NEURAL DARWINISM IN MULTIMODAL AI: ARCHITECTURES THAT EVOLVE TO LEARN

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern deep learning architectures, particularly Vision-Language Models (VLMs), have achieved remarkable success across multimodal tasks. However, these models are often constrained by manually engineered, static topologies—predefined architectural blueprints that limit adaptability, diversity, and evolutionary potential. This rigidity hampers their ability to generalize across domains, scale efficiently, and innovate beyond human design. In response, we present AI, Architect Thyself, a meta-learned evolutionary framework that enables neural networks to design, diversify, and evolve their own architectures. Unlike conventional neural architecture search or fixed multimodal blueprints, our approach treats topology as a dynamic learnable variable, optimized jointly with parameters. The system introduces three key innovations: (i) parametric plurality, where multiple instantiations of archetypes (Transformers, LSTMs, ResNets, Squeeze-and-Excite) coexist with distinct hyperparameters, (ii) a Graph Attention Router that performs per-sample expert routing across a dynamically evolving module zoo; and (iii) a co-evolutionary hybridization engine that recombines architectural traits of high-performing ancestors to generate novel configurations beyond human design. We trained with a self-growth strategist and replay memory, and the model exhibits strong adaptability with stability. On 12 multimodal and vision-language benchmarks, including Hateful Memes, VQA v2.0, COCO Captions, Food-101, and OpenImages, this framework consistently surpasses state-of-the-art baselines, achieving improvements of $+0.9\%$ to $+4.1\%$ across accuracy, AUC, and F1-Score. These results demonstrate that models can transition from being engineered artifacts to evolving organisms, advancing the frontier of autonomous machine intelligence.

1 INTRODUCTION

The design of neural architectures has historically relied on manual, trial-and-error exploration, demanding significant expertise and computational effort. Practitioners iteratively tune hyperparameters and evaluate static blueprints, a rigid process constrained by human intuition and resistant to adaptivity. Neural architecture search (NAS) emerged to automate this pipeline, yet remains bounded by predefined search spaces and static optimization strategies—whether reinforcement learning, evolutionary algorithms, or gradient-based methods—ultimately treating architecture as a fixed hyperparameter rather than a dynamic, learnable variable.

Despite progress, current NAS approaches face critical limitations. They depend on **constrained, human-engineered search spaces** that restrict discovery (Ouertatani et al., 2025; Lopes & Alexandre, 2025), and employ **computationally expensive evaluation strategies** requiring full training of candidate networks (Barradas-Palmeros et al., 2025; Xun et al., 2023). Moreover, search strategies are typically **static**, lacking mechanisms to adapt from prior learning Wang & Zhu (2024); Yang et al. (2021). Finally, existing methods fail to capture **parametric diversity**, overlooking the potential of multiple instantiations of architectural components with distinct hyperparameters Ouertatani et al. (2025); Lim & Kim (2022).

To overcome these limitations, we propose a fully autonomous neural framework that enables networks to self-architect, self-optimize, and continuously self-evolve. Unlike conventional NAS approaches constrained by static topologies, our system engages in a co-evolutionary process guided by a meta-cognitive controller that learns not only weights but also architectural principles. The **core intuition** is that by permitting a network to modify its own structure during training, it can discover novel, high-performing designs beyond human foresight. The controller observes which structural modifications enhance performance, internalizing design strategies from experience and enabling continuous refinement. To further boost specialization, the system maintains a team of neural modules, granting each—even repeated components such as transformers—unique internal configurations (e.g., attention heads or depth), allowing each module to master distinct subproblems. Here are the core challenges and our novel solutions presented:

- **Challenge 1: Static and Inefficient Inference**
  Conventional networks follow a fixed structure and computational path for all inputs, regardless of complexity.
  Our Solution: We introduce a **Graph Attention Router** that dynamically selects a data-dependent path through the network. Using learned attention, it activates only the most relevant expert modules per input, enabling context-sensitive and highly efficient inference.
- **Challenge 2: Limited Architectural Search Spaces**
  Standard NAS is restricted to predefined, human-engineered architectures, limiting creative potential.
  Our Solution: Our **Co-Evolution Engine** expands the search space via **biologically-inspired modular recombination**, generating entirely new architectures by intelligently combining high-performing features from existing modules.
- **Challenge 3: Generating Novel and Effective Architectures**
  Random mutations or naive search often yield inefficient designs.
  Our Solution: Through **Intelligent Hybridization**, the co-evolution engine identifies successful structural motifs and strategically cross-breeds them. This guided evolution accumulates "architectural wisdom," producing innovative and effective designs beyond the limits of human-constrained search spaces.

The central question we address in this paper is:

"Can a neural network learn to become its own architect, continuously evolving its internal structure to better master a task?"

Our key contributions are threefold:

- **A Framework for Autonomous Architectural Evolution**
  Instead of a static, manually-defined architecture, we introduced a **co-evolutionary hybridization engine** that allows the network to design itself. This process is guided by a **self-growth strategist** that learns effective evolutionary policies from a **replay memory** of successful past modifications. This intelligently recombines the structural traits and hyperparameters of high-performing "ancestor" networks to create entirely new and more effective modules. This transforms the network's topology from a fixed blueprint into a dynamic variable that is optimized jointly with the model's weights.
- **Parametric Plurality with Dynamic Expert Routing**
  We introduced the novel concept of **parametric plurality**, where the network builds and maintains a diverse "zoo" of specialized modules. Under this principle, even modules of the same type (e.g., multiple transformers or ResNets) are instantiated with unique hyperparameters, allowing each one to become an expert at a specific sub-task. To leverage this diversity, the Graph Attention Router dynamically selects the most suitable expert module(s) for each individual data sample, creating a unique and context
- **State-of-the-Art Performance and Architectural Discovery**
  We demonstrated the superiority of our framework through extensive experiments on 12 diverse multimodal and vision-language benchmarks, including challenging datasets like Hateful Memes, VQA v2.0, COC Captions, and Food-101. Our self-evolving model consistently outperforms state-of-the-art baselines, achieving signficant performance gains ranging from $+0.9\%$ to $+4.1\%$ across different metrics. Beyond these quantitative improvements, our analysis shows that the framework discovers novel and effective architectural motifs that were not engineered by humans, proving its capability for genuine automated design.

Table 1: Summary of Selected Research Works

Paper	Year	Venue	Core Contribution	Key Techniques	Challenges	Experimental Validation
Rahman et al. (2025)	2025	Nature Scientific Reports	LLM-guided neural architecture discovery without predefined search spaces	Expert system with LLM, configurable rules, multi-objective optimization	Computational efficiency concerns, dependency on LLM capabilities	CIFAR-10/100, ImageNet16-120
Wang et al. (2025)	2025	BDMA	Parameter disentanglement for diverse representations in neural networks	Parameter disentanglement, latent sub-parameters, lightweight refinement module	Computational overhead, limited theoretical guarantees	ImageNet, detection tasks
Junchi et al. (2025)	2025	Nature Scientific Reports	Multimodal attention network for emotion-cause pair extraction	BERT/Wav2Vec/ViT feature extraction, Graph Attention Networks, Transformer fusion	Dataset-specific performance, computational complexity	IEMOCAP, MELD, ECF, ConvECPE
Kim et al. (2025)	2025	CVPR	Cross-modality self-distillation for vision-language pre-training	Cross-attention module, text-cropping strategy, global/local view augmentation	Limited cross-modal alignment evaluation	Multimodal benchmarks
Li et al. (2025)	2024	IEEE TCSVT	Meta-learning framework for efficient EC-based NAS with adaptive surrogate models	Meta-learning rate scheme, adaptive surrogate model, period mutation operator	High computational cost, limited to specific network types	CIFAR-10/100, ImageNet1K
Joshi & Kokulavani (2025)	2024	SSRN	Self-evolving neural framework with meta-learning for continual learning	Adaptive neural weight modulation, neuro-symbolic evolution, self-referential architectures	Lacks comprehensive evaluation on large-scale benchmarks	Limited benchmark evaluation
Yang et al. (2024)	2024	CVPR	Multi-agent collaboration for neural architecture design using LLMs	Multi-agent collaboration, graph-based architecture representation, reflector mechanism	Limited scalability to very large architectures, dependency on LLM quality	Multiple benchmark datasets
Lim et al. (2023)	2024	ICLR	Graph neural networks for processing diverse neural architectures	Parameter graph representation, neural DAG automorphisms, GNN processing	Scalability to billion-parameter models, computational complexity	Multiple metanetwork tasks
Hu et al. (2024)	2024	ECAI	Cooperative coevolutionary reinforcement learning for scalable policy optimization	Cooperative coevolution, partial gradient search, proximal policy updates	Computational complexity, hyperparameter sensitivity	Six locomotion tasks

2 Related Work

Neural Architecture Search. Neural Architecture Search (NAS) automates network design, reducing reliance on manual trial-and-error J. Hao (2021). Early RL-based methods achieved strong performance but were computationally expensive Tang et al. (2021); Wang et al. (2024); Liu (2025). Gradient-based approaches like DARTS Liu et al. (2019) improved efficiency by relaxing discrete choices into continuous parameters Ma et al. (2024); Zhang et al. (2021); Huang et al. (2023), though they remain restricted by predefined search spaces and risk suboptimal convergence Mun et al. (2023); Cai et al. (2024). Recent work introduces multi-objective formulations balancing accuracy, latency, and model size, but still treats architectures as static hyperparameters requiring extensive evaluation Ding et al. (2022a).

Meta-Learning and Self-Adaptive Systems. Meta-learning extends automation to hyperparameter tuning and optimization, with methods like MAML and its variants enabling rapid adaptation across domains Killamsetty et al. (2022); Voon et al. (2024); Gai & Wang (2019); Antoniou et al. (2019). Recent work applies meta-learning to architecture adaptation Elsken et al. (2020); Lian et al. (2020); Ding et al. (2022b), though most approaches remain confined to incremental modifications within fixed search spaces. Self-organizing neural systems inspired by biological development dynamically rewire connectivity Fehérvári & Elmenreich (2014); Chakraborty & Chakrabarti (2015), yet current models largely depend on stochastic or handcrafted rules rather than learned decision policies Meyer et al. (2017); Ikeda et al. (2023); Li et al. (2021).

Dynamic Neural Networks and Mixture of Experts. Dynamic neural networks adapt computation graphs per input, improving efficiency and specialization Guo et al. (2025); Verma et al. (2024). Mixture-of-Experts (MoE) architectures leverage gating to route inputs to expert subnetworks, yielding state-of-the-art results across language and vision Antoniak et al. (2024); Alboody & Slama (2024); Chowdhury et al. (2024); Alboody & Slama (2025). Yet, most employ a fixed pool of experts, lacking mechanisms for evolving or pruning experts over time Abbasi et al. (2016);

Abbasi & Hooshmandasl (2021). Recent attention-based routers dynamically weight expert contributions He et al. (2022); Xu et al. (2022), but still fall short of enabling self-evolving expert sets Van Bolderik et al. (2024); Xu & McAuley (2023).

Table 1 summarizes recent progress at the intersection of neural architecture design, multimodal learning, and evolutionary/meta-learning frameworks. The surveyed works span leading venues such as Nature Scientific Reports, CVPR, ICLR, and IEEE journals, showcasing diverse strategies from LLM-guided architecture discovery Rahman et al. (2025); Yang et al. (2024) and parameter disentanglement Wang et al. (2025), to multimodal fusion with attention Junchi et al. (2025); Prabhu & Seethalakshmi (2025) and cross-modality self-distillation Kim et al. (2025). Despite notable advances, challenges of scalability, efficiency, dataset bias, and limited theoretical grounding persist. Evaluations across CIFAR, ImageNet, IEMOCAP, MELD, and large-scale navigation benchmarks highlight both the maturity of existing methods and the open gaps motivating our framework.

3 Problem Formulation and Architectural Framework

We cast our approach as a joint optimization problem over both **model parameters** and a **time-varying architecture**. Given a multimodal dataset $\mathcal{D} = \{(x_i^{(v)}, x_i^{(t)}, y_i)\}_{i=1}^{N}$, the model must learn to produce predictions $\hat{y}$ while simultaneously adapting its structure to improve efficiency and generalization.

At training step t, the system state is defined by the current architecture $\mathcal{A}_t$, which includes the **active modules** (drawn from our Neural Module Zoo), their **hyperparameter instantiations**, and the **Graph Attention Router** that determines how information flows among them. Standard network weights W_t are updated continuously by gradient descent, while the architecture $\mathcal{A}_t$ evolves episodically under the control of an **Evolutionary Strategist** π_ϕ. This meta-controller applies three classes of operations: *pruning* underperforming modules, *growing* new variants via hyperparameter mutation, and *hybridizing* promising parents to create offspring. Together, these yield a trajectory $\{\mathcal{A}_t\}_{t=0}^{T}$ rather than a fixed design.

A central concept is **parametric plurality**: instead of a single instantiation per archetype (e.g., ”the transformer block”), multiple variants are maintained in parallel, each with distinct hyperparameters. This allows the router to specialize modules for different input characteristics and prevents the system from collapsing prematurely onto a single inductive bias.

The learning objective combines the standard supervised loss (binary cross-entropy for multimodal classification) with additional terms that enforce **resource constraints** (e.g., parameter and FLOP budgets) and reward **diversity** across module instances. Formally, the strategist seeks architectures that minimize validation error while respecting cost bounds and maintaining pluralism. To stabilize learning under topology changes, a **replay memory** is used, preventing catastrophic forgetting when modules are removed or replaced.

In summary, the problem is formulated as a **bi-level optimization**:

- the inner loop optimizes weights W_t for the current architecture $\mathcal{A}_t$,
- the outer loop optimizes the strategist's policy π_ϕ, which governs the evolution of $\mathcal{A}_t$ itself.

This framing enables the system to “design itself” by coupling gradient-based learning with discrete architectural evolution.

3.1 Multimodal Feature Extraction

Given pair of multimodal inputs $(x^{(t)}, x^{(v)})$, where $x^{(t)} \in \mathcal{X}_t$ denotes textual tokens and $x^{(v)} \in \mathcal{X}_v$ denotes visual patches, we employed pretrained backbones: **DistilBERT** for text and **CLIP-ViT** for vision.

$$h^{(t)} = f_{DistilBERT}(x^{(t)}) \in \mathbb{R}^{L_t \times d_t}, \quad h^{(v)} = f_{CLIP-ViT}(x^{(v)}) \in \mathbb{R}^{L_v \times d_v} \tag{1}$$

where L_t and L_v denote sequence lengths, and d_t, d_v denote feature dimensions. To ensure cross-modal compatibility, we project both to a shared latent space $\mathbb{R}^d$ with $d = 512$:

$$z^{(t)} = W_t h^{(t)}, \quad z^{(v)} = W_v h^{(v)}, \quad W_t \in \mathbb{R}^{d \times d_t}, W_v \in \mathbb{R}^{d \times d_t}, \tag{2}$$

This yields modality-aligned embeddings $z^{(t)}, z^{(v)} \in \mathbb{R}^d$.

Novelty. Unlike prior works that treated pooled CLS tokens as unimodal anchors, our approach encodes both first-order (mean) and second-order (covariance) statistics, ensuring richer modality alignment. This statistical dual encoding produces embeddings that retain semantic consistency and structural diversity, which is critical when later routed into the Graph Attention Router (see subsection 3.5). Thus, the feature extraction stage is not only a preprocessing step but a statistically-grounded bridge that prepares multimodal signals for asymmetric cross-modal fusion (see subsection 3.2).

3.2 Cross-Modal Attention Fusion

A critical challenge in multimodal reasoning lies in fusing heterogeneous embeddings into a representation that preserves semantic complementarity while mitigating modality imbalance. We addressed this by introducing a **Multi-Head Cross-Modal Fusion (MHCMF)** mechanism with asymmetric query-key-value design, where vision serves as the query space and text serves as the key-value space. This reflects the intuition that textual tokens often provide grounding semantics, while vision queries those semantics for disambiguation.

$$Q = W_Q z^{(v)}, \quad K = W_K z^{(t)}, \quad V = W_V z^{(t)}. \tag{3}$$

The attention weights are computed as:

$$\alpha = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right), \quad z^{(f)} = \alpha V \tag{4}$$

where $z^{(f)} \in \mathbb{R}^d$ is the fused embedding. Then, multi-head extensions are applied:

$$z^{(f)} = \bigoplus_{m=1}^{H} z_m^{(f)}, \quad z_m^{(f)} = \alpha_m V_m, \tag{5}$$

ensuring cross-modal alignment and robustness to modality asymmetries.

Novelty. This fused embedding $z^{(cm)} \in \mathbb{R}^d$ becomes the interface variable for the Neural Module Zoo (see subsection E). Importantly, the asymmetric design preserves the interpretability with visual grounding driving the queries, and text supplies contextual semantics. The gating mechanism introduced here ensured balanced flow, preventing mode collapse to vision or text. And the multi-head structure produces diverse perspectives, which will be later exploited by Graph Attention Router (see subsection 3.5).

3.3 Neural Module Zoo and Dynamic Routing

Once we obtain the fused embedding, the next challenge is enabling the system to process this representation through a **diverse set of specialized transformations.** We introduce the **Neural Module Zoo $\mathcal{M}$,** a dynamic, extensible collection of neural operators. Unlike static ensembles, our zoo is evolutionary and parametric: each operator type admits multiple **parametric instantiations**, ensuring representational richness.

Given the fused representation $z^{(f)}$, each module produces a candidate transformation $u_j = m_j(z^{(f)}; \theta_j)$. The set of outputs u_j forms a pool of representations with complementary perspectives. This design transforms the zoo into a self-organizing ecosystem of operators, where diversity is preserved and expanded through evolutionary pressure, and relevance is determined by the Graph Attention Router.

Novelty. Unlike traditional static ensembles or MoE (Mixture of Experts), our **Neural Module Zoo** is

- **Parametrically plural** - multiple instantiations per operator family.
- **Evolutionary adaptive** - modules can be pruned, grown, or hybridized.
- **Routing-aware** - contributions are explicitly tracked via attention weights, forming the fitness signal for evolution.

This design creates a **self-organizing functional ecosystem** of operators, where diversity is not hand-designed but **emerges through evolutionary pressure** guided by the task objective.

3.4 Graph Attention Router (GAR) - Self Evolution Engine

The Graph Attention Router (GAR) is the central mechanism that (i) selects and composes expert module outputs on a per-sample basis, (ii) exposes a differentiable routing signal that trains both router and modules, and (iii) produces long-term contribution statistics used by the Evolutionary Strategist for pruning, growth, and hybridization. GAR departs from standard MoE routers by (a) combining **query** $\rightarrow$ **module relevance** with **module** $\rightarrow$ **module synergy** in a single attention mechanism, (b) supporting controlled sparsity (top-k routing) with differentiable approximations, and (c) emitting robust, temporally smoothed *fitness metrics* that serve as evolution signals.

Let the **fused multimodal embedding** be denoted as $h \in \mathbb{R}^d, \quad d = 512$, which acts both as the **query** and a global context signal. Each module $m_j \in \mathcal{M}$ produces an output representation $u_j = m_j(h), \quad u_j \in \mathbb{R}^d$, which is treated as the **value vector** in the routing mechanism. We parameterize the keys as learnable projections of the module outputs:

$$k_j = W_k u_j, \quad v_j = W_v u_j, \quad W_k, W_v \in \mathbb{R}^{d \times d} \tag{6}$$

The router computes **attention weights** by matching the fused embedding h against each key:

$$\alpha_j = \frac{\exp\left(\frac{(W_q h)^\top k_j}{\sqrt{d}}\right)}{\sum_{\ell=1}^{|\mathcal{M}|} \exp\left(\frac{(W_q h)^\top k_j}{\sqrt{d}}\right)}, \tag{7}$$

where $W_q \in \mathbb{R}^{d \times d}$ is the query projection. The **final routed representation** is a convex combination:

$$z = \sum_{j=1}^{|\mathcal{M}|} \alpha_j v_j. \tag{8}$$

Unlike standard mixture-of-experts routing, GAR is graph-aware: each module's contribution is recursively tracked over time and updated via a contribution score γ_j, which is incorporated into the attention computation by biasing the logits:

$$\alpha_j \propto \left(\frac{(W_q h)^\top k_j}{\sqrt{d}} + \lambda \gamma_j\right), \tag{9}$$

where λ regulates the strength of evolutionary feedback. This design enables the router not only to select modules adaptively per input, but also to provide evolutionary signals that guide the meta-controller in pruning, growing, and recombining modules.

3.5 Evolutionary Strategist — Meta-Controller for Structural Self-Growth

The **Evolutionary Strategist** is a meta-learning controller that dynamically modifies the Neural Module Zoo $\mathcal{M}$ during training, enabling the network to **prune, spawn, and hybridize** modules based on their contributions and diversity. It operates on **module genotypes** (architectural and hyperparameter specifications) and **phenotypes** (learned weights), with the goal of maximizing long-term validation performance while maintaining computational constraints.

Module Contribution and Fitness. Each module $m \in \mathcal{M}_t$ is assigned a **contribution score** combining:

1. **Router-based attention weight** $\bar{\beta}_m$ from the Graph Attention Router, capturing per-sample importance.
2. **Loss impact** $\bar{\Delta}\ell_m$, measuring validation loss change if the module is ablated.

These are combined via an exponential moving average:

$$C_m(t) \leftarrow (1-\rho)C_m(t-1) + \rho \left(w_\beta \frac{\bar{\beta}_m}{\max_k \bar{\beta}_k} + w_\ell \frac{\max(0, \bar{\Delta}\ell_m)}{\max_k \max(0, \bar{\Delta}\ell_m)} \right), \tag{10}$$

which serves as a fitness proxy for pruning and growth decisions. A **novelty term** encourages parametric diversity among modules.

Pruning and Growth.

- **Pruning**: Modules with consistently low fitness scores are removed, respecting a minimum-age threshold to ensure reliable contribution estimation.
- **Growth/Mutation**: New modules are spawned from high-fitness parents via hyperparameter perturbation:

$$\theta_c^{(i)} = \theta_p^{(i)} \cdot \exp(\sigma_\theta \cdot \epsilon^{(i)}), \quad \epsilon^{(i)} \sim \mathcal{N}(0,1), \tag{11}$$

with weights initialized via **soft inheritance**:

$$w_c = \gamma_{inh} w_p + (1-\gamma_{inh})\mathcal{N}(0, \sigma_w^2). \tag{12}$$

Hybridization. The **Co-Evolution Engine** recombines topological motifs from two parent modules m_i, m_j to produce children with novel architectures:

$$\theta_c^{(k)} = \lambda \theta_{m_i}^{(k)} + (1-\lambda)\theta_{m_j}^{(k)}, \quad \lambda \sim \mathcal{U}(0,1), \tag{13}$$

with weight inheritance for shared subgraphs and interface projection layers, ensuring compatibility. This allows the search space to expand beyond pre-defined topologies.

Controller Optimization. The strategist is trained via a reinforcement/meta-gradient objective, where the reward balances validation performance, computational cost, and architectural diversity:

$$r_t = \Delta\text{ValMetric} - \eta_{comp}\Delta\text{Cost} + \eta_{div}\overline{\text{novelty}}. \tag{14}$$

Actions are sampled from a learned policy $\pi_\phi(a_t|S_t)$, which determines when to prune, frow, or hybridize modules.

Stabilization. To maintain training stability, newly spawned or hybrid modules are warm-started with a small learning rate and replayed over a buffer of recent examples. Minimum-age constraints and EMA-based contribution tracking prevent oscillatory pruning or excessive growth.

Table 2: State-of-the-art comparison of classification tasks across multiple benchmarks. Best results are in **bold**.

Dataset	Model	Accuracy	AUC	F1	Precision	Recall	mAP	Improvement vs. SOTA
Hateful Memes	**Our Model**	**86.20%**	**0.9145**	0.8580	0.8620	0.8540	-	-
	SOTA (Dis-VLM)	84.10%	0.8930	-	-	-	-	Acc: +2.1%, AUC: +0.0215 Mei et al. (2025)
MM-IMDb	**Our Model**	**95.10%**	**0.9720**	**0.9485**	0.9510	0.9460	-	-
	SOTA (M3P)	93.30%	-	0.9330	-	-	-	Acc: +1.8%, F1: +1.55% Ni et al. (2021)
Food-101	**Our Model**	**94.20%**	**0.9850**	**0.9415**	0.9430	0.9400	-	-
	SOTA (PaLI-X 55B)	93.30%	-	-	-	-	-	Acc: +0.9% Chen et al. (2023)
OpenImages V6	**Our Model**	**92.50%**	**0.9680**	**0.9240**	0.9260	0.9220	**92.52%**	-
	SOTA (BEiT-3)	-	-	-	-	-	90.30%	mAP: +2.2% Wang et al. (2022)

4 Experiments

4.1 Dataset and Experimental Settings

We evaluated on **12 benchmark datasets,** which spanned a diverse set of multimodal reasoning tasks. This includes **Hateful Memes(10k)** Kiela et al. (2021), **MMIMDB (26K)** Jin et al. (2021), **Food-101 (101K)** Yu et al. (2024), **VQA v2.0 (444K)** Mi et al. (2024), **Conceptual Captions (CC) (3.3M)** Sharma et al. (2018), **COCO Captions (123K)** Lin et al. (2015), **Flickr30K (32K)** Young et al. (2014), **SentiCap (2.4K)** Mathews et al. (2015), **TextVQA (45K)** Singh et al. (2019), **VisualGenome (108K)** Krishna et al. (2017), **MSCOCO Detection (118K)** Lin et al. (2015), and **OpenImages (1.9M)** Kuznetsova et al. (2020).

All datasets are used under their respective licenses. We adopt the **official train/validation/test splits** and report results averaged over three random seeds. For **text**, inputs are tokenized with the DistilBERT WordPiece tokenizer (max length 128), with shorter sequences zero-padded and longer ones truncated. For **vision**, images are resized to 224×224 and normalized using ImageNet statistics; for detection tasks (MSCOCO, OpenImages), bounding-box annotations are preserved and cropped regions embedded.

Training Protocol. Models are trained for up to 25 epochs with early stopping (patience 5, validation AUC). Optimization uses AdamW (5×10^{-5} LR, $\beta_1 = 0.9,\ \beta_2 = 0.999$, weight decay 0.01), batch size 32, and a linear warmup over 10% of steps followed by cosine decay. The Neural Module Zoo is restricted to nine active modules, with evolutionary updates every three epochs. Dropout (0.1) is applied to text and vision embeddings, alongside L2 weight regularization.

Implementation Details. Our framework is implemented in PyTorch (v2.1), with HuggingFace Transformers for DistilBERT and TorchVision for CLIP-ViT. All experiments run on single NVIDIA A100 GPUs (80GB), with wall-clock times ranging from 2.5h (SentiCap) to 18h (Conceptual Captions). Reproducibility is ensured via fixed random seeds (Python, NumPy, PyTorch), deterministic GPU ops where available, and epoch-level checkpointing. The best model is selected based on validation AUC, and all code, pretrained weights, and logs will be released.

Evaluation Metrics. Task-specific metrics are reported: classification (Accuracy, AUC, F1, Precision, Recall) for Hateful Memes, MMIMDB, Food-101, and OpenImages; captioning (BLEU-4, METEOR, CIDEr, plus sentiment accuracy for SentiCap); VQA soft accuracy for VQA v2.0 and TextVQA; mean Average Precision (mAP, IoU 0.5–0.95) for MSCOCO and OpenImages detection; and Recall@K with classification accuracy for VisualGenome.

4.2 Comparison with State-of-the-Art

4.3 Cross-Dataset Generalization

4.4 Ablation Studies

4.5 Efficiency Analysis

5 Conclusion

References

E. Abbasi and M. R. Hooshmandasl. Semi-explicit mixture of experts based on information table. *Journal of Ambient Intelligence and Humanized Computing*, 14(7):8409–8420, November 2021. ISSN 1868-5145. doi: 10.1007/s12652-021-03607-w. URL http://dx.doi.org/10.1007/s12652-021-03607-w.

Elham Abbasi, Mohammad Ebrahim Shiri, and Mehdi Ghatee. A regularized root–quartic mixture of experts for complex classification problems. *Knowledge-Based Systems*, 110:98–109, October 2016. ISSN 0950-7051. doi: 10.1016/j.knosys.2016.07.018. URL http://dx.doi.org/10.1016/j.knosys.2016.07.018.

Ahed Alboody and Rim Slama. Ept-moe: Toward efficient parallel transformers with mixture-of-experts for 3d hand gesture recognition. In *Proceedings of the 10th World Congress on Electrical Engineering and Computer Systems and Science*, EECSS 2024. Avestia Publishing, August 2024. doi: 10.11159/mvml24.105. URL http://dx.doi.org/10.11159/mvml24.105.

Ahed Alboody and Rim Slama. *PMoET: Going Wider Than Deeper Using the Parallel Mixture of Experts Transformer for 3D Hand Gesture Recognition*, pp. 83–97. Springer Nature Switzerland, 2025. ISBN 9783031821561. doi: 10.1007/978-3-031-82156-1_7. URL http://dx.doi.org/10.1007/978-3-031-82156-1_7.

Szymon Antoniak, Michał Krutul, Maciej Pióro, Jakub Krajewski, Jan Ludziejewski, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Marek Cygan, and Sebastian Jaszczur. Mixture of tokens: Continuous moe through cross-example aggregation, 2024. URL https://arxiv.org/abs/2310.15961.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml, 2019. URL https://arxiv.org/abs/1810.09502.

Jesús-Arnulfo Barradas-Palmeros, Carlos-Alberto López-Herrera, Héctor-Gabriel Acosta-Mesa, and Efrén Mezura-Montes. *Efficient Neural Architecture Search: Computational Cost Reduction Mechanisms in DeepGA*, pp. 125–134. Springer Nature Switzerland, 2025. ISBN 9783031838828. doi: 10.1007/978-3-031-83882-8_12. URL http://dx.doi.org/10.1007/978-3-031-83882-8_12.

Zicheng Cai, Lei Chen, Peng Liu, Tongtao Ling, and Yutao Lai. Eg-nas: Neural architecture search with fast evolutionary exploration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(10):11159–11167, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i10.28993. URL http://dx.doi.org/10.1609/aaai.v38i10.28993.

Kabir Chakraborty and Abhijit Chakrabarti. *Classification of Voltage Security States Using Unsupervised ANNs*, pp. 153–173. Springer India, 2015. ISBN 9788132223078. doi: 10.1007/978-81-322-2307-8_7. URL http://dx.doi.org/10.1007/978-81-322-2307-8_7.

Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, Siamak Shakeri, Mostafa Dehghani, Daniel Salz, Mario Lucic, Michael Tschannen, Arsha Nagrani, Hexiang Hu, Mandar Joshi, Bo Pang, Ceslee Montgomery, Paulina Pietrzyk, Marvin Ritter, AJ Piergiovanni, Matthias Minderer, Filip Pavetic, Austin Waters, Gang Li, Ibrahim Alabdulmohsin, Lucas Beyer, Julien Amelot, Kenton Lee, Andreas Peter Steiner, Yang Li, Daniel Keysers, Anurag Arnab, Yuanzhong

Xu, Keran Rong, Alexander Kolesnikov, Mojtaba Seyedhosseini, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. Pali-x: On scaling up a multilingual vision and language model, 2023. URL https://arxiv.org/abs/2305.18565.

Mohammed Nowaz Rabbani Chowdhury, Meng Wang, Kaoutar El Maghraoui, Naigang Wang, Pin-Yu Chen, and Christopher Carothers. A provably effective method for pruning experts in fine-tuned sparse mixture-of-experts, 2024. URL https://arxiv.org/abs/2405.16646.

Yadong Ding, Yu Wu, Chengyue Huang, Siliang Tang, Fei Wu, Yi Yang, Wenwu Zhu, and Yueting Zhuang. Nap: Neural architecture search with pruning. *Neurocomputing*, 477:85–95, March 2022a. ISSN 0925-2312. doi: 10.1016/j.neucom.2021.12.002. URL http://dx.doi.org/10.1016/j.neucom.2021.12.002.

Yadong Ding, Yu Wu, Chengyue Huang, Siliang Tang, Yi Yang, Longhui Wei, Yueting Zhuang, and Qi Tian. Learning to learn by jointly optimizing neural architecture and weights. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 129–138. IEEE, June 2022b. doi: 10.1109/cvpr52688.2022.00023. URL http://dx.doi.org/10.1109/cvpr52688.2022.00023.

Thomas Elsken, Benedikt Staffler, Jan Hendrik Metzen, and Frank Hutter. Meta-learning of neural architectures for few-shot learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12362–12372. IEEE, June 2020. doi: 10.1109/cvpr42600.2020.01238. URL http://dx.doi.org/10.1109/cvpr42600.2020.01238.

István Fehérvári and Wilfried Elmenreich. *Evolution as a Tool to Design Self-organizing Systems*, pp. 139–144. Springer Berlin Heidelberg, 2014. ISBN 9783642541407. doi: 10.1007/978-3-642-54140-7_12. URL http://dx.doi.org/10.1007/978-3-642-54140-7_12.

Sibo Gai and Donglin Wang. Sparse model-agnostic meta-learning algorithm for few-shot learning. In *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)*, pp. 127–130. IEEE, September 2019. doi: 10.1109/cchi.2019.8901909. URL http://dx.doi.org/10.1109/cchi.2019.8901909.

Jifeng Guo, C. L. Philip Chen, Zhulin Liu, and Xixin Yang. Dynamic neural network structure: A review for its theories and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 36(3):4246–4266, March 2025. ISSN 2162-2388. doi: 10.1109/tnnls.2024.3377194. URL http://dx.doi.org/10.1109/tnnls.2024.3377194.

Xiaoyu He, Suixiang Shi, Xiulin Geng, and Lingyu Xu. Information-aware attention dynamic synergetic network for multivariate time series long-term forecasting. *Neurocomputing*, 500:143–154, August 2022. ISSN 0925-2312. doi: 10.1016/j.neucom.2022.04.124. URL http://dx.doi.org/10.1016/j.neucom.2022.04.124.

Chengpeng Hu, Jialin Liu, and Xin Yao. Evolutionary reinforcement learning via cooperative coevolution, 2024. URL https://arxiv.org/abs/2404.14763.

Lan Huang, Shiqi Sun, Jia Zeng, Wencong Wang, Wei Pang, and Kangping Wang. U-darts: Uniform-space differentiable architecture search. *Information Sciences*, 628:339–349, May 2023. ISSN 0020-0255. doi: 10.1016/j.ins.2023.01.129. URL http://dx.doi.org/10.1016/j.ins.2023.01.129.

Narumitsu Ikeda, Dai Akita, and Hirokazu Takahashi. Noise and spike-time-dependent plasticity drive self-organized criticality in spiking neural network: Toward neuromorphic computing. *Applied Physics Letters*, 123(2), July 2023. ISSN 1077-3118. doi: 10.1063/5.0152633. URL http://dx.doi.org/10.1063/5.0152633.

W. Zhu J. Hao. Architecture self-attention mechanism: nonlinear optimization for neural architecture search. *Journal of Nonlinear and Variational Analysis*, 5(1):119–140, 2021. ISSN 2560-6778. doi: 10.23952/jnva.5.2021.1.08. URL http://dx.doi.org/10.23952/jnva.5.2021.1.08.

Woojeong Jin, Maziar Sanjabi, Shaoliang Nie, Liang Tan, Xiang Ren, and Hamed Firooz. Modality-specific distillation. In Amir Zadeh, Louis-Philippe Morency, Paul Pu Liang, Candace Ross, Ruslan Salakhutdinov, Soujanya Poria, Erik Cambria, and Kelly Shi (eds.), *Proceedings of the Third Workshop on Multimodal Artificial Intelligence*, pp. 42–53, Mexico City, Mexico, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.maiworkshop-1.7. URL https://aclanthology.org/2021.maiworkshop-1.7/.

Kamal Kant Joshi and K. Kokulavani. Adaptive self-evolving neural networks a meta-learning approach for continual lifelong learning in dynamic environments. *SSRN Electronic Journal*, 2025. ISSN 1556-5068. doi: 10.2139/ssrn.5142382. URL http://dx.doi.org/10.2139/ssrn.5142382.

Ma Junchi, Hassan Nazeer Chaudhry, Farzana Kulsoom, Yang Guihua, Sajid Ullah Khan, Sujit Biswas, Zahid Ullah Khan, and Faheem Khan. Multicausenet temporal attention for multimodal emotion cause pair extraction. *Scientific Reports*, 15(1), June 2025. ISSN 2045-2322. doi: 10.1038/s41598-025-01221-w. URL http://dx.doi.org/10.1038/s41598-025-01221-w.

Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. The hateful memes challenge: Detecting hate speech in multimodal memes, 2021. URL https://arxiv.org/abs/2005.04790.

Krishnateja Killamsetty, Changbin Li, Chen Zhao, Feng Chen, and Rishabh Iyer. A nested bi-level optimization framework for robust few shot learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):7176–7184, June 2022. ISSN 2159-5399. doi: 10.1609/aaai.v36i7.20678. URL http://dx.doi.org/10.1609/aaai.v36i7.20678.

Sanghwan Kim, Rui Xiao, Mariana-Iuliana Georgescu, Stephan Alaniz, and Zeynep Akata. Cosmos: Cross-modality self-distillation for vision language pre-training, 2025. URL https://arxiv.org/abs/2412.01814.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73, 2017. doi: 10.1007/s11263-016-0981-7. URL https://doi.org/10.1007/s11263-016-0981-7.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision*, 128(7):1956–1981, March 2020. ISSN 1573-1405. doi: 10.1007/s11263-020-01316-z. URL http://dx.doi.org/10.1007/s11263-020-01316-z.

Tao Li, Kaijun Wu, Mingjun Yan, Zhengnan Liu, and Huan Zheng. Stochastic dynamic behavior of fitzhugh–nagumo neurons stimulated by white noise. *International Journal of Modern Physics B*, 35(10):2150137, April 2021. ISSN 1793-6578. doi: 10.1142/s021797922150137x. URL http://dx.doi.org/10.1142/s021797922150137x.

Yangyang Li, Guanlong Liu, Ronghua Shang, and Licheng Jiao. Meta knowledge assisted evolutionary neural architecture search, 2025. URL https://arxiv.org/abs/2504.21545.

Dongze Lian, Yin Zheng, Yintao Xu, Yanxiong Lu, Leyu Lin, Peilin Zhao, Junzhou Huang, and Shenghua Gao. Towards fast adaptation of neural architectures with meta learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1eowANFvr.

Derek Lim, Haggai Maron, Marc T. Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures, 2023. URL https://arxiv.org/abs/2312.04501.

Heechul Lim and Min-Soo Kim. Tenas: Using taylor expansion and channel-level skip connection for neural architecture search. *IEEE Access*, 10:84790–84798, 2022. ISSN 2169-3536. doi: 10.1109/access.2022.3195208. URL http://dx.doi.org/10.1109/access.2022.3195208.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2019. URL https://arxiv.org/abs/1806.09055.

Jiadong Liu. Comparative study of neural architecture search methods: Random search, rnn + reinforcement learning, and evolutionary algorithms on cifar-10. *IET Conference Proceedings*, 2025(2):32–37, April 2025. ISSN 2732-4494. doi: 10.1049/icp.2025.1009. URL http://dx.doi.org/10.1049/icp.2025.1009.

Vasco Lopes and Luís A. Alexandre. Toward less constrained macro-neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 36(2):2854–2868, February 2025. ISSN 2162-2388. doi: 10.1109/tnnls.2023.3326648. URL http://dx.doi.org/10.1109/tnnls.2023.3326648.

Benteng Ma, Yanning Zhang, and Yong Xia. Momentum recursive darts. *Pattern Recognition*, 156:110710, December 2024. ISSN 0031-3203. doi: 10.1016/j.patcog.2024.110710. URL http://dx.doi.org/10.1016/j.patcog.2024.110710.

Alexander Mathews, Lexing Xie, and Xuming He. Senticap: Generating image descriptions with sentiments, 2015. URL https://arxiv.org/abs/1510.01431.

Jingbiao Mei, Jinghong Chen, Guangyu Yang, Weizhe Lin, and Bill Byrne. Robust adaptation of large multimodal models for retrieval augmented hateful meme detection, 2025. URL https://arxiv.org/abs/2502.13061.

Bernd Meyer, Cedrick Ansorge, and Toshiyuki Nakagaki. The role of noise in self-organized decision making by the true slime mold physarum polycephalum. *PLOS ONE*, 12(3):e0172933, March 2017. ISSN 1932-6203. doi: 10.1371/journal.pone.0172933. URL http://dx.doi.org/10.1371/journal.pone.0172933.

Li Mi, Syrielle Montariol, Javiera Castillo-Navarro, Xianjie Dai, Antoine Bosselut, and Devis Tuia. Convqg: Contrastive visual question generation with multimodal guidance, 2024. URL https://arxiv.org/abs/2402.12846.

Jiwoo Mun, Seokhyeon Ha, and Jungwoo Lee. De-darts: Neural architecture search with dynamic exploration. *ICT Express*, 9(3):379–384, June 2023. ISSN 2405-9595. doi: 10.1016/j.icte.2022.04.005. URL http://dx.doi.org/10.1016/j.icte.2022.04.005.

Minheng Ni, Haoyang Huang, Lin Su, Edward Cui, Taroon Bharti, Lijuan Wang, Jianfeng Gao, Dongdong Zhang, and Nan Duan. M3p: Learning universal representations via multitask multilingual multimodal pre-training, 2021. URL https://arxiv.org/abs/2006.02635.

H. Ouertatani, C. Maxim, S. Niar, and E-G. Talbi. *Neural Architecture Tuning: A BO-Powered NAS Tool*, pp. 82–93. Springer Nature Switzerland, 2025. ISBN 9783031779411. doi: 10.1007/978-3-031-77941-1_7. URL http://dx.doi.org/10.1007/978-3-031-77941-1_7.

R. Prabhu and V. Seethalakshmi. A comprehensive framework for multi-modal hate speech detection in social media using deep learning. *Scientific Reports*, 15(1), April 2025. ISSN 2045-2322. doi: 10.1038/s41598-025-94069-z. URL http://dx.doi.org/10.1038/s41598-025-94069-z.

Md Hafizur Rahman, Zafaryab Haider, and Prabuddha Chakraborty. An automated multi parameter neural architecture discovery framework using chatgpt in the backend. *Scientific Reports*, 15(1), May 2025. ISSN 2045-2322. doi: 10.1038/s41598-025-97378-5. URL http://dx.doi.org/10.1038/s41598-025-97378-5.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of ACL*, 2018.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read, 2019. URL https://arxiv.org/abs/1904.08920.

Lang Tang, Huixia Li, Chenqian Yan, Xiawu Zheng, and Rongrong Ji. Survey on neural architecture search. *Journal of Image and Graphics*, 26(2):245–264, 2021. ISSN 1006-8961. doi: 10.11834/jig.200202. URL http://dx.doi.org/10.11834/jig.200202.

Bram Van Bolderik, Vlado Menkovski, Sonia Heemstra, and Manil Dev Gomony. Mean: Mixture-of-experts based neural receiver. In *2024 IFIP/IEEE 32nd International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–4. IEEE, October 2024. doi: 10.1109/vlsi-soc62099.2024.10767787. URL http://dx.doi.org/10.1109/vlsi-soc62099.2024.10767787.

Preeti Raj Verma, Navneet Pratap Singh, Deepika Pantola, and Xiaochun Cheng. Neural network developments: A detailed survey from static to dynamic models. *Computers and Electrical Engineering*, 120:109710, December 2024. ISSN 0045-7906. doi: 10.1016/j.compeleceng.2024.109710. URL http://dx.doi.org/10.1016/j.compeleceng.2024.109710.

Wingates Voon, Yan Chai Hum, Yee Kai Tee, Wun-She Yap, Khin Wee Lai, Humaira Nisar, and Hamam Mokayed. Imaml-idcg: Optimization-based meta-learning with imagenet feature reusing for few-shot invasive ductal carcinoma grading. *Expert Systems with Applications*, 257:124969, December 2024. ISSN 0957-4174. doi: 10.1016/j.eswa.2024.124969. URL http://dx.doi.org/10.1016/j.eswa.2024.124969.

Aili Wang, Kang Zhang, Haibin Wu, Shiyu Dai, Yuji Iwahori, and Xiaoyu Yu. Noise-disruption-inspired neural architecture search with spatial–spectral attention for hyperspectral image classification. *Remote Sensing*, 16(17):3123, August 2024. ISSN 2072-4292. doi: 10.3390/rs16173123. URL http://dx.doi.org/10.3390/rs16173123.

Jingxu Wang, Jingda Guo, Ruili Wang, Zhao Zhang, Liyong Fu, and Qiaolin Ye. Parameter disentanglement for diverse representations. *Big Data Mining and Analytics*, 8(3):606–623, 2025. doi: 10.26599/BDMA.2024.9020087.

Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, and Furu Wei. Image as a foreign language: Beit pretraining for all vision and vision-language tasks, 2022. URL https://arxiv.org/abs/2208.10442.

Xin Wang and Wenwu Zhu. Advances in neural architecture search. *National Science Review*, 11 (8), July 2024. ISSN 2053-714X. doi: 10.1093/nsr/nwae282. URL http://dx.doi.org/10.1093/nsr/nwae282.

Canwen Xu and Julian McAuley. A survey on dynamic neural networks for natural language processing, 2023. URL https://arxiv.org/abs/2202.07101.

Yunqiu Xu, Meng Fang, Ling Chen, Gangyan Xu, Yali Du, and Chengqi Zhang. Reinforcement learning with multiple relational attention for solving vehicle routing problems. *IEEE Transactions on Cybernetics*, 52(10):11107–11120, October 2022. ISSN 2168-2275. doi: 10.1109/tcyb.2021.3089179. URL http://dx.doi.org/10.1109/tcyb.2021.3089179.

Zhou Xun, Liu Songbai, Wong Ka-Chun, Lin Qiuzhen, and Tan Kaychen. A hybrid search method for accelerating convolutional neural architecture search. In *Proceedings of the 2023 15th International Conference on Machine Learning and Computing*, ICMLC 2023, pp. 177–182. ACM, February 2023. doi: 10.1145/3587716.3587745. URL http://dx.doi.org/10.1145/3587716.3587745.

Yibo Yang, Shan You, Hongyang Li, Fei Wang, Chen Qian, and Zhouchen Lin. Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6663–6672.

IEEE, June 2021. doi: 10.1109/cvpr46437.2021.00660. URL http://dx.doi.org/10.1109/cvpr46437.2021.00660.

Zekang Yang, Wang Zeng, Sheng Jin, Chen Qian, Ping Luo, and Wentao Liu. Nader: Neural architecture design via multi-agent collaboration, 2024. URL https://arxiv.org/abs/2412.19206.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

Xinyao Yu, Hao Sun, Ziwei Niu, Rui Qin, Zhenjia Bai, Yen-Wei Chen, and Lanfen Lin. Memory-inspired temporal prompt interaction for text-image classification, 2024. URL https://arxiv.org/abs/2401.14856.

Miao Zhang, Steven Su, Shirui Pan, Xiaojun Chang, Ehsan Abbasnejad, and Reza Haffari. idarts: Differentiable architecture search with stochastic implicit gradients, 2021. URL https://arxiv.org/abs/2106.10784.

APPENDIX

A DIFFERENCE BETWEEN TRADITIONAL NAS AND MALS

Traditional NAS iterates in outer loops, fully retraining candidate architectures between searches. Our proposed framework MALS here interleaves architectural updates with gradient updates using

- Micro-timescale (τ_g): Standard stochastic gradient descent updates the model weights.
- Macro-timescale (τ_a): Every k gradient steps, the Meta Controller executes an evolutionary adaptation event.

If $\tau_a \ll \tau_g$, the architecture can quickly adapt to novel data patterns without overfitting stale topologies. This dual time-scale formalism can be expressed as in Eq. 15.

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta_g \nabla_\theta \mathcal{L}_{task}(\theta_t, \mathcal{M}_t) \\ \mathcal{M}_{t+1} &= \mathcal{F}_{evolve}(\mathcal{M}_t, \pi_\theta, \mathcal{H}_t) \quad \text{only if} \quad t \mod k = 0 \end{aligned} \tag{15}$$

Here, θ represents module parameters, and $\mathcal{F}_{evolve}$ is the learned evolutionary update function.

B PROBLEM FORMULATION AND ARCHITECTURAL OVERVIEW

B.1 NOTATION AND CORE OBJECTS

Let

- $\mathcal{D} = \{(x_i^{(v)}, x_i^{(t)}, y_i)\}_{i=1}^N$ be the dataset of multimodal examples (visual, textual, label), drawn i.i.d. from an unknown distribution $\mathcal{P}_{data}$.
- d be the shared latent dimension (we use $d = 512$ in experiments).
- $\mathcal{A}_t$ denotes the **architecture state** at training step (or epoch) t. $\mathcal{A}_t$ comprises:
  - a set of active modules (the **Neural Module Zoo**) $\mathcal{M}_t = \{m_{t,1}, \dots, m_{t,N_t}\}$,
  - router parameters $\theta_t^{(r)}$,
  - module hyperparameter descriptors $\Theta_t = \{\theta_{t,1}, \dots, \theta_{t,1}\}$ (these describe structural choices like layers, heads, drouput, activation type),
  - global resource counters (parameter count, FLOPs).
- W_t denotes all learnable weights at step t: module weights, router weights, projection heads, classifier head, and any meta-controller weights (except where separated explicitly).
- π_ϕ denote the **Evolutionary Strategist** (meta-controller) parameterized by ϕ; it issues discrete/continuous actions that transform $\mathcal{A}_t \mapsto \mathcal{A}_{t+1}$/

A single forward pass on sample x under architecture $\mathcal{A}_t$ yields prediction $\hat{y}(x : W_t, \mathcal{A}_t)$. The per-sample task loss is $l(\hat{y}, y)$, e.g. cross-entropy.

Why this representation? Treating architecture as an explicit, time-indexed object $\mathcal{A}_t$ makes it possible to 1) reason about changes over training, 2) define budget constraints that vary over time, and 3) expose π_ϕ a state on which to condition actions — all necessary for principled co-evolution.

B.2 JOINT (BI-LEVEL) OPTIMIZATION: WEIGHTS AND TOPOLOGY

We designed this as a bi-level optimization where weights are optimized continuously while the meta-controller optimizes the architecture trajectory:

$$\begin{aligned} \text{(Outer / meta)} \quad & \min_\phi \mathbb{E}[\mathcal{L}_{val}(W_T(\phi), \mathcal{A}_T(\phi))] \\ \text{subject to} \quad & \mathcal{A}_{t+1} \sim \pi_\phi(\cdot|s_t), t = 0, \dots, T-1, \\ \text{(Inner/ weights)} \quad & W_{t+1} = \mathcal{U}(W_t, \nabla_{W_t} \mathcal{L}_{train}(W_t, \mathcal{A}_t)), \end{aligned} \tag{16}$$

where:

- $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ are empirical train and validation losses,
- $\mathcal{U}$ denotes the inner-loop optimizer (SGD/Adam step),
- s_t is the strategist state,
- expectations are over data sampling and any stochastic components of π_ϕ.

Why a bi-level view? Architecture decisions change the downstream loss landscape; optimizing ϕ requires evaluating the effect of architectural actions after weight updates. The bi-level view captures this causal dependency. Directly solving this exact bi-level problem is computationally intractable for large models, so we adopt approximations (meta-gradient, reward shaping, and fitness proxies) discussed below.

B.3 Parametric Plurality: Configuration Spaces and Module Instancing

We define an **archetype set** $\mathcal{T}$ (e.g., Transformer, LSTM, ResNet, MLP, Squeeze-Excite). For each archetype $a \in \mathcal{T}$, we defined a configuration (hyperparameter) space Ω_a. A module instance is then:

$$m = (a, \theta^{(arch)}, \omega), \quad \theta^{(arch)} \in \Omega_a, \omega = \text{learned weights} \tag{17}$$

We denoted the probability distribution over configurations as $P(\theta^{(arch)}|a)$ - the strategist can sample from or choose points in this space.

Parametric plurality means for a fixed archetype a, we allow multiple instances $\{m_i\}$ with different $\theta_i^{(arch)}$. Formally:

$$\mathcal{M}_t = \bigcup_{a \in \mathcal{T}} \{m_{t,i}^{(a)} : \theta_{t,i}^{(arch)} \sim P_t(\cdot|a)\}. \tag{18}$$

Why? Because of two main reasons, the first one being that multiple instantiations of the same structural bias with different internal hyperparameters produce distinct inductive priors and optimization dynamics. The second one is reducing reliance on a single optimum configuration for an archetype and enables per-sample specialization via the router.

Here, we quantify module diversity with a metric $\mathcal{D}(\mathcal{M}_t)$,

$$\mathcal{D}(\mathcal{M}_t) = \frac{1}{N_t^2} \sum_{i,j} \Delta(\theta_{t,i}^{(arch)}, \theta_{t,j}^{(arch)}) + \frac{1}{N_t^2} \sum_{i,j} \mathbb{E}_x ||u_{t,i}(x) - u_{t,j}(x)||_2, \tag{19}$$

where Δ measures configuration distance (mixed categorical/continuous) and the second term measures output diversity.

B.4 Router, Contribution, and the Strategist State

The Graph Attention Router (GAR) produces a per-sample distribution over modules:

$$\alpha(x; \mathcal{A}_t, W_t) = \text{GAR}(f(x; \mathcal{A}_t, W_t), \mathcal{M}_t) \in \Delta^{N_t - 1}, \tag{20}$$

and routed representation $z^{(r)} = \sum_m \alpha_m v_m$. To make an evolution decision, the strategist receives summary statistics (the **state** s_t) that include per-module fitness traces $\Phi_{t,m}$ (defined in 3.5), module utilization $\bar{\alpha}_{t,m}$, resource vector $c(\mathcal{A}_t)$ (parameter count, FLOPs, latency), global performance indicators and diversity $\mathcal{D}(\mathcal{M}_t)$.

Why these state features? They connect short-term routing behavior (utilization) with long-term utility (fitness), and expose resource constraints so π_ϕ can make capacity-aware decisions (prune low-utility modules, grow when capacity allows).

B.5 Evolutionary Operators

The strategist operates via a small set of operators that map architectures to architectures:

- **Prune operator** $\mathcal{P}_\tau$: We remove modules m with $\Phi_{t,m} < \tau_{prune}$ for $T_{patience}$ steps.
- **Mutate/Grow operator** $\mathcal{G}$**:** We sampled a parent m_p (probability proportional to positive fitness) and create child m_c by:

$$\theta_c^{(arch)} = \theta_p^{(arch)} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_{mut}^2), \tag{21}$$

  and initialize weights ω_c (either random or derived via partial weight inheritance).
- **Hybridization/Crossover operator** $\mathcal{H}$**:** For parents m_i, m_j, we selected proportional to fitness, and produced a child with mixed hyperparameters:

$$\theta_c^{(arch)} = \text{CROSS}(\theta_i^{(arch)}, \theta_j^{(arch)}), \tag{22}$$

  where CROSS handles continuous parameters by convex combination and categorical parameters by probabilistic selection or learned mapping (e.g., one parent chosen per categorical field with probability proportional to fitness).
- **Reinsertion/Assignment:** The newly created modules are inserted into $\mathcal{M}_t$ if resource budget permits, else they replace low-fitness modules.

The selection probabilities for parents are softmaxed fitness scores:

$$P(m_i, \text{chosen}) = \frac{\exp(\Phi_{t,i}/\tau_{sel})}{\sum_j \exp(\Phi_{t,i}/\tau_{sel})}. \tag{23}$$

Why these operators? They emulate biological mechanisms while remaining interpretable and tunable. Crossover blends complementary traits; mutation explores local neighbourhoods; pruning removes dead weight. Soft selection and patience thresholds prevent noisy immediate deletions.

B.6 Constraints and Resource-Aware Objective

Real systems operate under budgets. Let $C(\mathcal{A}_t)$ be a vector of costs (parameters, inference latency per sample, memory). The strategist must respect constraints $C(\mathcal{A}_t) \preceq C_{max}$. We embedded resource costs into the meta reward so the strategist optimizes utility under budgets. We defined the per-decision reward (to be maximized):

$$r_t = -\mathcal{L}_{val}(W_t, \mathcal{A}_t) - \lambda_c \cdot \text{cost}(C(\mathcal{A}_t)) + \lambda_d \mathcal{D}(\mathcal{M}_t), \tag{24}$$

where $\text{cost}(\cdot)$ aggregates resource usage into a scalar penalty and $\mathcal{D}(\cdot)$ is the diversity reward. The outer optimization becomes:

$$\max_\phi \mathbb{E}_{\pi_\phi} \left[\sum_{t=0}^{T-1} \gamma_{disc}^t r_t \right]. \tag{25}$$

Why reward shaping? Directly minimizing final validation loss is costly to estimate. A dense reward combining validation performance, resource penalties, and diversity fosters architectures that generalize, are efficient, and preserve pluralism.

Replay memory and stability Architecture changes introduce non-stationarity. To stabilize training, we maintain a replay buffer $\mathcal{R}$ storing representative samples (and their labels). When a module changes (spawned, hybridized), we interleave replay training on $\mathcal{R}$ to preserve past capabilities:

$$W_{t+1} \leftarrow \mathcal{U}(w_t, \nabla_{W_t}[\mathcal{L}_{train}(W_t, \mathcal{B}) + \mu \mathcal{L}_{replay}(W_t, \mathcal{R})]). \tag{26}$$

This is important as replay mitigates catastrophic forgetting when architecture topology changes and modules are inserted/removed. It also provides a stable baseline for computing module utility.

B.7 Practical approximations and Algorithmic summary

Solving the exact bi-level is impractical. Therefore, we adopted these approximations:

1. **Local fitness proxies:** We used $\Phi_{t,m}$ instead of full retraining-based evaluation for parent selection.
2. **Policy optimization:** We trained π_ϕ with reinforcement learning (PPO/actor-critic) using the dense reward r_t.
3. **Warm-start and patience:** We delayed pruning/hybridization for E_{warm} epochs to allow modules and router to stabilize.
4. **Deterministic operations at eval time:** We sparsified via deterministic top-K for reproducible inference.

Algorithmically. We alternated inner-loop updates of W_t (with replay) with occasional strategist decision steps that apply $\mathcal{P}, \mathcal{G}, \mathcal{H}$ based on Φ and s_t. The GAR provides per-sample routing and the contribution traces that ground evolutionary choices.

C Multimodal Feature Extraction

Let the input pair be $(x^{(t)}, x^{(v)})$, where $x^{(t)} \in \mathcal{X}$, denotes a sequence of text tokens and $x^{(v)} \in \mathcal{X}_v$ denotes an image decomposed into visual patches. Our objective is to map these heterogeneous modalities into a shared latent manifold $\mathcal{Z} \subseteq \mathbb{R}^d$, enabling subsequent cross-modal alignment and adaptive modular routing.

C.1 Textual Encoding

We tokenize the text sequence as

$$x^{(t)} = \{w_1, w_2, \ldots, w_{L_t}\}, \quad w_i \in \mathcal{V}, \tag{27}$$

where $\mathcal{V}$ is the vocabulary. A pretrained DistilBERT encoder $f_t : \mathcal{X}_t \in \mathbb{R}^{L_t \times d_t}$ produces contextualized embeddings:

$$h^{(t)} = f_t(x^{(t)}), \quad h^{(t)} = [h_1^{(t)}, h_2^{(t)}, \ldots, h_{L_t}^{(t)}], \quad h_i^{(t)} \in \mathbb{R}^{d_t}. \tag{28}$$

We applied a statistical pooling operator ϕ_t that preserves both mean and covariance structure:

$$\mu^{(t)} = \frac{1}{L_t} \sum_{i=1}^{L_t} h_i^{(t)}, \quad \sum^{(t)} = \frac{1}{L_t} \sum_{i=1}^{L_t} (h_i^{(t)} - \mu^{(t)})(h_i^{(t)} - \mu^{(t)})^\top \tag{29}$$

A low-rank factorization (Nyström approximation) compresses covariance into a vector:

$$c^{(t)} = \mathrm{vec}(U_k^\top \sum^{(t)} U_k), \quad U_k \in \mathbb{R}^{d_t \times k}. \tag{30}$$

Thus, the final text embedding is

$$z^{(t)} = W_t \begin{bmatrix} \mu^{(t)} \\ c^{(t)} \end{bmatrix}, \quad z^{(t)} \in \mathbb{R}^d \tag{31}$$

C.2 VISUAL ENCODING

We partition an image into L_v patches:

$$x^{(v)} = \{p_1, p_2, \ldots, p_{L_v}\}, \quad p_j \in \mathbb{R}^{h \times w \times c}. \tag{32}$$

A pretrained CLIP-ViT encoder $f_v : \mathcal{X}_v \to \mathbb{R}^{L_v \times d_v}$ yields patch embeddings using:

$$h^{(v)} = f_v(x^{(v)}), \quad h^{(v)} = [h_1^{(v)}, h_2^{(v)}, \ldots, h_{L_v}^{(v)}], \quad h_j^{(v)} \in \mathbb{R}^{d_v}. \tag{33}$$

Similar to the text above, we define

$$\mu^{(v)} = \frac{1}{L_v} \sum_{j=1}^{L_v} h_j^{(v)}, \quad \sum^{(v)} = \frac{1}{L_v} \sum_{j=1}^{L_v} (h_j^{(v)} - \mu^{(v)})(h_j^{(v)} - \mu^{(v)})^\top, \tag{34}$$

and compress via low-rank covariance embedding using

$$c^{(v)} = \text{vec}(U_k^\top \sum^{(v)} U_k). \tag{35}$$

The visual representation is then:

$$z^{(v)} = W_v \begin{bmatrix} \mu^{(v)} \\ c^{(v)} \end{bmatrix}, \quad z^{(v)} \in \mathbb{R}^d. \tag{36}$$

C.3 SHARED LATENT ALIGNMENT

Both the modalities are projected into the shared latent space $\mathcal{Z}$:

$$z^{(t)} = P_t(h^{(t)}), \quad z^{(v)} = P_v(h^{(v)}), \quad z^{(t)}, z^{(v)} \in \mathcal{Z}. \tag{37}$$

We enforce distributional proximity between $(z^{(t)}, z^{(v)})$ using a contrastive alignment term:

$$\mathcal{L}_{align} = -\log \frac{\exp(\text{sim}(z^{(t)}, z^{(v)})/\tau)}{\sum_{(z^{(t)}, z^{(v')})} \exp(\text{sim}(z^{(t)}, z^{(v)})/\tau)}, \tag{38}$$

where $\text{sim}(\cdot, \cdot)$ is cosine similarity and τ a temparature parameter.

D CROSS-MODAL ATTENTION FUSION

From the above, we obtain projected embeddings as $z^{(t)} \in \mathbb{R}^{L_t \times d}, \quad z^{(v)} \in \mathbb{R}^{L_v \times d}$, where $d = 512$. We construct modality-specific query, key, and value matrices, as:

$$Q^{(v)} = z^{(v)} W_Q^{(v)}, K^{(t)} = z^{(t)} W_K^{(t)}, V^{(t)} = z^{(t)} W_V^{(t)}, \tag{39}$$

with $W_Q^{(v)}, W_K^{(t)}, W_V^{(t)} \in \mathbb{R}^{d \times d}$. Next, we defined cross-modal attention from vision to text as:

$$\alpha = \text{softmax}\left(\frac{Q^{(v)} (K^{(t)})^\top}{\sqrt{d}} \right) \in \mathbb{R}^{L_v \times L_t}. \tag{40}$$

Here, each visual token attends to all textual tokens, producing fused representations as $z^{(f)} = \alpha V^{(t)} \in \mathbb{R}^{L_v \times d}$. Unlike symmetric co-attention, this asymmetric scheme ensures that visual grounding is enriched by linguistic semantics while avoiding representation dilution from treating both modalities equivalently. For robustness, we extended to a multi-head formulation using

$$z^{(f)} = \bigoplus_{h=1}^{H} z_h^{(f)}, \quad z_h^{(f)} = \alpha_h V_h^{(t)}, \tag{41}$$

Thus, the fused representation is a concatenation of head-specific semantic refinements. To prevent dominance of either modality, we introduced a modality gating mechanism. The scalar gate here is defined as:

$$g = \sigma(w^\top[\text{mean}(z^{(v)}), \text{mean}(z^{(t)})], \tag{42}$$

where $g \in (0, 1)$. The final fusion is a convex combination:

$$z^{(cm)} = g \cdot \text{mean}(z^{(f)}) + (1 - g) \cdot \text{mean}(z^{(t)}). \tag{43}$$

This adaptive gate balances contributions from visual-grounded fusion and raw textual semantics, ensuring stable cross-modal alignment.

E NEURAL MODULE ZOO AND DYNAMIC ROUTING

Formal definition. Let the zoo at time t contain M_t active modules:

$$\mathcal{M}_t = \{m_1(\cdot; \theta_1), m_2(\cdot; \theta_2), \ldots, m_{M_t}(\cdot; \theta_{M_t})\}. \tag{44}$$

Each module is a parametric function $m_j : \mathbb{R}^d \to \mathbb{R}^d, m_j(z^{(f)}; \theta_j) = u_j$, where $u_j \in \mathbb{R}^d$ is the output embedding from module j. Thus, given $z^{(f)}$, the zoo produces a candidate set of transformed representations: $U = [u_1, u_2, \ldots, u_{M_t}]^\top \in \mathbb{R}^{M_t \times d}$.

Module Families. The zoo supports multiple **operator families**, each corresponding to distinct inductive biases:

- **MLP modules** (dense projections):

$$m_{MLP}(z^{(f)}; \theta) = \sigma(W_2 \phi(W_1 z^{(f)} + b_1) + b_2), \tag{45}$$

  where $\phi(\cdot)$ is ReLU or GeLU, and $\sigma(\cdot)$ is a nonlinearity or identity.
- **Transformer modules** (contextual reasoning):

$$m_{Trans}(z^{(f)}; \theta) = \text{MHA}(z^{(f)}) + \text{FFN}(z^{(f)}), \tag{46}$$

  where MHA denotes the multi-head attention over $z^{(f)}$.
- **LSTM modules** (sequential bias):

$$h_t, c_t = \text{LSTM}(z^{(f)}, h_{t-1}, c_{t-1}; \theta). \tag{47}$$

- **ResNet-style modules**(residual feature refinement):

$$m_{Res}(z^{(f)}; \theta) = z^{(f)} + F(z^{(f)}; \theta), \tag{48}$$

  where F is a stack of nonlinear layers.
- **Squeeze-and-Excitation modules** (channel re-weighting):

$$m_{SE}(z^{(f)}; \theta) = z^{(f)} \odot \sigma(W_2 \phi(W_1 \text{pool}(z^{(f)}))). \tag{49}$$

These families are not fixed, and new families may be introduced during evolution (see subsection 3.5). Next, to encourage **structural diversity**, each module type admits multiple instantiations with distinct hyperparameters using $\theta_j = \{W_j, b_j, \alpha_j, \dots\}$, where α_j represents hyperparameters such as hidden width, number of layers, or dropout rate. Let Ω denote the hyperparameter configuration space. Then for a module family $\mathcal{F}$:

$$\{m(\cdot; \theta^{(1)}), m(\cdot; \theta^{(2)}), \dots\}, \quad \theta^{(k)} \sim \Omega. \tag{50}$$

This ensures that even within the same operator family, modules exhibit functional non-redundancy, avoiding collapse into homogeneous transformations.

Theoretical Motivation. Given $z^{(f)}$, an optimal transformation is not known *a priori*. The zoo, therefore, acts as a **basis expansion** of nonlinear operators, where the router learns convex combinations:

$$z^{(r)} = \sum_{j=1}^{M_t} \beta_j m_j(z^{(f)}; \theta_j), \quad \beta_j \geq 0, \sum_j \beta_j = 1. \tag{51}$$

This setup can be viewed as a **functional mixture model**:

$$\mathcal{F}(z^{(f)}) \approx \sum_{j=1}^{M_t} \beta_j m_j(z^{(f)}; \theta_j). \tag{52}$$

By evolving $\mathcal{M}_t$, the model dynamically expands the representational capacity, while the router ensures sparse and efficient selection.

F Graph Attention Router

Notations and Inputs. Let the Neural Module zoo previously at time t contain N active modules $\mathcal{M}_t = \{m_1, m_2, \dots, m_N\}$. For a single input sample (or a batch handled elementwise), we denoted the fused embedding (router query) as $\mathbf{f} \in \mathbb{R}^d$ (from subsection 3.2), and module outputs as $\mathbf{u}_m \in \mathbb{R}^d$ for $m = 1 \dots N$. We stacked them into $U = [\mathbf{u}_1; \dots; \mathbf{u}_N] \in \mathbb{R}^{N \times d}$. Next, we implemented multi-head attention with H heads; index head by h. Each head uses projection matrices $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)} \in \mathbb{R}^{d_h \times d}$ with $d_h = d/H$.

Headwise compatibility: relevance + synergy. For head h, we computed

$$\mathbf{q}^{(h)} = W_Q^{(h)} \mathbf{f}, \quad \mathbf{k}_m^{(h)} = W_K^{(h)} \mathbf{u}_m, \quad \mathbf{v}_m^{(h)} = W_V^{(h)} \mathbf{u}_m. \tag{53}$$

We defined two components for the per-module compatibility score:

1. **query-to-module relevance** (standard scaled dot-product):

$$r_m^{(h)} = \frac{\langle \mathbf{q}^{(h)}, \mathbf{k}_m^{(h)} \rangle}{\sqrt{d_h}}. \tag{54}$$

2. **module-synergy score** that captures how module m complements other modules for this input. We compute a learned module affinity via scaled dot-products on keys:

$$S_{m,j}^{(h)} = \frac{\langle \mathbf{k}_m^{(h)}, \mathbf{k}_j^{(h)} \rangle}{\sqrt{d_h}} \quad (j = 1 \dots N). \tag{55}$$

Algorithm 1 GAR Forward & Bookkeeping (per batch)

Require: Fused embeddings $\{f^{(i)}\}_{=1}^{B}$, module outputs $U^{(i)}$, router params θ_r, module params $\{\theta_m\}$
Ensure: Router outputs $\{z^{(r,i)}\}$, updated running stats $\{\bar{\alpha}, \Phi\}$

```
for each sample i do
    for each head h do
        Compute q^(h) = W_Q^(h) f^(i), k_m^(h) = W_K^(h) u_m^(i), v_m^(h) = W_V^(h) u_m^(i)
    end for
    Compute r_m^(h) = <q^(h), k_m^(h)> / sqrt(d_h)
    Compute S_{m,j}^(h) and s_m^(h) = r_m^(h) + γ^(h) · Σ_j softmax(S_{m,*}^(h)) · q_int(S_{m,j}^(h))
    α_m^(h) = softmax_m(s_m^(h)); aggregate α_m over heads → α_m
    Optionally sparsify α → α̃ (sparsemax or top-K)
    z^(r,i) = Σ_m α̃_m · v_m^agg
end for
Compute task loss L_task using {z^(r,i)}
Compute router regularizers L_ent, L_load, L_budget
Backprop: update θ_r and {θ_m} (with per-module LR scaling)
Bookkeeping:
for each m do
    Ũ_{m,t} = mean_i [ α_m^(i) · (baseline_loss_i − loss_i) ]
end for
Φ_m ← (1 − η)Φ_m + η Ũ_{m,t}
ᾱ_m ← (1 − ρ)ᾱ_m + ρ mean_i[α_m^(i)]
Send {Φ_m, ᾱ_m} to Evolutionary Strategist
```

The synergy was aggregated for m as a normalized attention over other modules:

$$s_m^{(h)} = \sum_{j=1}^{N} \omega_{m,j}^{(h)} \cdot q_{int}(S_{m,j}^{(h)}), \quad \omega_{m,j}^{(h)} = \frac{\exp(S_{m,j}^{(h)})}{\sum_{k=1}^{N} \exp(S_{m,j}^{(h)})}. \tag{56}$$

Here, $q_{int}(\cdot)$ is an optional nonlinearity (e.g., ReLU or identity) that lets the synergy term be asymmetric and saturating if desired. We combined relevance and synergy linearly (learnable balance):

$$s_m^{(h)}(\mathbf{f}, U) = r_m^{(h)} + \gamma^{(h)} s_m^{(h)}, \tag{57}$$

where $\gamma^{(h)} \in \mathbb{R}_{\geq 0}$ is a learned (or scheduled) head-wise scalar controlling the emphasis on inter-module synergy.

Novelty. The synergy term lets the router prefer modules that not only individually match the query but that form *complementary coalitions* for the current input - capturing pairwise (and via repeated application, higher-order) interactions among experts. This is distinct from class MoE routers that treat modules as independent.

Multi-head attention and normalized routing weights. For head h, we normalized capabilities with softmax over modules:

$$\alpha_m^{(h)} = \frac{\exp(s_m^{(h)})}{\sum_{j=1}^{N} \exp(s_j^{(h)})}. \tag{58}$$

We aggregated heads into a single routing weight per module (head-averaging or learned projection):

$$\alpha_m = \frac{1}{H}\sum_{h=1}^{H}\alpha_m^{(h)} \quad \text{or} \quad \alpha = \text{softmax}(W_{agg}[\alpha^{(1)};\ldots;\alpha^{(H)}]), \tag{59}$$

where W_{agg} projects head-wise vectors to a final distribution is desired. Here, the output of the router is the weighted mixture:

$$\mathbf{z}^{(r)} = \sum_{m=1}^{N}\alpha_m \cdot \mathbf{v}_m^{agg}, \quad \mathbf{v}_m^{agg} = \frac{1}{H}\sum_{h=1}^{H}\mathbf{v}_m^{(h)}. \tag{60}$$

This $\mathbf{z}^{(r)}$ flows to the classification head and participates in standard backpropagation: gradients pass to W_V, W_K, W_Q and - via $\mathbf{v}_m$ and $\mathbf{u}_m$ - to module parameters.

Controlled sparsity: top-k routing (efficient, capacity-aware). To enforce the **Max Active Modules** constraint and reduce compute, we designed **Soft** $\rightarrow$ **Sparse** path, where we computed dense α_m as above, then apply a differentiable sparsification to keep at most K modules per sample. Here, we had two practical, differentiable options:

1. **Sparsemax/Entmax:** We replaced softmax with sparsemax/entmax, which produces exact zeros for many entries while remaining subgradient-based and differentiable.
2. **Gumbel-TopK with straight-through (ST) estimator:** We sampled a binary mask g_m indicating top-K modules (determinisitc top-K at inference). During the forward pass, we used hard top-K selection:

$$g_m = \mathbf{1}\{\alpha_m \text{in top-K}\}, \quad \tilde{\alpha}_m = \frac{g_m \cdot \alpha_m}{\sum_j g_j \cdot \alpha_j} \tag{61}$$

For backprop, we used straight-through, where we propagated gradients to α_m as if soft selection had been used (or we kept the option of Gumbel-softmax relaxation for a differentiable approximation).

We used (and recommend) sparsemax in training for stable gradients and deterministic top-K at evaluation for reproducibility.

Router regularizers and losses. To prevent collapse onto a small subset of modules and to encourage exploration and load balancing, we incorporated three auxiliary terms in router training:

1. **Entropy Regularizer (exploration early in training):**

$$\mathcal{L}_{ent} = -\frac{1}{N}\sum_{m=1}^{N}\alpha_m \log(\alpha_m). \tag{62}$$

2. **Load-balancing penalty:** We encouraged average router usage $\tilde{\alpha}_m$ (running mean across samples/batches) to match uniform expectation $1/N$:

$$\mathcal{L}_{load} = \sum_{m=1}^{N}\left(\bar{\alpha}_m - \frac{1}{N}\right)^2, \quad \bar{\alpha}_m \leftarrow (1-\rho)\bar{\alpha}_m + \rho\mathbb{E}_{batch}[\alpha_m]. \tag{63}$$

3. **Sparsity budget:** If using sparsity, we penalized deviation from target active K via:

$$\mathcal{L}_{budget} = \left(\frac{1}{N}\sum_{m=1}^{N}\mathbf{1}\{\alpha_m > 0\} - \frac{K}{N}\right)^2 \tag{64}$$

(or an L1 surrogate on α).

G EVOLUTIONARY STRATEGIST — META-CONTROLLER FOR STRUCTURAL SELF-GROWTH

The **Evolutionary Strategist** is a meta-learning controller that continually modifies the **Neural Module Zoo** $\mathcal{M}$ during training. It operates at the level of **module genotypes** (architecture + hyperparameters) and **phenotypes** (weights, performance types), and its goal is to maximize long-term validation performance while respecting computation/complexity constraints and encouraging parametric plurality. The strategist combines: (i) an interpretable fitness signal derived from the Graph Attention Router, (ii) a set of genetic operators (prune, mutate, hybridize), and (iii) a policy π_ϕ trained with a reinforcement/meta-gradient objective. Below, we define state, actions, fitness, evolution operators, the learning objective for the controller, and practical stabilizers.

Notation and State Representation. At discrete evolution decision times $t \in \{0, T_e, 2T_e, \dots\}$, the system maintains:

- Module pool: $\mathcal{M}_t = \{m_1, \dots, m_{N_t}\}$.
- Each module m has:
  - genotype (hyperparameters, topology): θ_m (e.g., depth, width, dropout, heads, activation type),
  - phenotype (weights): w_m,
  - usage/metadata: age_m, params_m (parameter count), FLOPs_m,
  - contribution statistics: tracked variables defined below.
- Global training state S_t comprises:

$$S_t = \left\{ \{(\theta_m, w_m, \text{age}_m, \text{params}_m, C_m)\}_{m \in \mathcal{M}_t}, \text{val\_metrics}_{t-\Delta:t}, \text{budget\_remaining} \right\}, \tag{65}$$

  where C_m is a numeric contribution/fitness proxy.

The controller $\pi_\phi(a_t|S_t)$ outputs actions a_t altering $\mathcal{M}_t$ (prune, spawn/mutate, hybridize, no-op, or other maintenance actions). Actions can be multi-step (e.g., hybridize two parents into one child + spawn).

Contribution and Fitness Estimation. A robust, low-variance fitness signal is central. We combine two complementary, efficiently computable signals in each evolution epoch:

1. **Attention-contribution proxy (router-based)**

   For module m, collect the per-batch average routing weight from the Graph Attention router over a recent buffer $\mathcal{B}$ (the last B mini-batches):

$$\bar{\beta}_m = \frac{1}{B} \sum_{b \in \mathcal{B}} \beta_m^{(b)}. \tag{66}$$

2. **Leave-one-out loss impact (performance-proxy)**

   For a mini-batch b compute the batch loss with full routing $\mathcal{L}_{full}^{(b)}$ and the loss with module m ablated (zeroing or masking its output) $\mathcal{L}_{-m}^{(b)}$. We defined per-batch delta:

$$\Delta\ell_m^{(b)} = \mathcal{L}_{-m}^{(b)} - \mathcal{L}_{full}^{(b)}. \tag{67}$$

   Positive $\Delta\ell_m$ indicates the module is helpful. The average over $\mathcal{B}$:

$$\overline{\Delta\ell}_m = \frac{1}{B} \sum_{b \in \mathcal{B}} \Delta\ell_m^{(b)}. \tag{68}$$

We combine these into an exponential moving average contribution score $C_m(t)$:

$$C_m(t) \leftarrow (1-\rho)C_m(t-1) + \rho \left(w_\beta \frac{\bar{\beta}_m}{\max_k \bar{\beta}_k} + w_\ell \frac{\max(0, \bar{\Delta}\ell_m)}{\max_k \max(0, \bar{\Delta}\ell_m)} \right), \tag{69}$$

with $\rho \in (0,1)$ smoothing factor and weights w_β, w_ℓ (0.5 each). Normalization avoids scale issues. C_m is the primary short-term fitness proxy used by selection and pruning. To encourage novelty and penalize redundancy, we also compute a novelty score:

$$\text{novelty}_m = \frac{1}{N_t - 1} \sum_{k \neq m} \exp(-\gamma_\theta ||\theta_m - \theta_k||_2^2, \tag{70}$$

and defined a combined fitness:

$$F_m = \alpha_C C_m - \alpha_{cost} \cdot \text{cost}_m + \alpha_{nov}(1 - \text{novelty}_m), \tag{71}$$

where cost_m is the normalized computational cost (params or FLOPs), and α are tuning scalars. Lower novelty_m (i.e., more dissimilar) increases fitness via $1 - \text{novelty}$.

Selection and Pruning We removed modules whose long-run contribution is consistently low while respecting stability constraints:

- **Minimum survival age:** a module must survive at least A_{min} evolution intervals before being eligible for pruning.
- **Prune condition** (quantile-based):

$$\text{Prune} \quad m \quad if \quad F_m \leq Q_q(\{F_k\}_{k \in \mathcal{M}_t}) \quad and \quad age_m \geq A_{min}, \tag{72}$$

  where $Q_q(\cdot)$ is the q-th percentile $(q = 0.15)$. This avoids threshold tuning across varying pool sizes. Alternatively, a dynamic threshold $\tau_t = \mu_F - \mathcal{K}\sigma_F$ can be used (recommendation).

When pruning, we first attempt **weight recycling**: if another module has an identical genotype or an identical interface, its weights may be reused or used to initialize new offspring.

Growth (mutation) operator. To spawn variants, we sample parent modules according to a softmax over fitness:

$$p_{select}(m) = \frac{\exp(\eta F_m)}{\sum_k \exp(\eta F_k)}. \tag{73}$$

Given parent m_p with genotype θ_p and weights w_p, we create child genotype θ_c via parameter-space mutation:

- For continuous hyperparameters (dropout, width multipliers):

$$\theta_c^{(i)} = \theta_p^{(i)} \cdot \exp(\sigma_\theta \cdot \epsilon^{(i)}), \quad \epsilon^{(i)} \sim \mathcal{N}(0,1). \tag{74}$$

- For discrete hyperparameters (number of heads), we applied categorical perturbation (random $\pm$ step with small probability).

Child weights are initialized by soft inheritance:

$$w_c = \gamma_{inh} w_p + (1 - \gamma_{inh})\mathcal{N}(0, \sigma_w^2). \tag{75}$$

where $\gamma_{inh} \in [0,1]$ controls how much of parent knowledge is retained. This reduces cold-start training and stabilizes learning when the child shares structural motifs with the parent. A growth rate constraint keeps the pool budgeted: at most G_{max} new modules per evolution step and $N_t \leq N_{max}$.

Hybridization (Co-Evolutionary Crossover) Hybridization recombines structural motifs and hyperparameters from two high-fitness parents m_i and m_j to create a child m_c. We treat module genotypes as graph-structured objects (topology + attributes). Let $T_m = (V_m, E_m, \Theta_m)$ denote parent m's topology graph, node attributes Θ_m (layer types, widths, activation), and W_m the associated weight tensors.

Crossover operator (motif splice):

1. **Motif extraction:** We sampled subgraph $S_i \subseteq T_{m_i}$ and $S_j \subseteq T_{m_j}$ by selecting contiguous substructures using a size distribution (small-to-medium). We represent these as adjacency and attribute sets.
2. **Interface alignment:** We find interface nodes $u \in S_i, v \in S_j$ where input/output dimensionalities can be projected. If dims differ, create small projection layers $P_{in} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_c}$ and P_{out} as learned linear maps. This enforces compatibility.
3. **Splice:** We create child topology

$$T_c = (T_{m_i} \ S_i) \cup S_j, \tag{76}$$

where S_j is grafted into T_{m_i} at matched interfaces. (Symmetric alternatives allowed.)

4. **Hyperparameter recombination:** For scalar attributes in Θ, we performed convex interpolation:

$$\theta_c^{(k)} = \lambda \theta_{m_i}^{(k)} + (1 - \lambda)\theta_{m_j}^{(k)}, \quad \lambda \sim \mathcal{U}(0, 1). \tag{77}$$

For categorical attributes, we used parent-sampling with probability proportional to normalized parent fitness.

5. **Weight inheritance mapping:** The parameters for retained subgraphs are copied; for grafted subgraphs, we used soft weight blending where possible:

$$W_c[shared] = \mathcal{K} W_{m_i}[shared] + (1 - \mathcal{K}) W_{m_j}[shared] + \epsilon, \tag{78}$$

and new parameters are initialized as small-noise or adapted from th nearest parent via projection.

This motif-based crossover allows the child to inherit functional building blocks (e.g., a multi-head attention motif with a particular head-to-dimension ratio) and yields architectures not present in the initial search space.

Controller Optimization The controller π_ϕ must learn when to prune, spawn, and hybridize to maximize long-term validation performance under computation budget B. We pose this as a constrained expected reward maximization:

$$\max_{\phi} \mathbb{E}_{\tau \sim \pi_\phi} \left[\sum_{t=0}^{T} \gamma^t r(S_t, a_t) \right] \quad s.t. \quad \mathbb{E}_{\tau \sim \pi_\phi} \left[\text{Cost}(\tau) \right] \leq B, \tag{79}$$

where τ is an evolution trajectory, γ discount factor, and reward r is computed at evolution intervals. We used a Lagrangian relaxation:

$$\mathcal{J}(\phi, \lambda) = \mathbb{E} \left[\sum_t \gamma^t r_t - \lambda (\text{Cost}_t - B_t) \right], \tag{80}$$

and optimize ϕ via policy gradient (e.g., PPO) with gradient estimator:

$$\nabla_\phi \mathcal{J} \approx \mathbb{E} \left[\sum_t \nabla_\phi \log \pi_\phi(a_t | S_t) \tilde{A}_t \right], \tag{81}$$

Algorithm 2 Evolutionary Strategist for Neural Module Evolution

Require: Module set $\mathcal{M} = \{M_1, \ldots, M_K\}$, fused embeddings $\mathbf{z} \in \mathbb{R}^d$, contribution scores α_i, replay memory $\mathcal{R}$
Ensure: Updated module set $\mathcal{M}'$

Initialize policy π_θ for meta-controller
while training not converged **do**
  Sample task batch $\mathcal{B} \sim \mathcal{D}$
  Compute fused embedding $\mathbf{z}$
  Route $\mathbf{z}$ to modules using GraphAttentionRouter
  Compute contributions $\alpha_i = \mathrm{softmax}\big(\frac{\mathbf{z}^\top \mathbf{k}_i}{\sqrt{d}}\big)$
  Evaluate task loss $\mathcal{L}_{task}$ and reward $R(\mathcal{M}) = -\mathcal{L}_{task} + \lambda H(\alpha)$
  Store $(\mathbf{z}, \mathcal{M}, R)$ in replay memory $\mathcal{R}$
  {— Evolutionary Update —}
  if $\alpha_i < \tau_{prune}$ for consecutive T steps **then**
    Remove module M_i from $\mathcal{M}$ (Pruning Rule)
  end if
  if $R(\mathcal{M}) < \tau_{grow}$ **then**
    Spawn new module M'_j with parameters [1] $\Theta'_j = \Theta_j + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I)$ (Growth Rule)
    Add M'_j to $\mathcal{M}$
  end if
  if $\exists\, M_p, M_q \in \mathcal{M}$ with high complementarity **then**
    Generate child M_c via crossover: [1] $\Theta_c = \eta\Theta_p + (1-\eta)\Theta_q, \quad \eta \sim \mathcal{U}(0,1)$ (Hybridization Rule)
    Add M_c to $\mathcal{M}$
  end if
  {— Meta-Controller Update —}
  Compute policy gradient: [1] $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\big[\nabla_\theta \log \pi_\theta(a|\mathcal{M})\, R(\mathcal{M})\big]$
  Update $\theta \leftarrow \theta + \beta \nabla_\theta J(\theta)$
end while

return $\mathcal{M}'$

where $\tilde{A}_t$ is an advantage estimate (computed from actual validation metric improvement over a horizon H). The reward r_t is defined as:

$$r_t = \Delta\text{ValMetric}_{t\to t+H} - \eta_{comp}\Delta\text{Cost}_{t\to t+H} + \eta_{div}\overline{\text{overline}}_{t\to t+H}, \tag{82}$$

balancing short-term performance gain, computational cost, and architectural novelty. In practice, we set H to a modest number of training steps to trade off noise vs signal. Alternatively, a meta-gradient approach can be used where action parameters are differentiable (soft choices) and the outer validation loss is differentiated w.r.t. ϕ by unrolling a few inner optimization steps. We recommend policy-gradient (PPO) in experiments for stability and scalability, with meta-gradient used in ablations to evaluate potential improvements.

Stabilization, replay, and reproducibility. Structural modifications can destabilize training. We used three stabilizers:

1. **Replay memory $\mathcal{R}$:** We maintained a buffer of representative examples (stratified by class/modality) and replay them for R mini-batches immediately after structural changes. This limits catastrophic forgetting and calibrates newly created modules.
2. **Warm-start fine-tuning:** After spawning/hybridization, child modules are trained with a reduced learning rate $\eta_{child} = \zeta\eta$ for E_{warm} steps before making further evolutionary decisions.
3. **Minimum-age and hysteresis:** Modules must remain for A_{min} epochs to allow their contributions to be reliably estimated; pruning decisions incorporate running variance to prevent thrashing.

For reproducibility, every structural operation (prune/mutate/hybridize) is logged with a 64-bit RNG seed, parent IDs, and a deterministic construction routine. This results in reproducible architecture evolution given the same global initial seed.

H TRAINING OBJECTIVE

Notation & Problem Statement.

- Let an architecture (set of active modules and their hyperparameters) be $A = \{(m, \eta_m)\}_{m \in \mathcal{M}}$, where η_m are module hyperparameters (depth, heads, dropout, widths), and $\mathcal{M}$ is the active module index set.
- Let $\Theta = \{\theta_m\}_{m \in \mathcal{M}}$ denote all module weights plus router and head weights; let θ_{ext} denote the multimodal extractor weights (DistilBERT, CLIP-ViT).
- Router produces per-sample soft contributions $\beta_m(x)$ for sample x. For a minibatch B, denote $\beta_m(B) = \frac{1}{|B|} \sum_{x \in \mathcal{B}} \beta_m(x)$.
- Meta-controller (Evolutionary Strategist) is parameterized by ϕ and implements a policy π_ϕ which, at discrete architectural decision times, outputs actions $a \in \mathcal{A}$ (prune, grow, hybridize, and their parameters).
- Let $\mathcal{R}$ be the replay buffer (capacity N_R).

We cast the training as the following bilevel objective:

$$\begin{aligned} &\text{Outer / meta (architectural) objective:} \quad \max_\phi \mathbb{E}_{\tau \sim \pi_\phi} \left[\mathcal{P}_{val}(\Theta^\tau, A^\tau) - c \cdot \mathcal{C}(A^\tau) \right] \\ &\text{Inner / param (weights) objective:} \quad \Theta^\tau \approx \arg\min_\Theta \mathcal{L}_{train}(\Theta, A^\tau; \mathcal{D}_{train}), \end{aligned} \tag{83}$$

where $\mathcal{P}_{val}$ is a validation performance metric (e.g. AUC), $\mathcal{C}(A)$ is an architectural cost (parameters, FLOPs), and τ denotes a stochastic architecture trajectory induced by π_ϕ. Because architectures are discrete and evolution is online, we used a hybrid of gradient-based inner training and policy-gradient outer optimization.

Inner (parameter) loss. For a minibatch $B = \{(x, y)\}$. the *base task loss* is binary cross-entropy:

$$\begin{aligned} \mathcal{L}_{task}(B; \Theta, A) &= \tfrac{1}{|B|} \textstyle\sum_{(x,y) \in B} \text{CE}(y, \hat{y}(x; \Theta, A)) \\ \hat{y}(x; \Theta, A) &= \sigma(W_o z^{(r)}(x; \Theta, A)), \end{aligned} \tag{84}$$

where $z^{(r)}$ is the router's weighted mixture output. To encourage *per-sample routing diversity* (avoid collapse to a single module), we used an entropy reward on router weights averaged over the batch:

$$\mathcal{L}_{div}(B; \Theta, A) = -\frac{1}{|B|} \sum_{x \in B} \sum_{m \in \mathcal{M}} \beta_m(x) \log \beta_m(x). \tag{85}$$

To encourage *representational orthogonality* between module outputs (parametric plurality beyond mere usage), we included a pairwise cosine-similarity penalty:

$$\mathcal{L}_{orth}(B; \Theta, A) = \frac{2}{|\mathcal{M}|(|\mathcal{M}| - 1)} \sum_{i<j} \left(\frac{\langle u_i(B), u_j(B) \rangle}{||u_i(B)||||u_j(B)||} \right)^2, \tag{86}$$

where $u_m(B) = \frac{1}{|B|} \sum_{x \in B} u_m(x)$ is the batch-averaged module output (or one can use per-sample pairwise terms averaged). We penalized *architectural complexity* (to avoid unconstrained growth):

$$\mathcal{L}_{comp}(A) = \alpha_{param} \sum_{m \in \mathcal{M}} \text{params}(m) \cdot g_m, \quad g_m = \min\{1, \text{clip}(\beta_m^{avg}/\epsilon, 0, 1)\}, \tag{87}$$

where β_m^{avg} is a long-run usage estimate and g_m behaves as a soft gate: rarely used modules incur less cost. To mitigate catastrophic forgetting when the architecture changes, we used *replay loss:*

$$\mathcal{L}_{replay}(\mathcal{R};\Theta,A)=\frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}\subset\mathcal{R}}\mathrm{CE}(y,\hat{y}(x;\Theta,A)), \tag{88}$$

with $\mathcal{S}$ a randomly sampled minibatch from the buffer. Finally, the inner total loss used to update Θ is:

$$\boxed{\mathcal{L}_{train}(B;\Theta,A)=\mathcal{L}_{task}+\lambda_{div}\mathcal{L}_{div}+\lambda_{orth}\mathcal{L}_{orth}+\lambda_{replay}\mathcal{L}_{replay}+\lambda_{comp}\mathcal{L}_{comp}} \tag{89}$$

All λ's are hyperparameters tuned to balance accuracy, diversity, and compactness. Θ is updated by standard SGD/Adam steps minimizing $\mathcal{L}_{train}$. The router parameters (and extractor finetuning) are included in Θ and receive gradients through β and the mixture $z^{(r)}$.

Module Fitness and Contribution Estimator. The strategist must decide which modules to prune, which to hybridize, and which to use as parents for growth. Decisions rely on a fitness score f_m per module that reflects usefulness and marginal contribution. We propose a practical estimator that balances fidelity and computation:

1. **Usage estimate (fast):**

$$u_m^{(t)}=\mathrm{EMA}_\rho(\beta_m(B_t)), \tag{90}$$

   an exponential moving average over minibatches with decay ρ.

2. **Marginal contribution (periodic, higher fidelity):** For every T_{eval} minibatches, we estimated the marginal loss drop of module m on a small validation probe P:

$$\Delta\mathcal{L}_m\approx\frac{1}{|P|}\sum_{x\in P}(\mathcal{L}(x;\Theta,A/\{m\})-\mathcal{L}(x;\Theta,A)), \tag{91}$$

   where $A/\{m\}$ is the architecture with m ablated (set $\beta_m=0$ and renormalize). Positive $\Delta\mathcal{L}_m$ means the module helps.

3. **Composite fitness:** We combine both signals:

$$f_m=\gamma_1 u_m^{(t)}+\gamma_2\mathrm{ReLU}(\Delta\mathcal{L}_m), \tag{92}$$

   normalized across modules. γ weights trade off frequency vs casual contribution.

The strategist prunes modules with $f_m<\tau_{prune}$ and age ¿ A_{mini}; spawns children from parents sampled proportional to f_m; selects parents for hybridization stochastically using fitness-proportionate selection.

Evolutionary Actions. Let action set $\mathcal{A}$ include:

- **prune(m)**: remove module m permanently (or mark inactive).
- **grow**(p,δ_η): spawn new module from parent p with hyperparameter perturbation δ_n.
- **hybridize**(p_i,p_j,λ): create child hyperparameters

$$\eta_c=\lambda\eta_{p_i}+(1-\lambda)\eta_{p_j}+\epsilon,\epsilon\sim\mathcal{N}(0,\sigma^2). \tag{93}$$

Weight inheritance. Child weights θ_c are warm-started by structured inheritance:

- for hybridization: $\theta_c=\lambda\theta_{p_i}+(1-\lambda)\theta_{p_j}+\zeta$, with small noise $\zeta\sim\mathcal{N}(0,\sigma_w^2)$.

- for growth by mutation: copy and perturb parent: $\theta_c = \theta_p + \zeta$.
  After creation, children undergo a short warm-up period of T_{warm} minibatches with a smaller learning rate η_w to prevent destabilization.

Knowledge distillation on pruning. Before the pruning module m, we optionally perform a distillation step so that the remaining modules can absorb its functionality:

$$\mathcal{L}_{kd} = \frac{1}{|S|} \sum_{x \in S} ||z^{(r)}_{full}(x) - z^{(r)}_{ablated}(x)||_2^2, \tag{94}$$

where $z^{(r)}_{full}$ uses m and $z^{(r)}_{ablated}$ does not. Minimizing $\mathcal{L}_{kd}$ for a few steps softens the removal.

Outer (Meta) Objective and Optimization of ϕ. The strategist parameter ϕ defines a policy $\pi_\phi(a_t|s_t)$ that, given state s_t (module fitness vector $\{f_m\}$, age, resource usage, recent validation trajectory, etc.), outputs an action distribution. The meta-reward r_t should encourage long-term validation gains while penalizing cost:

$$r_t = \Delta\mathcal{P}_{val,t} - \eta_{param}\Delta\text{Params}_t - \eta_{flops}\Delta\text{FLOPs}_t - k \cdot \mathcal{C}_{instab,t}, \tag{95}$$

where $\mathcal{P}_{val,t} = \mathcal{P}_{val}(t+\Delta) - \mathcal{P}_{val}(t)$ is the improvement observed after applying action(s) and letting the model train for a short horizon, and $\mathcal{C}_{instab,t}$ penalizes validation volatility (to avoid reckless growth that yields unstable gains). We maximized expected return:

$$J(\phi) = \mathbb{E}_{\tau \sim \pi_\phi}\left[\sum_{t=0}^{T} r_t\right]. \tag{96}$$

We applied two practical optimization strategies here:

1. **Policy Gradient (REINFORCE).** We used sampled trajectories of length T_{meta}, estimate returns $R_t = \sum_{k=t}^{T} r_k$, and update:

   $$\nabla_\phi J \approx \mathbb{E}\left[\sum_t \nabla_\phi \log \pi_\phi(a_t|s_t)(R_t - b_t)\right], \tag{97}$$

   where b_t is a learned baseline (value network) to reduce variance. Entropy regularization $-\lambda_H \sum_t \mathcal{H}(\pi_\phi(\cdot|s_t))$ is added to encourage exploration.
2. **Truncated Meta-Gradient (Differentiable Unroll).** When computational budget allows, we unrolled k inner optimization steps of Θ after an action and differentiate the validation loss w.r.t. ϕ via chain rule (truncated backprop through optimization). Let $\Theta_{t+k}(\phi)$ denote the inner optimized weights after K steps influenced by decisions sampled from π_ϕ. Then,

   $$\nabla_\phi \mathcal{L}_{val}(\Theta_{t+k}(\phi)) = \frac{\partial \mathcal{L}_{val}}{\partial \Theta_{t+k}} \cdot \frac{\partial \Theta_{t+k}}{\partial \phi}, \tag{98}$$

   which we compute with automatic differentiation for small K. This gives lower variance but larger memory/computation. In practice, we combine both: use REINFORCE for long-horizon exploration and occasional truncated meta-gradient updates for fine-tuning.