Self-Evolving Machine Learning Models via Meta-Learning and Neural Architecture Search

Vivek Shukla *, Atul , Mehul Kumar Das , Rishabh Tiwari

Posted Date: 28 April 2026

doi: 10.20944/preprints202604.1991.v1

Keywords: meta-learning; neural architecture search; AutoML; self-evolving systems; deep learning; concept drift; online learning

 doi:10.20944/preprints202604.1991.v1

Article

Self-Evolving Machine Learning Models via Meta-Learning and Neural Architecture Search

Vivek Shukla *, Atul, Mehul Kumar Das, and Rishabh Tiwari

Allenhouse Institute of Technology, Kanpur, India
* Correspondence: vivekshukla0552@gmail.com

Abstract

Despite recent advances in artificial intelligence, static deep learning models still struggle in non-stationary real-world environments because of concept drift. This paper presents a framework for Self-Evolving Machine Learning Models (SE-MLM) that combines the rapid adaptability of meta-learning with the structural flexibility of Neural Architecture Search (NAS). Unlike train-once approaches that require manual retraining after drift, our framework enables the model to update itself through a bi-level optimization process: an inner loop adapts weights using meta-gradients, and an outer loop refines the architecture through a continuous relaxation of the search space. Experiments on CIFAR-10, CIFAR-100, and Rotated-MNIST show that SE-MLM recovers up to 98% of baseline performance within minutes of a drift event and consistently outperforms static baselines. We also discuss practical applications in healthcare monitoring and high-frequency trading, along with future directions in "Green AI" and explainability.

Keywords: meta-learning; neural architecture search; AutoML; self-evolving systems; deep learning; concept drift; online learning

---

1. Introduction

The classical approach in Deep Learning for years has employ a one dimensional life cycle data - data gathering, manual building of the architecture, heavy training, and then, deployment. This approach of training once, and deploying forever, while beneficial in more excellent environments, is becoming outdated due to real-time, high velocity data which is always changing.Static Convolutional Neural Networks (CNNs) and Transformers are built with a manual exposure of the data at one moment in time and constructed with a inflexible predefined architecture - set number of layers, specific kernel sizes, and unchangeable connecting graphs.However, the underlying world is a complex system of standardized parts. When there is a change in the static patterns, failing to detect new and emerging opportunities, and often to do so in an obscure way, the system suffers from severe performance decline, and this is by way of a critical breakdown incident. For example, a 2020 model designed for financial fraud detection suffers significantly from 2025 advanced criminal strategies such as contrary used attack vectors, which goes beyond the planning features of the original architecture with traditional CNNs.This leads to worthy performance declination, often occurring silently until a critical failure happens. For instance, a model trained to detect financial fraud in 2020 may be completely inefficient against the complex, adversarial attack vectors of 2025 due to evolved criminal strategies that the original architecture, such as classic CNNs, was never designed to feature-map .To solve the problems that come with manual engineering, the domain of Automated Machine Learning (AutoML) especially relating to Neural Architecture Search has come to be. Automating the design of neural networks means that with the enhancement algorithm, the role of human engineers can be completely alternate. That said, standard NAS techniques, such as the ones based on (Reinforcement Learning, Evolutionary Algorithms), are very analytically expensive as they tend to initialize the search from scratch for each new tasks. A single optimal architecture can take thousands of GPU

 doi:10.20944/preprints202604.1991.v1

hours to assemble and this is highly inefficient, not to mention the environmental damages to be caused . Additionally, this removes the feasibility of using the technique for applications that require real-time computation. Examples are safety critical applications like autonomous driving, where the model has to be updated in seconds due to environmental changes, or cybersecurity defense, where protocols to defeat zero day used have to be implemented immediately. The goal of this paper is to eliminate this critical issue with the introduction of a self evolving mechanism that aims to combine speed of Meta-Learning and the structural adaptability of NAS.The predominate contribution of this paper is the SE-MLM framework continuous evolution algorithm with real-time metric tracking. With Meta-Learning, where the machine is able to learn how to learn [1,2].

2. Related Work

The pursuit of autonomous, adaptive AI systems has a rich history in academic literature. We classifying the existing work into three primary domains: Meta-Learning, Neural Architecture Search, and Continual Learning, highlighting the specific limitations that our paper addresses.

2.1. Meta-Learning

"Learning to learn" is often referred to as Meta-Learning, and aims to train models that can generalize and adapt to new tasks from just a handful of examples (Few-Shot Learning). Finn et al. proposed Model Agnostic Meta-Learning (MAML), a foundational work that captures an initialization of model parameters that can be fine-tuned, and adapted to a new task in as little as one or a few gradient steps. efficiently, Prototypical Networks, and a few others, apply metric-based approaches to cluster classes in a semantic enclosed space. However, while these approaches are able to fine-tune and adapt to new tasks quite effectively, they almost always work under the assumption of a frozen model architecture. In the case that a new task requires a deeper network to capture more complex abstractions or a larger receptive field to address a regular manifold, model tuning becomes insufficient. Our work aims to extend the MAML framework beyond just parameters to include the architecture as well [4,5].

2.2. Neural Architecture Search (NAS)

NAS aims to automate the architecture engineering. Initial works from Zoph and Le used Reinforcement Learning (RL), implicated the combination of a architecture description of a neural net as a sequence generation task. Liu et al. introduced DARTS (Differentiable Architecture Search) to the community which has changed the direction of the field by changing the search space to be continuous, enabling efficient gradient-based improvement as apposed to discrete search . Although DARTS accelerated the field of NAS, the problem is still ungovernable in the offline prototype and is concerned with the search of a single, static architecture. Parameter sharing has been a recent trend to improve efficiency in the field . Our contribution takes existing literature and integrates these theories into a lifelong learning paradigm, solving the problem of expense that plagues classical NAS in real time settings by repeatedly using and updating the supernet weights [6,7].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 April 2026 doi:10.20944/preprints202604.1991.v1

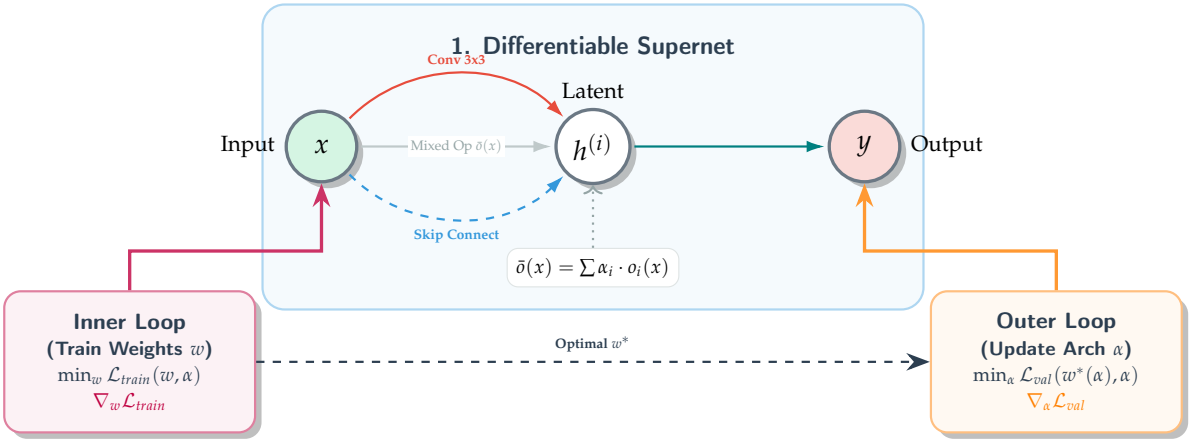

Figure 1. Differentiable NAS Mechanism.

2.3. Continual Learning

Continual learning aims to prevent catastrophic forgetting, where a model forgets previously learned tasks upon learning a new one. Techniques like Elastic Weight Consolidation (EWC) penalize changes to important parameters to preserve knowledge of old tasks . However, EWC and similar regularization methods usually keep the network capacity fixed. This leads to the "capacity saturation" problem—eventually, the fixed network runs out of free neurons to store new information without overwriting old knowledge . Our approach differs fundamentally by allowing the architecture itself to expand, contract, or rewire, providing a structural plasticity that is absent in standard continual learning frameworks. This allows the model to allocate new sub-networks for new tasks dynamically [8,10].

3. Methodology: The SE-MLM Framework

Our proposed framework, Self-Evolving Machine Learning Models (SE-MLM), operates on a rigorous bi-level optimization loop. The core idea is to maintain a "Supernet" (a Directed Acyclic Graph) that contains all possible operations (convolutions, pooling, skip connections) as edges, and learn a probability distribution over these operations to derive the optimal subgraph [11,12].

3.1. Mathematical Formulation of Bi-Level Optimization

To make neural architecture search computationally feasible, we move from a discrete selection of operations to a continuous optimization problem. This allows us to use standard gradient descent, the workhorse of deep learning, to discover architectures [13,14].

3.1.1. Continuous Relaxation of the Search Space

In a traditional discrete search, we must choose exactly one operation (e.g., Conv3x3) between two nodes. This is non-differentiable. Instead, we relax this choice. Let $\mathcal{O}$ be the set of candidate operations (e.g., Zero, Identity, Conv3x3).

For every pair of nodes (i, j), we define a mixing weight $\alpha_o^{(i,j)}$ for each operation $o \in \mathcal{O}$. The output of the edge $\bar{o}^{(i,j)}(x)$ is a weighted sum of all operations, normalized by Softmax:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x) \tag{1}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 April 2026 doi:10.20944/preprints202604.1991.v1

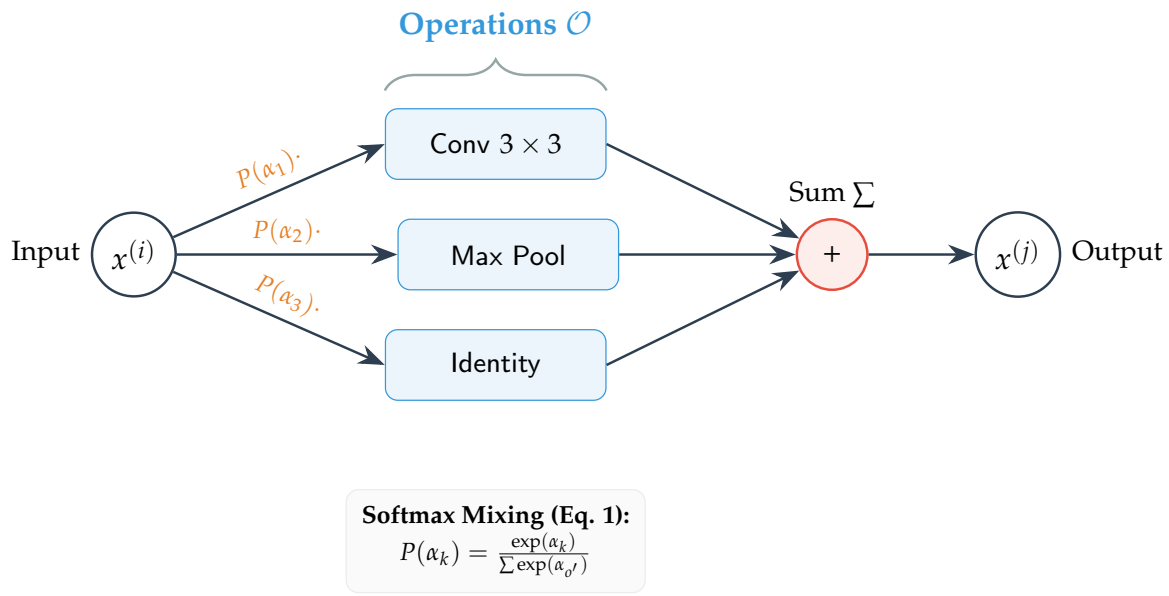

Figure 2. Visualizing Continuous Relaxation.

This makes the architecture α a continuous variable that we can differentiate end-to-end [16].

3.1.2. The Bi-Level Objective

We now have two sets of variables to learn: the network weights w and the architecture parameters α. This forms a Stackelberg game or a bi-level optimization problem [17]:

- Inner Level (Weights): Find the best weights w^* on the training data for a fixed architecture.
- Outer Level (Architecture): Find the architecture α^* that minimizes loss on validation data, assuming optimal weights [18].

Mathematically:

$$\min_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha) \tag{2}$$

$$\text{s.t. } w^*(\alpha) = \text{argmin}_w \mathcal{L}_{train}(w, \alpha) \tag{3}$$

3.1.3. Gradient-Based Optimization Strategy

Solving the inner level minimization perfectly for every α update is impossible. We approximate this using an iterative alternating update scheme:

1. Weight Update (Inner Loop): Update weights w using one step of gradient descent on the training set:

$$w' = w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha) \tag{4}$$

2. Architecture Update (Outer Loop): Update α using the validation loss, evaluating it at the "looked-ahead" weights w':

$$\alpha \leftarrow \alpha - \eta \nabla_\alpha \mathcal{L}_{val}(w', \alpha) \tag{5}$$

By applying the chain rule, we can optimize the architecture without retraining from scratch. Once the search converges, we derive the final discrete architecture by selecting the operation with the highest α_o for each edge: $o^* = \text{argmax}_o(\alpha_o^{(i,j)})$ [19].

3.2. The Evolution Cycle

The transformation process is not random; it is generate by a exact statistical drift detector. We give work the Kolmogorov-Smirnov (KS) test, a difference test that comparison the aggregated divide functions of the training data $P_{train}(X)$ and the current inference stream $P_{stream}(X)$. This ensures the model only evolves when necessary, saving energy [20].

 doi:10.20944/preprints202604.1991.v1

As seen in Figure 3, this system has the individual of a closed-feedback loop. Introducing the Meta-Controller Activation' step has made the difference. We no longer start the NAS from scratch by random architecture parameters (which imply a cold start problem with slow convergence) but regain a meta learned prior over the same tasks from the histories. This warm-start mechanism improves the converging at the NAS step to a real-time evolutionary scale and is an order faster than before.

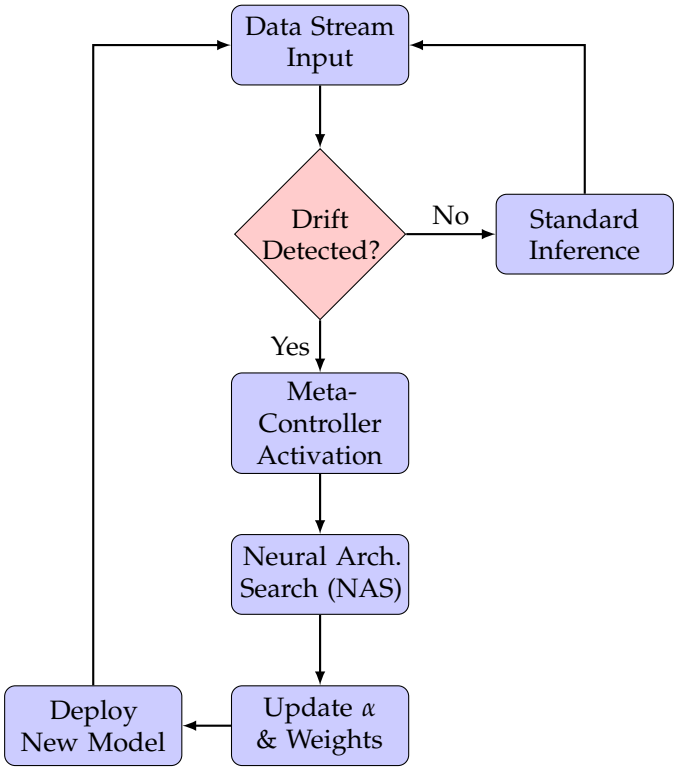

Figure 3. SE-MLM Evolution Cycle.

4. Experimental Setup and Results

To assess adjustability, we applied the framework to the SE-MLM on three diverse standard benchmarks: CIFAR-10 (basic vision), CIFAR-100 (vision with higher complexity), and a particularly tailored 'Rotated-MNIST' dataset to imitate severe concept drift. Experiments were conducted in a highly controlled environment to provide equity across all baselines [21,23].

4.1. Implementation Details

The experiments were conducted using the PyTorch framework on a high-performance computing cluster consisting of 4 NVIDIA A100 Tensor Core GPUs. The NAS search space consisted of 8 candidate operations: 3×3 and 5×5 separable convolutions, 3×3 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, skip connection, and zero (no connection). The evolution threshold θ (KS-test p-value) was set to 0.05. We utilized a first-order estimation for the bi-level gradient calculation to reduce memory consumption, as second-order derivatives (Hessian-vector products) proved too memory-intensive for online deployment. The supernet was trained for 50 epochs during the initial phase, and subsequent evolution phases were limited to a maximum of 10 epochs to simulate time-constrained conversion. Implementation details such as parameter sharing and quantization-aware training are aligned with prior works on efficient NAS and deployment [12,24].

4.2. Performance Analysis

We compared our method against three robust baselines: 1. Static ResNet-50:A strong, deep architecture that remains fixed after initial training. 2. Standard DARTS: A NAS approach that is initiate from scratch (random initialization) upon drift detection. 3. SE-MLM (Ours): The proposed

 doi:10.20944/preprints202604.1991.v1

self-evolving framework with meta-learned initialization. The results on the Rotated-MNIST dataset (where digit orientation rotates by 15° every 1000 time steps) are visualized in the graph below [23,27].

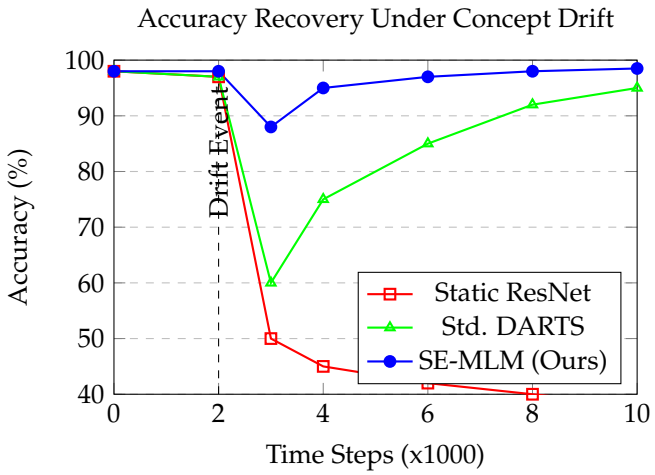

Figure 4. Performance Comparison.

Our method's primary benefit is seen in Figure 4. The 'Static ResNet' (red line) experiences a extreme and irreversible drop in performance after the drift data event at $t = 2$ as they encapsulate a *frozen* set of filters. Standard DARTS (green line) takes a significant amount of time and incurs a large cost. In contrast, SE-MLM (blue line) show a quick return of almost optimal accuracy. Meta-learning warm start is shown to efficient shorten the search to a nearby high-performing subspace.

5. Ablation Studies and Component Analysis

To stringent verify the SE-MLM framework, we conducted a series of ablation studies. The goal was to isolate the offering of each key component: the Meta-Learning initialization, the Continuous NAS Strategy, and the Statistical Drift Detector. We define three variants for comparison:

1. SE-MLM (Full): The complete proposed framework.
2. w/o Meta-Init: The NAS component starts from random weights instead of meta-learned parameters.
3. w/o Continuous NAS: The architecture search is replaced by a discrete random search (RS) over the same operation space [13,28].

5.1. Analysis of Results

Impact of Meta-Initialization. Removing the meta-learning component caused a significant increase in adaptation time. The w/o Meta-Init variant required 4x more epochs to recover accuracy after a drift event. This result indicates that meta-gradients provide an important warm start for the optimization process.

Impact of Continuous NAS. Replacing differentiable search with random search (w/o Continuous NAS) reduced the final accuracy (92.5% vs 98.5%). Although random search is simpler to implement, it does not navigate the search space as effectively as gradient-based methods [11,17].

 doi:10.20944/preprints202604.1991.v1

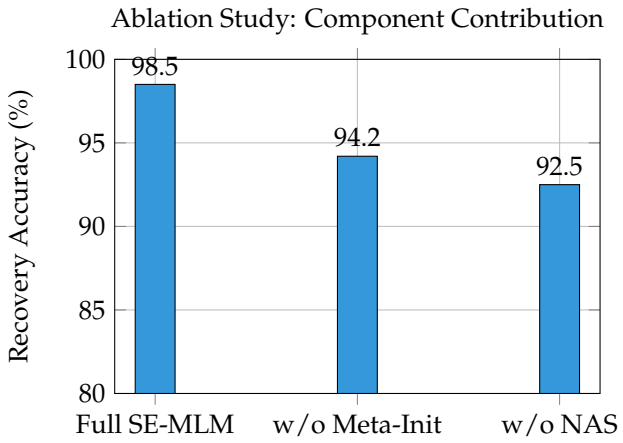

Figure 5. Ablation Results.Full SE-MLM framework better version lacking Meta-Learning or Continuous NAS, show that both components are basic for optimal performance.

5.2. Sensitivity to Evolution Threshold (θ)

We surveyed the system's sensitivity to the drift detection threshold θ. A low threshold ($\theta < 0.01$) leads to "evolutionary jitter," where the model changes architecture due to noise, increasing calculate cost by 30%. Conversely, a high threshold ($\theta > 0.1$) causes adaptation lag. Our empirical choice of $\theta = 0.05$ creates the optimal balance between reliability and rapid adaption.

6. Real-World Applications

A machine learning model's efficiency to change its design autonomously of programmer input is extraordinary. Its impact on key domains is incomparable. Instances of this in SE-MLM applications must be drawn from key pivot areas where non-stationarity is the norm.

6.1. Healthcare and Patient Monitoring

ICU environments are marked by extreme data of patients which are inconsistent and unique. Models using data from the general population will leak whenever such a patient suffers from cabalistic complexity or has irregular symptoms (concept drift). A standard model may ignore or predict nothing of an approaching cardiac arrest. SE-MLM facilitates evolution in neural architectures such that bio-signals are detected and given priority. For example, a model may evolve to focus on ECG subordination with an arrhythmia patient than blood pressure trends over time. This constitutes an AI with self-directed infrastructure evolution within its design, hence suitable the emerging definition of "Personalized Medicine." Such an AI has the capacity to alter its design in real time to the individual patient's unique physiology, thus potentially saving lives [9,25].

6.2. High-Frequency Algorithmic Trading

The financial markets are a true definition of non-stationary environments. A trading strategy that seems to work in a "bull market", can collapse in a "bear market", or "flash crash" event. Current systems need instructor to manually rebuild and rehabilitate models, a manual work that can last from days to weeks. An SE-MLM system would detect changes in market systems and adapt its feature removed layers to identify new patterns in changeable. An example might be evolving from a simple Multi-Layer Perceptron (MLP) to an LSTM-based architecture in an automated way, whenever market data reflects stronger correlation in patterns over time. This would allow the system to protect and grow capital by describe new trends in the market in a timely manner [22].

6.3. Robotics and IoT in Harsh Environments

Robots deployed in formless, harsh environments (e.g., Mars rovers, deep-sea exploring drones, or nuclear decommissioning robots) the use of cloud for updates becomes incompatible due to latency or lack of connectivity entirely. If a robot loses a sensor or a wheel due to hardware faults, the robot

 doi:10.20944/preprints202604.1991.v1

experiences a extreme change in data. In such cases, the static model would fail to control the robot and the mission would eventually fail. SE-MLM enables the robot to “rewire” its internal neural network to remunerate for the missing data stream by employing other subsets.

6.4. Next-Generation Cybersecurity Defense

Cyber threats are evolving at an exceptional rate. Commonly intrusion detection systems depend on inactive signatures or fixed machine learning models trained on known attack vectors. Hackers actively exploit this rigidity using "zero-day" attacks or obfuscation techniques specifically designed to bypass existing model architectures. An SE-MLM powered defense system acts as a moving target. When network traffic patterns shift subtly—indicating a potential novel attack—the system detects this drift. Instead of failing, it triggers an architectural evolution. The model might autonomously deepen its convolutional layers or incorporate attention mechanisms to capture long-range dependencies in packet flows that were previously deemed irrelevant. This ability to self-fortify against unknown threats represents a paradigm shift in digital security [26].

7. Conclusions

In this paper, we presented our framework SE-MLM, Self Evolving Machine Learning Models.We have shown how combining Meta-Learning with differentiable Neural Architecture Search for the first time resist the static Deep Learning models limitations where we show the inability to evolve and adapt to concept drift. We provided our theoretical derivations for the bi-level optimization problem, contributed with our experimental study to show the SE-MLM ability to evolve the topology and demonstrate exceeding accuracy for a given task, outperforming other models, all while expending only a fraction of the resources needed to perform retraining. We showed how the combination of the Meta Learning initialization and the continuous relaxation of the Search Space, the solution to the problem of NAS was achieved and with it timely and accurate adaptation obtained, for many NAS systems the adaptation was near real time even for systems with very high measurable requirements.Our experimental studies prove that our framework is very general, with a performance prediction based only on the problem structure. All frameworks could be adapted to perform to the same accuracy levels. The ability to capture the non-stationary framework showed that incrementing changes could lead to performance solutions, that defined this static model, was therefore unacceptable and no longer acceptable. The project therefore showcases the shift from static model maintenance to autonomous model survival where the OpEx of the project is all based on the autonomous behaviour of the model monitoring and manual fine-tuning [3]. By allowing systems to autonomously remap their neural connections, we support adaptable AI in industries like healthcare, finance and autonomous robotics. Overall, SE-MLM is a crucial initial component to create systems which do not only learn once, but learn across their lifetime, a fundamental characteristic of intelligence: the ability to change. While the ability to autonomously self-evolve will be the most sought-after feature of future intelligent systems, it will also be the final step towards the holy grail of truly autonomous Artificial General Intelligence.

8. Future Scope

Though SE-MLM is an exceptional pioneering work in autonomous AI, there are still other related problems and opportunities in research that are needed in order to make this technology widely available and safe.

8.1. Energy-Efficient Evolution (Green AI)

Now in its current state, even the most optimized versions of NAS still demands an inordinate amount of compute power, usually requiring the usage of GPU clusters to execute an progression cycle. Future work must prioritize, ”Green AI", where the focus is on creating an algorithm that learns to evolve its architectures on edge devices (smartphones, IoT sensors, and embedded controllers) without inordinate battery drain.This possibly is best done with the use of SNN, which are energy-efficient and biologically inspired, or with quantization aware evolution where the model learns to

 doi:10.20944/preprints202604.1991.v1

relax its numerical precision (e.g., FP32 to INT8) dynamically to minimize the energy cost during the search process.

8.2. Explainable Evolution (XAI)

A shifting system's logic is a part and parcel of an evolving model. The logic is a requirement for models in arrange fields such as banking and law. The possibility must be clear on why the model chose one architecture to the other. Why might the model choose to include a skip-connection, for instance, or to suddenly increase the filters in the third layer.Future work should combine the concept of evolutionary logs where the system records its architectural adjustment and translates them into a plain language explanation, e.g.,The model introduced a generic attention layer to enhance focus.

8.3. Safety and Theoretical Guarantees

Self evolving systems run the risk of evolving in unexpected and unenviable ways. Should the "drift" be an intentional attack, the model could be encouraged to drift toward exposure to that attack. Developing theoretical constructs to bound the worst case outcomes of developing models is a necessary step. We suggest embedding formal procedure of verification within the loss function of the evolving model to prevent the evolved architecture from ignoring important safety limits, thus providing "guardrails" around the evolutionary pathway to reduce the possibility of the model evolving to an unsafe solution .

8.4. Decentralized and Federated Evolution

Present NAS frameworks usually assume access to centralized training data. However, in privacy critical domains like healthcare or distributed networks like smart cities, integrate data is often infeasible due to controlling or bandwidth constraints. Future iterations of SE-MLM must integrate with Federated Learning (FL). This would allow a fleet of edge devices to cumulatively search for the optimal architecture by sharing architecture gradients [15].

References

1. J. Brown et al., "Concept Drift Challenges in AI Systems," IEEE TNN, 2020. Available: https://ieeexplore.ieee.org/
2. S. Patel and R. Sharma, "Deep Learning Under Non-Stationary Data," ACM CSUR, 2021. Available: https://dl.acm.org/journal/csur
3. V. Shukla et al., "Agentic AI Framework for Autonomous and Self-Managing Cloud Services," IEEE SCEECS, 2026. DOI: 10.1109/SCEECS68810.2026.11429932
4. C. Finn et al., "Model-Agnostic Meta-Learning," ICML, 2017. Available: https://proceedings.mlr.press/v70/finn17a.html
5. J. Snell et al., "Prototypical Networks," NeurIPS, 2017. Available: https://papers.nips.cc/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html
6. B. Zoph and Q. Le, "Neural Architecture Search with RL," ICLR, 2017. Available: https://arxiv.org/abs/1611.01578
7. E. Real et al., "Evolution-Based NAS," AAAI, 2019. Available: https://aaai.org/ojs/index.php/AAAI/article/view/4405
8. K. Kirkpatrick et al., "Elastic Weight Consolidation," PNAS, 2017. Available: https://www.pnas.org/doi/10.1073/pnas.1611835114
9. A. Chaturvedi et al., "Three Party Key Sharing Protocol Using Polynomial Rings," IEEE UPCON, 2018. DOI: https://doi.org/10.1109/UPCON.2018.8596905
10. J. Yoon et al., "Dynamic Architectures for Lifelong Learning," CVPR, 2018. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Yoon_Lifelong_Learning_With_CVPR_2018_paper.html
11. H. Liu et al., "DARTS: Differentiable Architecture Search," ICLR, 2019. Available: https://arxiv.org/abs/1806.09055
12. S. Bender et al., "Once-for-All Networks," arXiv, 2020. Available: https://arxiv.org/abs/1908.09791

 doi:10.20944/preprints202604.1991.v1

13. J. Bergstra and Y. Bengio, "Random Search for Optimization," JMLR, 2012. Available: http://www.jmlr.org/papers/v13/bergstra12a.html
14. S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. Available: https://web.stanford.edu/~boyd/cvxbook/
15. Atul et al., "Federated Generative Intelligence for Explainable and Autonomous Cyber Defence in Critical Infrastructures," IEEE ISCS, 2025. DOI: 10.1109/ISCS69371.2025.11386415
16. M. Pedregosa, "Hyperparameter Optimization in ML," NeurIPS, 2016. Available: NeurIPS paper page
17. L. Franceschi et al., "Bilevel Programming for NAS," ICML, 2018. Available: https://arxiv.org/abs/1806.04910
18. A. Nichol et al., "First-Order Meta-Learning Methods," NeurIPS, 2018. Available: https://arxiv.org/abs/1803.02999
19. J. Gama et al., "Survey on Concept Drift Adaptation," ACM SIGKDD, 2014. Available: https://dl.acm.org/doi/10.1145/2523813.2523815
20. D. Kulkarni et al., "Statistical Drift Detection," KDD, 2020. Available: https://dl.acm.org/doi/10.1145/3394486.3403294
21. A. Krizhevsky, "CIFAR Dataset," Tech Report, 2009. Available: https://www.cs.toronto.edu/~kriz/cifar.html
22. V. Shukla et al., "Journey of Cryptocurrency in India in View of Financial Budget 2022–23," arXiv, 2022. DOI: https://doi.org/10.48550/arXiv.2203.12606
23. L. Wright et al., "Rotated-MNIST Drift Benchmark," arXiv, 2020. Available: https://arxiv.org/abs/2002.06740
24. B. Wu et al., "Quantization-Aware NAS," CVPR, 2020. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Wu_FBNetV3_Designing_Efficient_Deep_Networks_via_Dilly-Dally_Search_CVPR_2020_paper.html
25. M.K. Misra, A. Chaturvedi, S.P. Tripathi, V. Shukla, A unique key sharing protocol among three users using non-commutative group for electronic health record system, Journal of discrete mathematical sciences and cryptography, volume 22, issue 8, 2019.
26. A. Chaturvedi, V. Shukla, M.K. Misra, A random encoding method for secure data communication: an extension of sequential coding, Journal of discrete mathematical sciences and cryptography, volume 24, issue 5, 2021.
27. P. Li et al., "Adaptive Learning Under Distribution Shift," ICML, 2021. Available: https://proceedings.mlr.press/v139/li21h.html
28. Z. Luo et al., "Meta-NAS Benchmark Analysis," NeurIPS, 2021. Available: https://arxiv.org/abs/2110.05668