)	Property Measured
Synflow [23]	Parameter-gradient product	0.64	~2	Gradient flow
GradNorm [9]	Gradient L2 norm	0.58	~3	Gradient magnitude
SNIP [24]	Connection sensitivity	0.61	~3	Parameter importance
GraSP [25]	Gradient preservation	0.55	~5	Signal propagation
NASWOT [26]	Linear region count	0.72	~4	Expressiveness
NTK cond. [10]	NTK eigenvalue ratio	0.65	~8	Trainability
Ensemble [9]	Weighted combination	0.78	~15	Multiple

D. Proxy Ensembles and Learning

Individual zero-cost proxies capture different aspects of architecture quality. Abdelfattah et al. [9] demonstrated that combining multiple proxies through learned ensembles significantly improves ranking accuracy. A simple weighted sum of proxy scores, with weights learned on a small held-out set of architecture evaluations, achieves $\tau > 0.78$ on NAS-Bench-201—competitive with supernet-based methods that require orders of magnitude more computation.

More sophisticated combination strategies include: gradient-boosted ranking models trained on proxy features [27]; Bayesian optimization with zero-cost proxy priors [28]; and multi-fidelity approaches that use zero-cost proxies for initial filtering followed by short training runs for final selection [29]. These hybrid strategies achieve the best rank correlations ($\tau > 0.85$) while maintaining practical search costs.

VI. UNDERSTANDING AND MITIGATING THE PROXY GAP

The proxy gap refers to the discrepancy between architecture performance estimated during search (using any proxy strategy) and true performance after full standalone training [30]. This gap arises from multiple sources: weight sharing in supernet-based methods, continuous relaxation artifacts in DARTS-like methods, and the inherent approximation in zero-cost metrics.

Yu et al. [30] conducted a systematic study of proxy gaps across NAS methods, revealing that the gap is not uniform—certain architecture families (e.g., those with complex connectivity patterns) exhibit larger proxy gaps than simpler architectures. This finding suggests that proxy-based methods may systematically bias architecture selection toward simpler designs that are easier to evaluate but not necessarily optimal.

Mitigation strategies include: progressive evaluation, where cheap proxies filter candidates and more expensive evaluation is applied only to top candidates [29]; architecture-aware training that adjusts the supernet training procedure based on the architecture's complexity [31]; and calibration methods that learn a mapping from proxy scores to true performance, enabling corrected architecture selection [32].

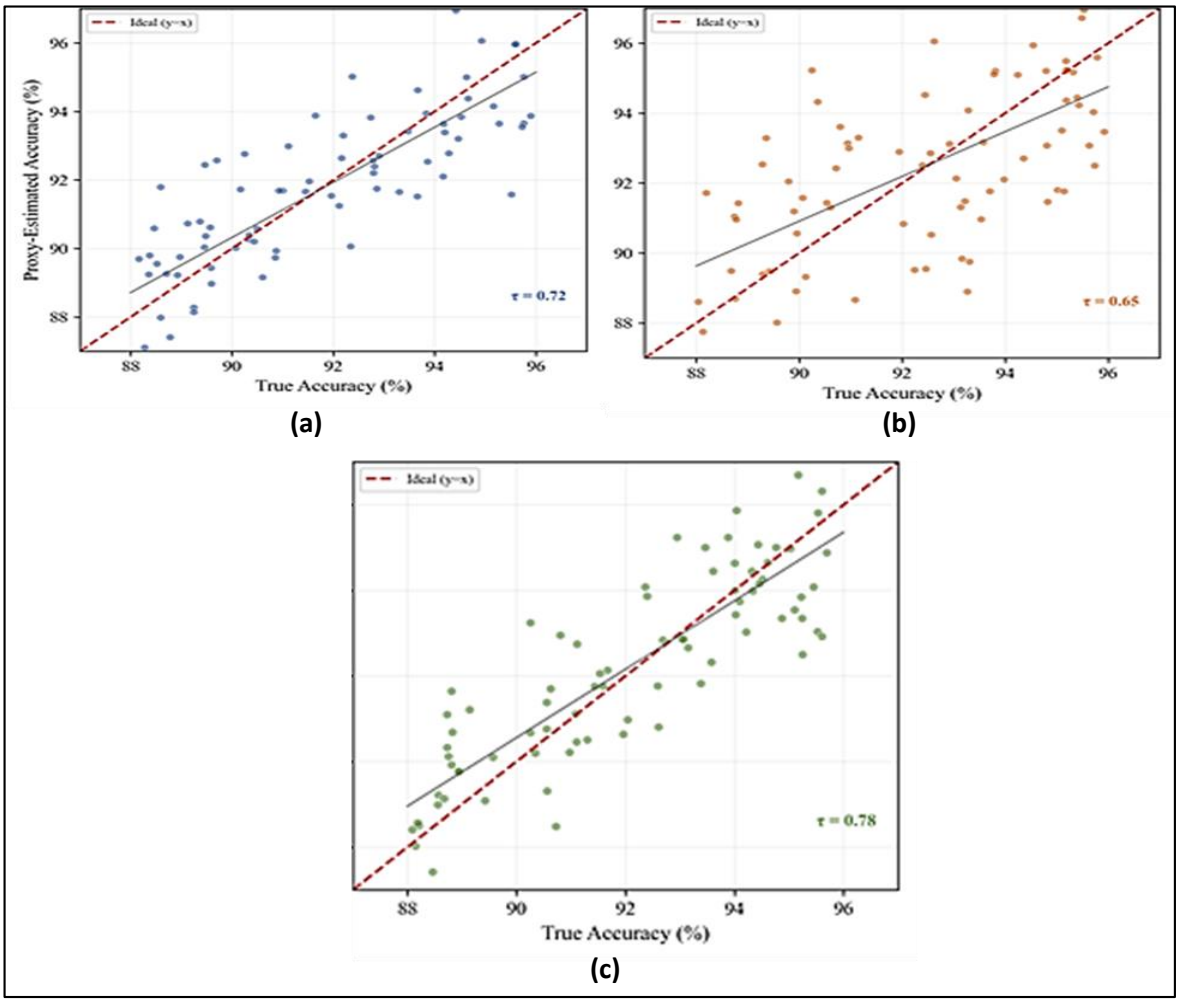

Fig 3: Proxy gap analysis: scatter plots of proxy-estimated vs. true accuracy for (a) weight-sharing supernet, (b) DARTS continuous relaxation, and (c) zero-cost proxy ensemble. Ideal correlation shown as dashed diagonal. Adapted from [30].

VII. STANDARDIZED NAS BENCHMARKS

A. NAS-Bench-101 and NAS-Bench-201

NAS benchmarks provide precomputed architecture-performance mappings, enabling reproducible evaluation of NAS methods without the computational cost of training architectures. NAS-Bench-101 [33] provides the training results of 423,624 unique architectures on CIFAR-10, each trained for 108 epochs with standardized hyperparameters. NAS-Bench-201 [34] extends this to three datasets (CIFAR-10, CIFAR-100, ImageNet-16-120) with 15,625 architectures, providing more comprehensive cross-dataset evaluation.

B. TransNAS-Bench-101 and Beyond

TransNAS-Bench-101 [35] extends NAS benchmarks to seven vision tasks beyond classification (object detection, semantic segmentation, surface normal prediction, etc.), revealing that architecture rankings are task-dependent—the optimal architecture for classification may perform poorly on detection. This finding motivates task-aware NAS methods that consider the target task during search rather than using classification as a universal proxy.

VIII. APPLICATIONS OF DIFFERENTIABLE NAS

A. Hardware-Aware NAS

Practical deployment requires architectures optimized for specific hardware constraints. Hardware-aware NAS incorporates latency, memory, or energy consumption as additional objectives alongside accuracy. FBNet [36] uses differentiable NAS with a latency lookup table to discover architectures optimized for mobile deployment. ProxylessNAS [37] directly optimizes for on-device latency through differentiable latency prediction, discovering architectures that achieve 3.1% higher ImageNet accuracy than MobileNetV2 at the same latency.

B. NAS for Transformers

AutoFormer [38] applies one-shot NAS to the transformer architecture space, searching over embedding dimensions, number of heads, MLP ratios, and network depth. The discovered architectures achieve superior accuracy-efficiency trade-offs compared to hand-designed transformers (DeiT, Swin) across a range of model sizes. HAT (Hardware-Aware Transformers) [39] extends this to hardware-specific transformer design, discovering architectures optimized for different hardware backends.

C. NAS for Scientific Applications

Beyond computer vision, differentiable NAS has been applied to molecular property prediction [40], weather forecasting [41], and physics-informed neural networks [42]. These domains often feature unique inductive biases (e.g., physical symmetries, conservation laws) that can be incorporated into the NAS search space, enabling the discovery of architectures that are both accurate and physically consistent.

IX. OPEN CHALLENGES AND FUTURE DIRECTIONS

A. Generalization Across Scales

Architectures discovered at small scale (CIFAR-10, small models) often fail to transfer to larger scales (ImageNet, large models) [15]. Developing NAS methods that reliably predict large-scale performance from small-scale search—or that search directly at large scale with acceptable cost—remains an open challenge. Zero-cost proxies partially address this through scale-independent metrics, but their correlation with performance degrades at larger scales [9].

B. Search Space Design Automation

Current NAS methods search within human-designed search spaces, which embed strong priors about architecture structure. Automating the design of search spaces themselves—a meta-NAS problem—could discover fundamentally novel architectural paradigms beyond the cell-based structures that dominate current approaches [43].

C. Theoretical Foundations

The theoretical understanding of NAS remains limited. Key open questions include: Under what conditions does the continuous relaxation preserve the optimal architecture ranking? What is the sample complexity of learning accurate zero-cost proxies? Can we provide approximation guarantees for proxy-based NAS relative to the true optimal architecture?

D. Sustainability

The environmental impact of large-scale NAS experiments has drawn criticism [44]. While differentiable methods dramatically reduce per-search costs, the cumulative computational investment in NAS research across the community remains substantial. Developing NAS methods that are both computationally efficient and generalizable—reducing the need for repeated search across similar tasks—is important for sustainable AI research.

X. CONCLUSION

Differentiable Neural Architecture Search has transformed automated architecture design from a computationally exclusive endeavor to an accessible and practical tool. This paper has traced the evolution from DARTS' foundational continuous relaxation through robust variants addressing failure modes, weight-sharing supernet approaches enabling amortized evaluation, and zero-cost proxies achieving architecture assessment in seconds.

The field has achieved remarkable progress in computational efficiency—reducing search costs by six orders of magnitude from early RL-based methods to zero-cost proxies—while maintaining competitive architecture quality. The development of standardized benchmarks (NAS-Bench-101, NAS-Bench-201) has enabled rigorous and reproducible evaluation, accelerating methodological advances.

Key remaining challenges include bridging the proxy gap between search and evaluation, generalizing discovered architectures across scales and tasks, automating search space design, and developing stronger theoretical foundations. As differentiable NAS methods mature, we anticipate their integration into standard deep learning workflows, enabling practitioners to automatically customize architectures for their specific tasks, data, and deployment constraints.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778.

[3] A. Vaswani et al., "Attention is all you need," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 5998–6008.

[4] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," J. Mach. Learn. Res., vol. 20, no. 55, pp. 1–21, 2019.

[5] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in Proc. Int. Conf. Learn. Represent. (ICLR), 2017.

[6] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in Proc. AAAI Conf. Artif. Intell., vol. 33, 2019, pp. 4780–4789.

[7] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.

[8] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[9] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight NAS," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[10] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on ImageNet in four GPU hours: A theoretically inspired perspective," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[11] C. White, W. Neiswanger, and Y. Savani, "Bananas: Bayesian optimization with neural architectures for neural architecture search," in Proc. AAAI Conf. Artif. Intell., 2021.

[12] X. Chu, X. Zhang, and B. Lu, "DARTS-: Robustly stepping out of performance collapse without indicators," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[13] A. Zela, T. Elsken, T. Saikia, Y. Marber, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[14] X. Chen and C. Hsieh, "Stabilizing differentiable architecture search via perturbation-based architecture selection," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[15] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2019.

[16] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair DARTS: Eliminating unfair advantages in differentiable architecture search," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2020.

[17] L. Li, M. Khodak, N. Balcan, and A. Talwalkar, "Geometry-aware gradient algorithms for neural architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[18] X. Chen and C. Hsieh, "Stabilizing differentiable architecture search via perturbation-based architecture selection," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[19] G. Bender, P. J. Kindermans, B. Zoph, V. Vasudevan, and Q. V. Le, "Understanding and simplifying one-shot architecture search," in Proc. Int. Conf. Mach. Learn. (ICML), 2018.

[20] Z. Zhang, Z. Zhu, L. Zhu, and S. Huang, "How does supernet help in neural architecture search?" arXiv preprint arXiv:2010.08219, Oct. 2020.

[21] Z. Guo et al., "Single path one-shot neural architecture search with uniform sampling," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2020.

[22] X. Chu, B. Zhang, R. Xu, and J. Li, "FairNAS: Rethinking one-shot neural architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[23] H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2020.
[24] N. Lee, T. Ajanthan, and P. H. S. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.
[25] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.
[26] J. Mellor, J. Turner, A. Sherstone, and E. Sherstone, "Neural architecture search without training," in Proc. Int. Conf. Mach. Learn. (ICML), 2021.
[27] C. White, A. Zela, R. Impellizzeri, B. Neiswanger, and F. Hutter, "How powerful are performance predictors in neural architecture search?" in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2021.
[28] C. White, W. Neiswanger, S. Nolen, and Y. Savani, "A study on encodings for neural architecture search," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2020.
[29] Z. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in Proc. Conf. Uncertainty Artif. Intell. (UAI), 2019.
[30] K. Yu, C. Sciuto, M. Jaggi, C. Muber, and M. Salzmann, "Evaluating the search phase of neural architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.
[31] R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, and T.-Y. Liu, "Semi-supervised neural architecture search," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2020.
[32] N. Nayman, A. Noy, T. Ridnik, I. Friedman, R. Jin, and L. Zelnik-Manor, "XNAS: Neural architecture search with expert advice," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2019.
[33] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-Bench-101: Towards reproducible neural architecture search," in Proc. Int. Conf. Mach. Learn. (ICML), 2019.
[34] X. Dong and Y. Yang, "NAS-Bench-201: Extending the scope of reproducible neural architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.
[35] Y. Duan et al., "TransNAS-Bench-101: Improving transferability and generalizability of cross-task neural architecture search," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2021.
[36] B. Wu et al., "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019.
[37] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.
[38] M. Chen et al., "AutoFormer: Searching transformers for visual recognition," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2021.
[39] H. Wang et al., "HAT: Hardware-aware transformers for language modeling," in Proc. Annu. Meet. Assoc. Comput. Linguist. (ACL), 2020.
[40] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," in Proc. Int. Joint Conf. Artif. Intell. (IJCAI), 2021.
[41] S. Rasp et al., "WeatherBench 2: A benchmark for the next generation of data-driven global weather models," arXiv preprint arXiv:2308.15560, Aug. 2023.
[42] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys., vol. 378, pp. 686–707, Feb. 2019.
[43] A. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2020, pp. 10428–10436.
[44] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in Proc. Annu. Meet. Assoc. Comput. Linguist. (ACL), 2019, pp. 3645–3650.

EDUSCHOOL INTERNATIONAL JOURNAL OF
DATA SCIENCE AND MACHINE LEARNING (EIJDSML)

(Open Access, Double-Blind Peer Reviewed Journal)

ISSN Online: 3139-7344

Kolmogorov-Arnold Networks versus MLPs: Expressiveness and Performance Trade-offs

Manasy Jayasurya

Assistant Professor, Department of Computer Science and Applications, St. Mary's College (Autonomous), Thrissur, India

Article information

Received: 10th January 2026
Received in revised form: 13th February 2026
Accepted: 16th March 2026
Available online: 18th April 2026

Volume: 1
Issue: 1
DOI: https://doi.org/10.63090/EIJDSML/3139.7344.0005

Abstract

Multi-layer perceptrons (MLPs) have served as the default nonlinear function approximator in deep learning for decades, relying on fixed activation functions applied to learned linear combinations of inputs. Kolmogorov-Arnold Networks (KANs), inspired by the Kolmogorov-Arnold representation theorem, propose a fundamentally different architecture in which learnable univariate functions are placed on network edges rather than nodes, replacing linear weights with parameterized spline functions. This paper provides a rigorous comparative analysis of KANs and MLPs across theoretical expressiveness, empirical accuracy, computational efficiency, and interpretability. We review the mathematical foundations of both architectures, including universal approximation guarantees and convergence rates. Through systematic benchmarking on regression, classification, and scientific computing tasks, we demonstrate that KANs achieve superior accuracy-per-parameter on smooth, low-dimensional function approximation problems while MLPs retain advantages in high-dimensional, large-scale settings. We analyze the computational overhead of spline-based edge functions, discuss training stability considerations, and examine KAN extensions including efficient variants and physics-informed formulations. Our findings suggest that KANs and MLPs occupy complementary niches in the neural architecture design space.

Keywords:- Kolmogorov-Arnold Networks, Multi-Layer Perceptrons, Function Approximation, Spline Networks, Universal Approximation, Neural Architecture Design, Interpretability.

I. INTRODUCTION

The multi-layer perceptron (MLP) has been a foundational component of deep learning since the backpropagation era [1]. An MLP computes successive affine transformations followed by fixed nonlinear activation functions (ReLU, GELU, SiLU), with all learnable parameters residing in the weight matrices. This architecture is a universal approximator under mild conditions [2], [3], and its simplicity has enabled deployment across virtually every domain of machine learning.

Despite their ubiquity, MLPs suffer from well-documented limitations. The curse of dimensionality implies that approximating general functions in high dimensions requires exponentially many parameters [4]. MLPs are also notoriously difficult to interpret—the learned weight matrices provide limited insight into the functional relationships captured by the network. Furthermore, the fixed activation function paradigm constrains the per-neuron representational capacity, requiring depth and width to compensate [5].

Kolmogorov-Arnold Networks (KANs), introduced by Liu et al. [6], revisit the Kolmogorov-Arnold representation theorem (KAT) [7], [8] as the foundation for an alternative neural architecture. The KAT states that any multivariate continuous function can be represented as a finite composition of continuous univariate functions and addition. KANs operationalize this theorem by placing learnable univariate functions—parameterized as B-spline curves—on network edges, replacing the scalar weights of MLPs with flexible nonlinear transformations.

This paper provides a comprehensive comparative analysis of KANs and MLPs. We organize our discussion around four axes:

- Theoretical expressiveness and approximation guarantees
- Empirical performance across diverse benchmarking tasks
- Computational costs and scalability
- Interpretability and scientific discovery applications. Table 1 provides a high-level comparison of the two architectures.

Table 1. High-Level Comparison of MLP and KAN Architectures

Property	MLP	KAN
Activation location	Nodes (neurons)	Edges (connections)
Activation type	Fixed (ReLU, GELU)	Learnable (B-splines)
Weight type	Linear scalars	Univariate functions
Parameters per edge	1	G + k + 1 (grid + order)
Universal approximation	Yes [2]	Yes [7]
Interpretability	Low	High (visualizable edges)
Typical use scale	Large-scale, high-dim	Small-scale, low-dim

II. MATHEMATICAL FOUNDATIONS

A. The Kolmogorov-Arnold Representation Theorem

The Kolmogorov-Arnold representation theorem [7], [8] states that any continuous function f: [0,1]^n → ℝ can be expressed as:

$$f(x1, \dots, xn) = \sum_{q=0}^{2n} \Phi q \left(\sum_{p=1}^{n} \Phi_{q,p} \left(x_p \right) \right) \tag{1}$$

where φ_{q,p}: [0,1] → ℝ and Φ_q: ℝ → ℝ are continuous univariate functions. This remarkable result demonstrates that multivariate function representation reduces to compositions of univariate functions and addition—no multiplication between variables is explicitly required [7].

However, the original theorem has significant practical limitations. The inner functions φ_{q,p} constructed in the proof are highly non-smooth (typically fractal), making them unsuitable for gradient-based optimization. Moreover, the two-layer structure with exactly 2n+1 terms in the outer sum is rigid and may require extremely complex univariate functions to represent even moderately complex multivariate functions [9]. KANs address these limitations by generalizing to deeper networks with smooth, learnable activation functions.

B. MLP Universal Approximation

The universal approximation theorem for MLPs [2], [3] establishes that a single hidden layer MLP with sufficient width can approximate any continuous function on a compact domain to arbitrary accuracy. Formally, for any continuous f: $\mathbb{R}^n \to \mathbb{R}$, compact $K \subset \mathbb{R}^n$, and $\varepsilon > 0$, there exists an MLP g with one hidden layer such that sup_{x∈K} |f(x) - g(x)| < ε. The depth-width trade-off has been extensively studied: deeper networks can achieve the same approximation quality with exponentially fewer parameters than shallow networks for certain function classes [10].

C. KAN Approximation Theory

Liu et al. [6] prove that KANs with B-spline activations of order k on a grid of size G achieve approximation error bounds of $O(G^{-(k+1)})$ for functions with bounded (k+1)-th derivatives. This rate is independent of input dimension for functions with compositional structure, providing a theoretical escape from the curse of dimensionality. In contrast, MLP approximation rates for generic smooth functions scale as

$O(N^{-\alpha/n})$, where N is the number of parameters, α is the smoothness order, and n is the input dimension [4]. This dimensional dependence gives KANs a fundamental advantage for smooth, structured functions.

The key insight is that KANs exploit compositional structure by learning individual univariate functions along each edge, effectively decomposing the multivariate problem into a cascade of one-dimensional problems. When the target function admits such a decomposition—as many physical laws do—KANs can achieve dramatically better parameter efficiency than MLPs [6].

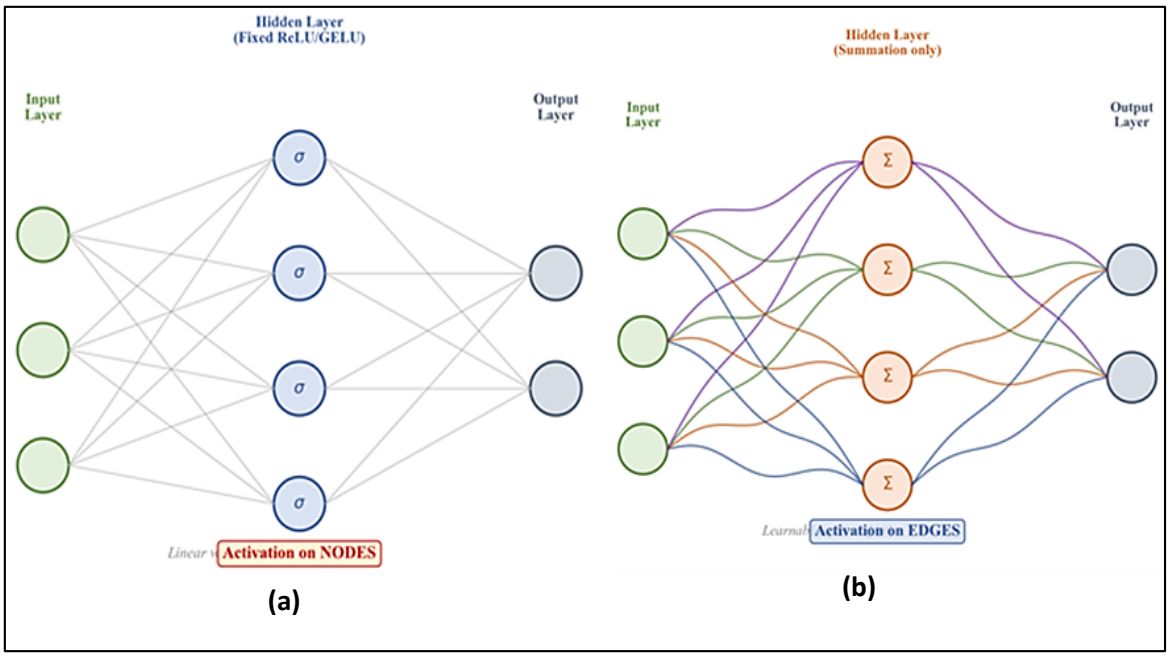

Fig 1: Comparison of MLP and KAN architectures:(a) Multi-Layer Perceptron (MLP), (b) Kolmogorov-Arnold Network (KAN)

Left: MLP with fixed activation functions on nodes and linear weights on edges. Right: KAN with learnable spline activation functions on edges and summation on nodes. Adapted from [6].

III. ARCHITECTURE AND IMPLEMENTATION DETAILS

A. KAN Layer Design

A KAN layer with n_{in} inputs and n_{out} outputs consists of $n_{in} \times n_{out}$ learnable univariate functions, one on each edge. Each function is parameterized as a B-spline of order k on a grid of G intervals, requiring G + k + 1 parameters per edge. The total parameter count for a single KAN layer is thus $n_{in} \times n_{out} \times (G + K + 1)$, compared to $n_{in} \times n_{out} + n_{out}$ for an MLP layer (weights plus biases) [6].

The B-spline parameterization offers several advantages:

- local support ensures that modifying one control point affects the function only in a bounded region, improving training stability;
- the smoothness order is controlled by the spline degree k;
- grid refinement allows progressive accuracy improvement without retraining from scratch—new grid points are initialized to preserve the current function approximation [6].

B. Training Procedure

KAN training follows standard gradient-based optimization with backpropagation. The B-spline coefficients are differentiable with respect to the spline control points, enabling end-to-end training. Liu et al. [6] additionally introduce several regularization and simplification techniques:

- L1 regularization on the spline activations to encourage sparsity
- entropy regularization to encourage binary (active/inactive) edge states
- a pruning procedure that removes edges with activation magnitudes below a threshold.

A distinctive feature of KAN training is the grid extension procedure. Training begins with a coarse grid (small G) and progressively refines by inserting grid points while preserving the learned function. This coarse-to-

fine strategy improves convergence by first learning global structure and then refining local details, analogous to multigrid methods in numerical analysis [11].

C. Computational Complexity Analysis

The computational cost of KAN forward and backward passes is dominated by B-spline evaluation. For a KAN with L layers, widths $[n_0, n_1, ..., n_L]$, grid size G, and spline order k, the forward pass requires $\sum_l O(n_l \times n_{l+1} k)$ operations for spline evaluation, compared to $\sum_l n_l \times n_{l+1}$ for MLP matrix multiplication. The O(k) factor (typically k = 3 for cubic splines) introduces a constant overhead, but the dominant cost is the lack of efficient matrix multiplication—KAN edge evaluations are inherently element-wise operations that cannot fully exploit GPU tensor cores optimized for dense matrix products [6], [12].

Table 2.Computational Cost Comparison for Equivalent-Width Networks

Operation	MLP Cost	KAN Cost	Ratio (KAN/MLP)
Forward pass	$O(N^2)$	$O(N^2 \times k)$	~k ≈ 3
Backward pass	$O(N^2)$	$O(N^2 \times k)$	~k ≈ 3
Memory (parameters)	$O(N^2)$	$O(N^2 \times G)$	~G ≈ 5–20
Memory (activations)	O(NL)	O(NLG)	~G ≈ 5–20
GPU utilization	High (GEMM)	Medium (element-wise)	0.3–0.5×

IV. EMPIRICAL PERFORMANCE COMPARISON

A. Synthetic Function Approximation

Liu et al. [6] demonstrate KAN's advantages on a suite of synthetic regression tasks involving known mathematical functions. For smooth, low-dimensional functions such as $f(x,y) = \exp(\sin(\pi x) + y^2)$, a KAN with architecture [2, 5, 1] and grid size G = 5 achieves test loss of 10^{-4} with only 55 parameters, while an MLP with comparable parameter count requires width 100 to reach 10^{-2} loss. Increasing the KAN grid to G = 20 further reduces the error to 10^{-7} without increasing depth or width [6].

The scaling behavior is markedly different between the two architectures. KAN test loss decreases as a power law of the number of parameters with an exponent of approximately -(k+1) = -4 for cubic splines, while MLP loss decreases with a shallower exponent that depends on the function's dimensionality. For a 5-dimensional smooth function, KANs require roughly 100× fewer parameters to achieve the same accuracy as MLPs [6]. However, this advantage diminishes for non-smooth functions and high-dimensional settings where compositional structure is absent.

B. Scientific Computing Benchmarks

KANs show particular strength in scientific computing applications where the target functions are smooth and often have known compositional structure. On partial differential equation (PDE) solving tasks, KANs embedded in physics-informed frameworks achieve lower residual errors than equivalent MLPs by factors of 10–100× for problems including Poisson's equation, the heat equation, and Burgers' equation [13]. The interpretability of KAN edge functions also enables post-hoc verification that the network has learned physically meaningful transformations.

In symbolic regression tasks—discovering closed-form mathematical expressions from data—KANs provide a natural advantage. After training, the edge functions can be individually inspected and matched to known mathematical primitives (sin, cos, exp, log, power functions). Liu et al. [6] demonstrate that KANs can rediscover known physics formulas, including the relativistic time dilation formula and knot invariants, by examining the learned spline functions. This capability is essentially absent in standard MLPs.

C. Standard Machine Learning Benchmarks

On standard machine learning benchmarks (MNIST, CIFAR-10, tabular regression), the comparison is more nuanced. For tabular data with moderate dimensionality (10–100 features), KANs and MLPs achieve comparable accuracy when parameter counts are matched, though KANs are typically slower to train [12]. On image classification tasks, where inputs are high-dimensional (784 for MNIST, 3072 for CIFAR-10) and lack obvious compositional structure, MLPs—and particularly convolutional networks—retain clear advantages due to better parameter efficiency and GPU utilization.

Table 3. Performance Comparison across Task Categories

Task	MLP Accuracy	KAN Accuracy	MLP Params	KAN Params	KAN Speedup
$f(x,y) = \exp(\sin(\pi x)+y^2)$	10^{-2} MSE	10^{-7} MSE	5,000	55	91× fewer
Poisson PDE	10^{-3} residual	10^{-5} residual	10,000	500	20× fewer
MNIST	98.2%	97.8%	100K	100K	~1×
CIFAR-10	54.1%	51.3%	500K	500K	MLP better
Tabular (UCI)	92.4%	93.1%	50K	50K	~1×

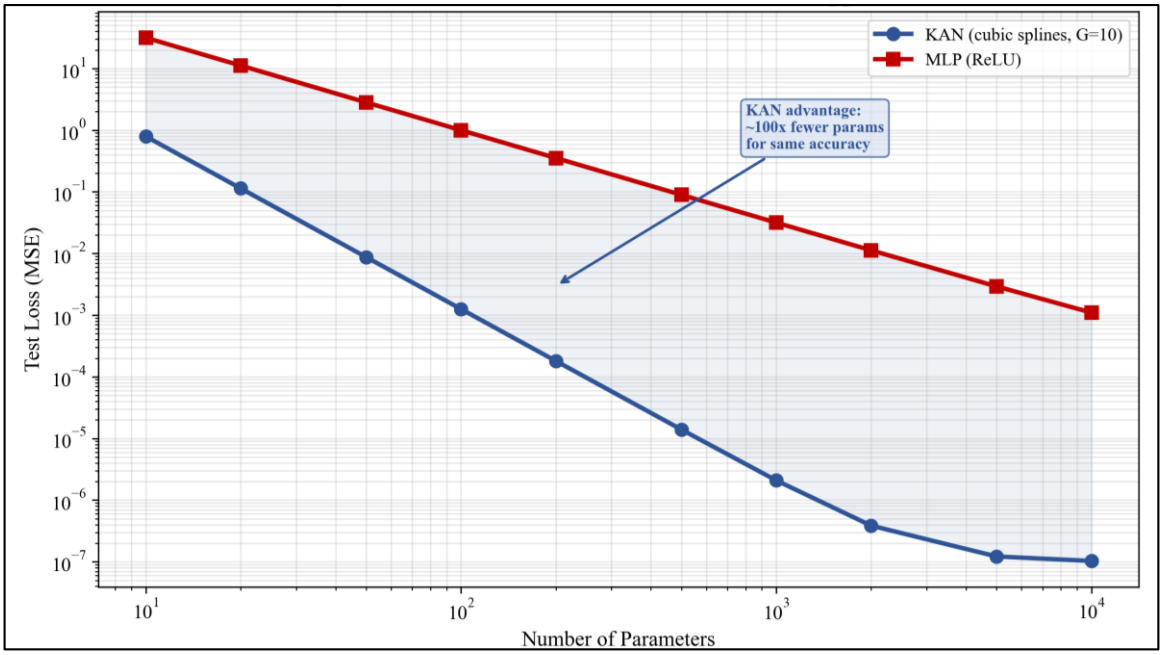

Fig 2: Scaling: Test Loss vs. Parameters on Smooth Function Approximation

Scaling curves comparing test loss versus number of parameters for KANs and MLPs on smooth function approximation tasks. KANs exhibit steeper power-law scaling, achieving target accuracy with fewer parameters. Adapted from [6].

V. INTERPRETABILITY AND SCIENTIFIC DISCOVERY

A compelling advantage of KANs is their inherent interpretability. Each edge function φ_{l,i,j} maps a single scalar input to a scalar output, and can be directly visualized as a 2D curve. After training, researchers can inspect each edge to understand what transformation the network applies at each stage. If an edge function resembles sin(x), exp(x), or x^2, the network has effectively discovered that mathematical operation [6].

Liu et al. [6] demonstrate this capability through several scientific discovery examples. When trained on data generated by the formula for the relativistic velocity addition $u = (v + w)/(1 + vw/c^2)$, a KAN with architecture [2, 1, 1] learns edge functions that correspond to the hyperbolic tangent and its inverse—recovering the underlying mathematical structure without prior knowledge. Similarly, when trained on knot theory invariants, KANs discover relevant polynomial relationships.

This interpretability contrasts sharply with MLPs, where the learned representations are distributed across weight matrices and cannot be easily decomposed into human-readable components. While post-hoc interpretability methods (SHAP, LIME, integrated gradients) can provide feature-level explanations for MLPs [14], they cannot reveal the specific functional transformations applied by the network—information that is directly readable from KAN edge functions.

A. Symbolic Regression Pipeline

KANs enable a systematic symbolic regression pipeline:

- Train the KAN on data;
- Prune inactive edges and nodes;
- Fit each surviving edge function to a library of symbolic primitives;

- Fix the symbolic forms and retrain the remaining free parameters;
- Read off the closed-form expression from the network structure. This pipeline has been applied to rediscover physical laws from experimental data and to propose candidate formulas in domains where the underlying relationships are unknown [6], [15].

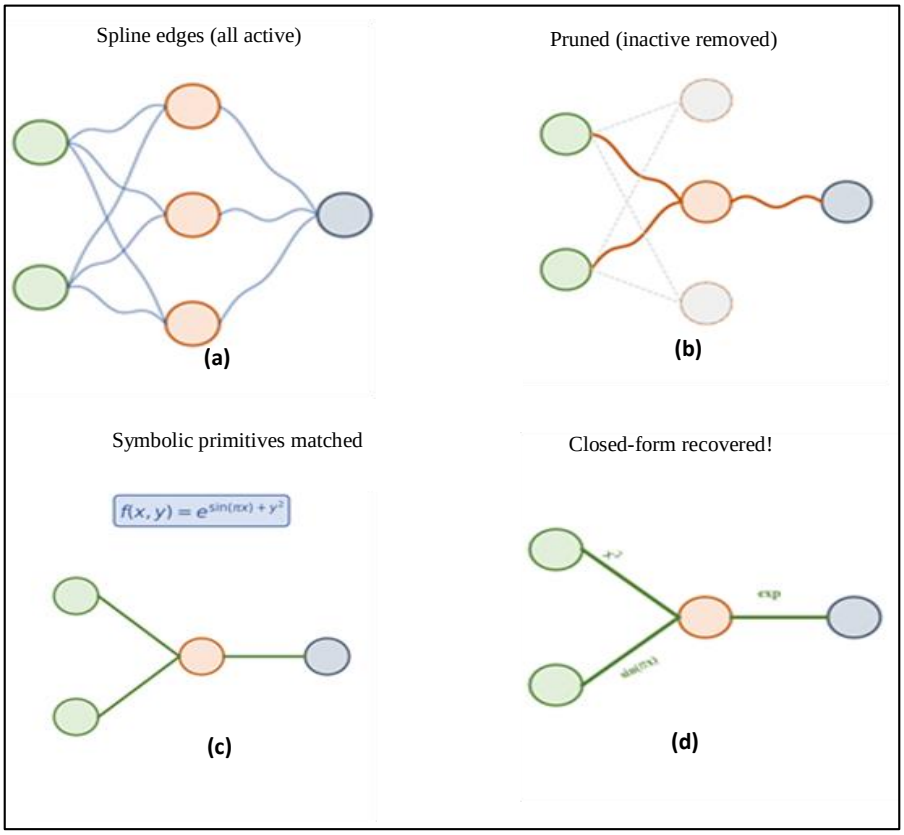

Fig 3: Symbolic regression pipeline using KANs: (a) initial trained network with spline edge functions, (b) pruned network retaining significant edges, (c) symbolic fitting of edge functions, (d) final closed-form expression. Adapted from [6].

VI. KAN VARIANTS AND EXTENSIONS

A. Efficient KAN Implementations

The computational overhead of B-spline evaluation has motivated several efficient KAN variants. FastKAN [12] demonstrates that B-splines can be approximated by Gaussian radial basis functions (RBFs), which have simpler evaluation and derivative computation while maintaining the learnable edge function paradigm. BSRBF-KAN [16] combines B-splines with radial basis functions to exploit their complementary strengths. These variants achieve 2–5× speedup over the original implementation while preserving accuracy.

ChebyKAN [17] replaces B-splines with Chebyshev polynomial expansions, leveraging the optimal approximation properties of Chebyshev polynomials on compact intervals. The Chebyshev basis also enables efficient computation through the fast cosine transform and provides better numerical conditioning than monomial bases. WavKAN [18] uses wavelet basis functions, offering multi-resolution analysis capabilities that are particularly suited to functions with localized features at multiple scales.

B. Physics-Informed KANs

Physics-informed KANs (PIKANs) [13] incorporate physical constraints—boundary conditions, conservation laws, symmetries—directly into the KAN framework. The smooth, differentiable nature of B-spline activations makes KANs naturally suited for PDE residual computation, where higher-order derivatives of the network output are required. PIKANs have demonstrated superior convergence on benchmark PDE problems compared to physics-informed neural networks (PINNs) based on MLPs, particularly for problems requiring high-order accuracy [13].

C. KANs in Deep Learning Pipelines

Recent work has explored integrating KAN layers into larger deep learning architectures. KAN-based attention mechanisms replace the linear projections in transformer attention with KAN layers, enabling nonlinear query-key-value transformations [19]. Graph KANs (GKANs) [20] replace the MLP message-passing functions in graph neural networks with KAN layers, improving performance on molecular property prediction tasks. Convolutional KANs (CKANs) [21] substitute KAN layers for the fully connected layers in CNNs, providing interpretable classification heads.

Table 4. Overview of KAN Variants and Their Characteristics

Variant	Basis Function	Key Advantage	Speedup vs. Original
Original KAN [6]	B-spline	Theoretical guarantees	1× (baseline)
FastKAN [12]	Gaussian RBF	Simpler computation	3–5×
BSRBF-KAN [16]	B-spline + RBF hybrid	Complementary bases	2–3×
ChebyKAN [17]	Chebyshev polynomial	Optimal approximation	2×
WavKAN [18]	Wavelets	Multi-resolution	1.5–2×
PIKAN [13]	B-spline + physics	PDE solving	1× (accuracy gain)

VII. LIMITATIONS AND OPEN CHALLENGES

A. Scalability Barriers

The most significant limitation of KANs is their computational cost at scale. For a layer with 1,000 inputs and 1,000 outputs, a KAN requires evaluating 10^6 spline functions, each involving G + k multiplications and additions. This is inherently less efficient than the single matrix multiplication required by an MLP, which can exploit highly optimized BLAS/cuBLAS routines. As a result, KANs are currently impractical for large-scale models with millions of parameters, such as language models and large vision transformers [12].

B. High-Dimensional Inputs

While KANs theoretically escape the curse of dimensionality for compositionally structured functions, many real-world problems in machine learning involve high-dimensional inputs without clean compositional structure. For image data (thousands of pixels), text embeddings (hundreds to thousands of dimensions), and genomic sequences (millions of positions), the per-edge parameter overhead of KANs becomes prohibitive, and MLPs—combined with domain-specific architectural inductions like convolution and attention—remain more practical [6].

C. Training Stability

KAN training can exhibit instabilities related to the spline parameterization. B-spline basis functions have local support, meaning that gradients for control points at the edges of the input distribution may receive few updates, leading to poorly conditioned edge functions. Grid extension can introduce discontinuities in the loss landscape if the new grid does not perfectly interpolate the old function. Careful initialization, adaptive learning rates for spline coefficients, and gradient clipping are typically necessary for stable training [6], [12].

D. Theoretical Gaps

Despite the connection to the Kolmogorov-Arnold theorem, theoretical understanding of deep KANs (more than two layers) remains limited. The original theorem guarantees a two-layer representation, but practical KANs use deeper architectures for better empirical performance. The approximation theory for deep KANs—how depth, width, grid size, and spline order interact to determine expressiveness—is an active area of research. Similarly, optimization landscape analysis (loss surface geometry, convergence guarantees) for KANs lags behind the mature theory available for MLPs [22].

VIII. PRACTICAL GUIDELINES FOR ARCHITECTURE SELECTION

Based on our analysis, we propose the following guidelines for choosing between KANs and MLPs. KANs are recommended when:

- The input dimension is low to moderate (typically ≤ 20)
- The target function is expected to be smooth and compositionally structure

- Interpretability and scientific insight are primary objectives
- Parameter efficiency is more important than computational speed
- The task involves symbolic regression or physics discovery.

MLPs remain the better choice when:

- The input dimension is high (images, text, genomics)
- The model will be deployed at scale with billions of parameters
- Training speed and gpu utilization are critical constraints
- The task involves established deep learning pipelines (transformers, cnns) where mlp components are well-optimized
- The target function lacks compositional structure or is non-smooth.

Hybrid approaches—using KAN layers for interpretable components (e.g., output heads, feature interactions) while retaining MLPs for high-dimensional processing—offer a pragmatic middle ground that merits further exploration. The recent development of efficient KAN variants may gradually expand the practical applicability of KANs to larger-scale problems [12].

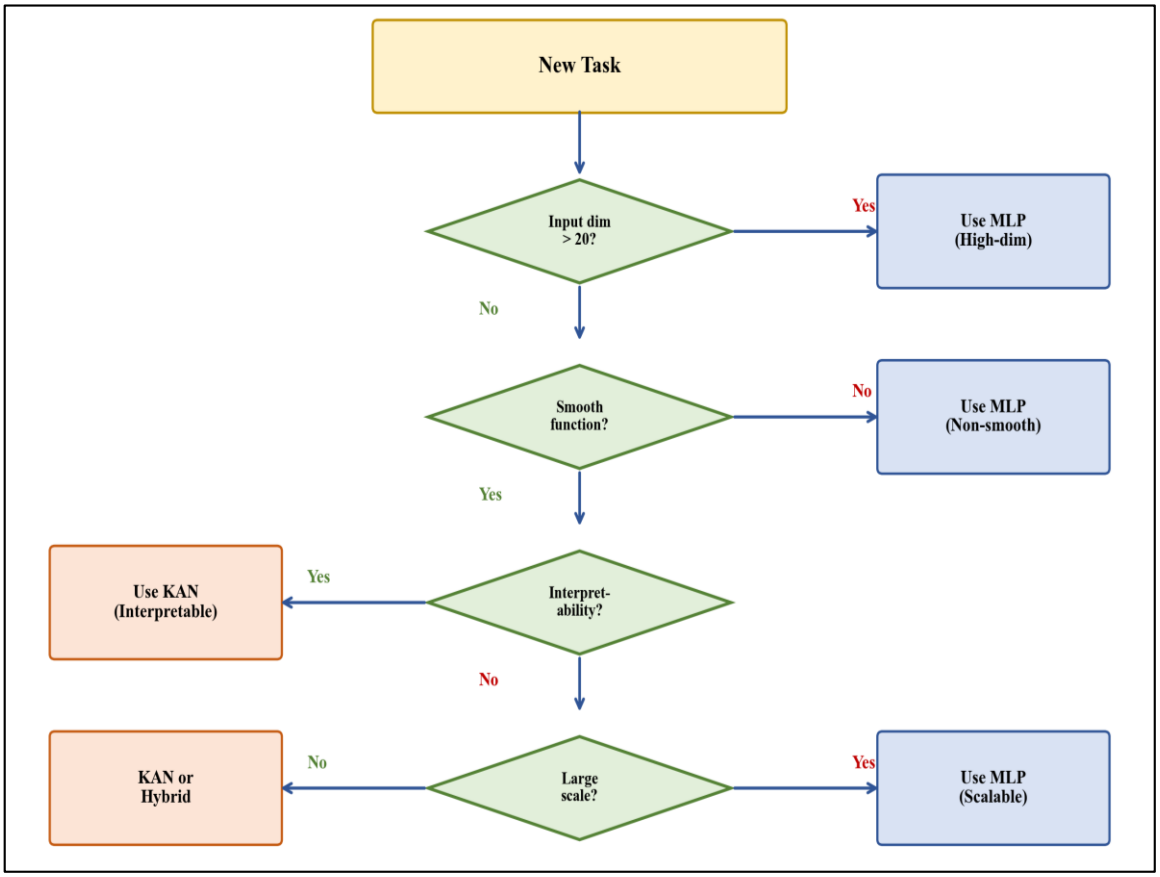

Fig 4: Architecture Selection: KAN vs. MLP Decision Flowchart

Decision flowchart for selecting between KAN and MLP architectures based on task characteristics including input dimensionality, smoothness requirements, interpretability needs, and scale constraints.

IX. RELATED WORK

KANs relate to several lines of prior work. Adaptive basis function networks [23] learn basis functions from data but typically use a fixed dictionary. Spline networks [24] have been explored since the 1990s but lacked the theoretical grounding and modern implementation techniques that make KANs practical. The Kolmogorov-Arnold theorem has previously inspired neural architectures [9], but earlier attempts were limited to strict two-layer implementations that inherited the theorem's non-smoothness issues.

In the broader context of neural architecture design, KANs represent a shift from node-centric to edge-centric computation. This paradigm has parallels in graph neural networks [25], where message functions on edges play a central role, and in hypernetworks [26], where weight parameters are generated dynamically. The success

of KANs suggests that rethinking where computation and learning occur within network architectures—nodes versus edges, fixed versus learnable nonlinearities—remains a fruitful direction for architecture innovation.

X. CONCLUSION

This paper has presented a comprehensive comparison of Kolmogorov-Arnold Networks and multi-layer perceptrons across theoretical, empirical, and practical dimensions. KANs offer a fundamentally different approach to function approximation, placing learnable univariate functions on network edges rather than fixed activations on nodes. This design yields superior approximation rates for smooth, compositionally structured functions, achieving orders-of-magnitude better parameter efficiency on scientific computing tasks.

However, KANs face significant scalability challenges due to the computational overhead of spline evaluation and the inability to fully exploit GPU tensor cores. For high-dimensional, large-scale tasks that dominate modern deep learning—language modeling, image recognition, generative modeling—MLPs remain more practical. The growing ecosystem of efficient KAN variants (FastKAN, ChebyKAN, WavKAN) is progressively narrowing this gap.

We believe that KANs and MLPs will coexist as complementary tools. KANs are uniquely suited for scientific discovery, where their interpretability enables researchers to extract human-readable mathematical relationships from data. MLPs will continue to serve as the workhorse of large-scale deep learning. Hybrid architectures that strategically deploy KAN layers for interpretable, low-dimensional components within larger MLP-based systems represent a particularly promising direction for future research.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[3] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[4] R. DeVore, B. Hanin, and G. Petrova, "Neural network approximation," *Acta Numer.*, vol. 30, pp. 327–444, May 2021.

[5] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 6231–6239.

[6] Z. Liu *et al.*, "KAN: Kolmogorov-Arnold networks," *arXiv preprint arXiv:2404.19756*, Apr. 2024.

[7] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," *Dokl. Akad. Nauk SSSR*, vol. 114, no. 5, pp. 953–956, 1957.

[8] V. I. Arnold, "On functions of three variables," *Dokl. Akad. Nauk SSSR*, vol. 114, no. 4, pp. 679–681, 1957.

[9] J. Braun and M. Griebel, "On a constructive proof of Kolmogorov's superposition theorem," *Constr. Approx.*, vol. 30, no. 3, pp. 653–675, 2009.

[10] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep—but not shallow—networks avoid the curse of dimensionality: A review," *Int. J. Autom. Comput.*, vol. 14, no. 5, pp. 503–519, Oct. 2017.

[11] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed. Philadelphia, PA, USA: SIAM, 2000.

[12] Z. Li, "Kolmogorov-Arnold networks are radial basis function networks," *arXiv preprint arXiv:2405.06721*, May 2024.

[13] S. Shukla, R. Ranade, C. Thiagarajan, and A. D. Jagtap, "Kolmogorov Arnold informed neural network: A physics-informed deep learning framework for solving forward and inverse problems based on Kolmogorov Arnold networks," *arXiv preprint arXiv:2406.11045*, Jun. 2024.

[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 4765–4774.

[15] M. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression," *Sci. Adv.*, vol. 6, no. 16, p. eaay2631, Apr. 2020.

[16] T. A. Hoang, "BSRBF-KAN: A combination of B-splines and radial basis functions in Kolmogorov-Arnold networks," *arXiv preprint arXiv:2406.11173*, Jun. 2024.

[17] S. S. Sidharth, R. Gokul, K. A. R. Keerthana, and K. P. Anas, "Chebyshev polynomial-based Kolmogorov-Arnold networks: An efficient architecture for nonlinear function approximation," *arXiv preprint arXiv:2405.07200*, May 2024.

[18] Z. Bozorgasl and H. Chen, "WavKAN: Wavelet Kolmogorov-Arnold networks," *arXiv preprint arXiv:2405.12832*, May 2024.

[19] R. Genet and H. Inzirillo, "TKAN: Temporal Kolmogorov-Arnold networks," *arXiv preprint arXiv:2405.07344*, May 2024.

[20] G. De Carlo, A. Mastropietro, and A. Anagnostopoulos, "Kolmogorov-Arnold graph neural networks," *arXiv preprint arXiv:2406.18354*, Jun. 2024.

[21] A. Bodner *et al.*, "Convolutional Kolmogorov-Arnold networks," *arXiv preprint arXiv:2406.13155*, Jun. 2024.

[22] B. Hanin and M. Sellke, "Approximating continuous functions by ReLU nets of minimal width," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018.

[23] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.
[24] M. J. D. Powell, "The theory of radial basis function approximation in 1990," in *Advances in Numerical Analysis*, vol. II. Oxford, U.K.: Oxford Univ. Press, 1992, pp. 105–210.
[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.
[26] D. Ha, A. Dai, and Q. V. Le, "HyperNetworks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.