REVIEW ARTICLE

Machine Learning for Lyapunov Function Synthesis: A Comprehensive Review

Lanhao Zhao[1] **and Shuai Liu**[2,*]

[1] College of Information Science Technology, Beijing University of Technology, Beijing 100124, China
[2] School of Mathematics and Information Science, Yantai University, Yantai 264005, China

Abstract

Lyapunov functions underpin the stability analysis and controller synthesis for aerospace vehicles, from spacecraft attitude control to autonomous flight systems and UAV swarm coordination. This review discusses the development of computational Lyapunov function synthesis methods. Within this scope, it traces the evolution from traditional Sum-of-Squares (SOS) programming to artificial intelligence symbolic discovery technologies, prompted by the increasing complexity of nonlinear systems. Specifically, the following stages can be identified: computational relaxation methods constrained by polynomial templates; data-driven paradigms leveraging neural network-based empirical approximations; and the fusion of machine learning fitting and formal verification through the Counter-Example Guided Inductive Synthesis (CEGIS) framework. Subsequently, researchers developed construction-based network architectures and hybrid scalable verification mechanisms to address bottlenecks in computational verification. Interestingly, the use of generative models and reinforcement learning to explore analytically expressible expressions with physical interpretability is emerging as an active research area in this field. To date, these tools have been extended to various dynamic scenarios, such as stochastic systems, decentralized multi-agent topologies (e.g., unmanned aerial vehicle swarms and satellite constellations), safe reinforcement learning for autonomous flight control, and partial differential equation (PDE) solving. This review evaluates the aforementioned methodological stages, with a focus on analyzing the trade-offs among model expressivity, computational scalability, and the rigor of formal guarantees across different technical routes.

Keywords: lyapunov function, computational relaxation method, data-driven paradigms, reinforcement learning, generative models.

1 Introduction: From Manual Derivation to Automated Construction

1.1 Lyapunov Functions

Lyapunov's direct method is the foundation of stability analysis for nonlinear systems

$$\dot{x} = f(x),$$

Submitted: 29 May 2026
Revised: 16 July 2026
Accepted: 19 July 2026
Published: 11 August 2026

Vol. 1, **No.** 3, 2026.
10.62762/AEC.2026.822279

***Corresponding author:**
Shuai Liu
liushuai_qd@163.com

Citation
Zhao, L., & Liu, S. (2026). Machine Learning for Lyapunov Function Synthesis: A Comprehensive Review. *Aerospace Engineering Communications*, 1(3), 100–109.

where $x \in \mathbb{R}^n$ is the system state and $f : D \to \mathbb{R}^n$ is a locally Lipschitz continuous function defined on a domain $D \subseteq \mathbb{R}^n$ containing the origin, with $f(0) = 0$. Its core lies in constructing a scalar function $V(x)$ with certain properties; the existence of such a function serves as a sufficient condition for nonlinear system stability. As established in classical nonlinear systems theory [1], if the scalar function $V(x)$ is positive definite, i.e.,

$$V(0) = 0, \quad V(x) > 0, \quad \forall x \neq 0,$$

its Lie derivative along system trajectories is negative definite, i.e.,

$$\dot{V}(x) = \nabla V(x) \cdot f(x) < 0, \quad \forall x \neq 0,$$

and $V(x)$ satisfies the radial unboundedness condition

$$\lim_{\|x\| \to \infty} V(x) = \infty,$$

then the origin is globally asymptotically stable. Constructing the function $V(x)$ allows stability to be proved without explicitly solving the nonlinear differential equations. This approach provides formal robustness guarantees for closed-loop operations. These strict safety guarantees are particularly critical in aerospace engineering applications, including spacecraft attitude control, reusable launch vehicle recovery, and autonomous aircraft navigation [2, 3].

1.2 Main Challenges in Traditional Methods of Analysis and Computation

Although Lyapunov theory provides a rigorous theoretical foundation, finding a valid $V(x)$ for a given nonlinear system $f(x)$ remains computationally and analytically demanding. Historically, the construction of $V(x)$ has relied on the physical insight of domain experts.

However, this reliance on heuristics often reduces to trial and error, making the synthesis of Lyapunov functions for arbitrary nonlinear vector fields a challenging open problem [4, 5]. To systematize the search process, the control community developed computational relaxation methods, the most prominent of which is Sum-of-Squares (SOS) programming. SOS transforms the non-convex stability verification problem into a convex Semidefinite Programming (SDP) feasibility test by relaxing the positivity constraints into a search for polynomial sum-of-squares representations. Despite its adoption, SOS is constrained by three limitations:

- **Template dependency:** SOS requires the polynomial templates to be specified a priori. An inappropriate choice renders the resulting SDP infeasible, even when a valid Lyapunov function exists for the underlying system with actual stability properties.
- **Scalability and dimensionality:** For high-dimensional systems or high-degree polynomials, the scale of the resulting SDP grows exponentially with the combined state dimension and polynomial degree, leading to severe computational bottlenecks.
- **Non-polynomial systems:** When the dynamics $f(x)$ involve non-polynomial terms, SOS requires the introduction of auxiliary variables and equality constraints, which not only increase the problem size but also degrade numerical conditioning.

More fundamentally, Ahmadi et al. [6] demonstrated that there exist globally asymptotically stable polynomial vector fields that do not admit any finite-degree polynomial Lyapunov function. This result bounds the universality of SOS, forcing exploration within the broader function space of non-polynomials.

1.3 Machine Learning-Based Approaches

The aforementioned limitations provide a theoretical motivation for integrating Neural Networks (NN) and Machine Learning (ML). Supported by the Universal Approximation Theorem, Multi-Layer Perceptrons (MLPs) can approximate continuous functions on compact domains. This motivates a new approach: using ML parameterization to learn Lyapunov candidate functions directly from state-trajectory data or from the system's dynamical equations. This method aims to bypass the template restrictions of SOS while maintaining systematic automation. This review maps the logical evolution of this interdisciplinary field, as visualized in Figure 1: from initial empirical approximations, through the integration of precise formal verification, to interpretable symbolic certificates.

2 Early Attempts: Neural Networks as Empirical Approximators

2.1 Function Fitting Based on Trajectory Data

In the mid-to-late 2000s, researchers began attempting to parameterize Lyapunov functions directly via neural networks, primarily focusing on extracting stability patterns from state trajectories. Serpen [7] proposed an empirical approximation framework derived directly from the monotonically decreasing condition of Lyapunov functions. It employs an MLP (Lyapunov Neural Network) as the approximator. Given discrete

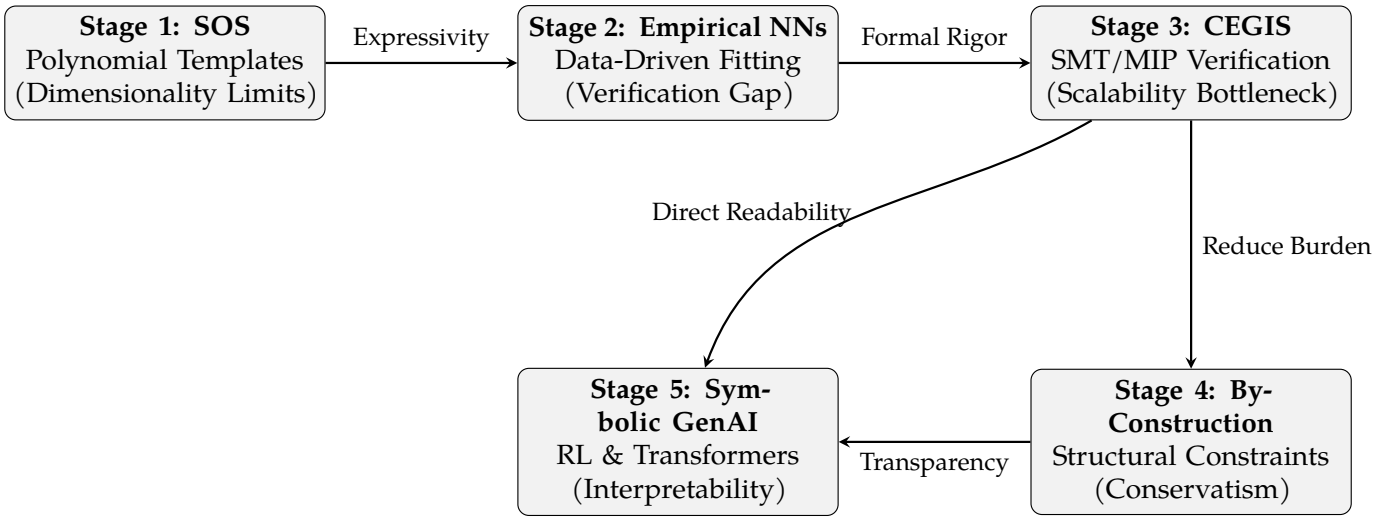

Figure 1. Evolutionary trajectory of computational Lyapunov function synthesis methods.

system states $x(k)$ and $x(k+1)$, the loss function enforces the decreasing property as follows:

$$L = \max(0, V(x(k+1)) - V(x(k))).$$

Network weights are updated via gradient descent to minimize a trajectory-based loss [7].

2.2 Constructive Algorithms

To improve training efficiency, Noroozi et al. [8] proposed structured data-generation algorithms:

- **Forward Method:** Initializes state sets far from the origin, iteratively training the network to satisfy $V(x_{k+1}) < V(x_k)$ along forward trajectories.
- **Backward Method:** Starts from $V(0) = 0$ and propagates backward along multiple trajectories, sequentially adding data points and assigning increasing V values.

2.3 Limitation: The Verification Gap

While these pioneering methods [7, 8] utilized neural networks to model dynamic properties, they present theoretical limitations. These methods are inherently data-driven, verifying the Lyapunov conditions only on a finite set of sampled points. They provide no deterministic guarantees for the continuous state space ($x \in D$) located between these discrete samples. Consequently, these empirical networks cannot serve as formal mathematical certificates, thereby creating a "verification gap" which subsequently drove the field toward synthesis architectures.

3 Neural Lyapunov Functions: Integrating Learning and Formal Verification

3.1 Counter-Example Guided Inductive Synthesis (CEGIS)

To resolve the verification gap, the research paradigm shifted toward rigorous construction, formally integrating the expressivity of deep learning with the strictness of automated theorem proving. The closed-loop framework, CEGIS [4, 9], decomposes the synthesis problem into iterative optimization against counterexamples, comprising a Learner and a Verifier.

- **Learner:** A neural network $V(x)$ optimizes a Lyapunov loss function over a finite sample set S:

$$L(V) = \mathbb{E}_{x \in S}[\max(0, -V(x)) + \max(0, \dot{V}(x))].$$

- **Verifier:** A formal verification tool analyzes the trained continuous function $V(x)$ over the entire compact domain D to find counterexamples (CEX), i.e., any state violating $V(x) > 0$ or $\dot{V}(x) < 0$.
- **Iteration:** If a counterexample is found, it is appended to the training set ($S \leftarrow S \cup \{\text{CEX}\}$), and the Learner retrains. If no counterexample exists globally, the function is formally certified.

3.2 Pioneering Work for Neural Lyapunov Control

Chang et al. [4] introduced a CEGIS framework for neural Lyapunov control, using Satisfiability Modulo Theories (SMT) solvers. Unlike empirical sampling, SMT solvers conduct formal verification on logical formulas involving nonlinear real arithmetic over continuous domains. Abate et al. [9] extended this approach using polynomial activation functions, simplifying the

SMT verification task. Dai et al. [10] and Wu et al. [11] applied similar logic to discrete-time systems using Mixed-Integer Programming (MIP) verifiers.

3.3 The Scalability Bottleneck

Despite being theoretically sound and complete, standard CEGIS is limited by verification bottlenecks. There is a computational mismatch: while neural network training is scalable, formal verification (via exact SMT/MIP combinatorics) is NP-hard. Literature confirms that SMT and MIP solvers typically face significant scaling challenges beyond networks with 30 to 200 neurons, largely depending on network architecture, system dimensionality, and specific solver settings [4, 9, 12]. This forces a trade-off: network expressivity sufficient to handle nonlinearities results in unverifiable certificates. Overcoming this bottleneck without sacrificing mathematical rigor has become the primary challenge in this field.

4 Addressing the Bottleneck: Hybrid Verification and Structural Constraints

4.1 Hybrid and Scalable Verification: Shifting the Bottleneck

Another path aims to accelerate the verifier itself. Zhao et al. [13] integrated neural network learning with classical SOS tools. By constraining the Learner to output polynomial functions, the verification stage is transformed from an NP-hard SMT query over the reals into a tractable Linear Matrix Inequality (LMI) feasibility test. While this leverages the polynomial-time solvability of LMI, it merely shifts the computational burden. For control practitioners, the reality of SOS programming faces the dimensionality issue: the scale of the underlying Semidefinite Programming (SDP) grows combinatorially, with the Gram matrix size scaling as $\binom{n+d}{d}$, where n is the state dimension and d is the Lyapunov polynomial half-degree. In practical numerical implementations, scaling this framework to higher-dimensional systems (e.g., beyond 10 dimensions) or searching for high-degree certificates frequently encounters out-of-memory (OOM) issues or solver timeouts in commercial tools like MOSEK, though this computational complexity is heavily influenced by system sparsity and solver configurations. Furthermore, by stripping the neural network of its nonlinear activation functions to cater to the SOS verifier, this methodology regresses to the original polynomial template restrictions.

To avoid the reliance on polynomial forms and the dimensionality constraints in the aforementioned SOS/LMI verification, another technical route abandons algebraic transformation and attempts to directly relax the strictness of formal verification itself. Yang et al. [14] replaced exact solvers with bound propagation techniques, rapidly estimating linear upper and lower bounds for nonlinear activations. While this permits networks exceeding 100 neurons, such bound relaxation techniques introduce conservatism. Calculating strict Lipschitz constants for deep, non-smooth networks remains computationally demanding, limiting the precision of formal bounds. Furthermore, geometric reachability tools like NNV [15] compute over-approximated reachable sets using zonotopes or star-sets, providing a robust but conservative alternative to point-wise exact solvers.

4.2 By-Construction Architectures

Unlike approaches that aim to accelerate external formal verifiers, architectural methods instead seek to eliminate portions of the verification burden by embedding mathematical constraints into the forward propagation of neural networks. Since Lyapunov functions require simultaneous satisfaction of positive definiteness ($V(x)>0$) and negative Lie derivatives along trajectories ($\dot{V}(x)<0$), by-construction architectures natively satisfy the former by modifying the network structure. Specifically, this idea has implementations in different works, but each introduces distinct drawbacks:

- **Quadratic Networks and Structured Offsets (Lyapunov-Net):** Richards et al. [16] proposed a specific form of quadratic network structure; subsequently, Gaby et al. [17] proposed Lyapunov-Net, whose core is adding an unconstrained neural network offset (ensured to be non-negative via activation functions) onto a globally positive definite scaffold function. This method decouples the positive definiteness requirement from optimization and verification. The formal verification of the derivative condition (negative Lie derivative) is not simplified. For high-dimensional systems, proving the negative definiteness of nonlinear derivatives remains a computational burden.
- **Input Convex Neural Networks (ICNN):** To address the challenge of derivative verification, Kolter and Manek [18] parameterized Input Convex Neural Networks (ICNN) to jointly learn system dynamics and certificates. The architecture of ICNN requires the network weights to be non-negative, which limits the fitting capability of the neural network, resulting in limitations when

modeling nonlinear complex dynamic systems.

- **Projected Dynamics:** A rigorous approach by Min et al. [19] directly projects the nominal model onto a predefined stability manifold, ensuring that the generated dynamics are inherently stable. Although this "stable-by-design" framework bypasses post-hoc verification, it constrains the expressive parameterization of the system dynamics model itself. Forcing the network to adhere to specific manifolds leads to conservative control strategies, reducing responsiveness to nonlinearities.

The architectures discussed above primarily focus on low-dimensional or nominal deterministic systems. To push neural Lyapunov methods from theory to physical deployment, research must address the dimensionality constraints, time-varying stochastic disturbances, and hardware safety constraints. The next section explores how machine learning integrates with advanced mathematical theories to tackle these practical challenges.

5 Emerging Frontiers: Symbolic Discovery and Interpretability

Although neural architectures have successfully extended Lyapunov synthesis to complex, high-dimensional, and safety-critical control scenarios, their black-box nature introduces a new set of challenges. Neural certificates, parameterized by dense continuous weights and nonlinear activations, obscure the underlying mathematical principles and physical insight of the system's stability. Furthermore, computing accurate formal bounds for these dense networks remains computationally intensive. These factors limit their broad acceptance in rigorous engineering domains where practitioners favor transparent and human-readable certificates. Consequently, the research focus is experiencing a paradigm shift: transitioning from parameterizing continuous black-box networks toward the automated discovery of classical, analytical symbolic formulas.

By reformulating the problem as symbolic regression, Alfarano et al. [5] trained sequence-to-sequence (seq2seq) Transformer models on synthetic datasets to predict global Lyapunov functions. However, this supervised pre-training may limit its generalizability to novel real-world systems characterized by local stability.

To eliminate dataset dependency, Zou et al. [20] deployed a model-free reinforcement learning generator to autoregressively propose symbolic formulas, evaluating them via numerical global optimizers. A key insight of this approach is the necessity of a risk-seeking policy gradient to isolate valid certificates. However, framing symbolic regression as an RL problem introduces theoretical and computational challenges. The mathematical symbolic space is inherently discrete and non-smooth, characterized by reward discontinuities—a single mutational step from x^2 to x^3 can violate the negative definiteness of the Lie derivative, resulting in a reward drop. Optimizing over this non-smooth space using gradient-based RL yields suboptimal sample efficiency and elevated training variance. Furthermore, calculating rewards for proposed formulas during training requires invoking global optimizers, creating a complex bilevel optimization loop. While effective for low-dimensional benchmarks, scaling this architecture constitutes a computational bottleneck, and lacks theoretical guarantees of algorithmic convergence [20].

From the perspective of generative models, current works directly generating analytical Lyapunov functions remain scarce. One of the reasons lies in the mismatch between the generative paradigm and the verification space.

Currently popular generative architectures, such as Diffusion Models [21] and Flow Matching Models [22], excel at processing probability distributions in continuous spaces, effectively capturing smooth trajectories, but they struggle to directly construct mathematical symbolic formulas with discrete syntax tree structures. Conversely, Transformer architectures are adept at handling discrete token sequences and topological syntax trees, making them a suitable choice for symbolic discovery. However, the validity of a Lyapunov function must ultimately be formally verified in a continuous state space through continuous differential inequalities (e.g., negative definiteness of the Lie derivative). This contradiction of using an architecture suited for discrete sequences to generate analytical formulas that must satisfy continuous physical constraints dictates that minor perturbations in the discrete symbolic tree can trigger variations in continuous properties. This is the underlying reason why the control community often favors continuous networks like MLPs for parameterization, whereas methods for directly generating analytical symbolic formulas remain relatively rare.

Further expanding mathematical discovery, the integration of Large Language Models (LLMs) with automated program evaluators presents a heuristic evolu-

tionary search mechanism for proposing novel analytical structures [23].

6 From Theory to Complex Scenarios: High-Order Dynamics and Physical Deployment

6.1 Stochastic and Hybrid Dynamic Systems

Real-world physical systems are affected by noise and system mode switching. Extending neural Lyapunov methods to these temporal and stochastic dynamics is essential for physical deployment.

- **Stochastic Dynamical Systems:** Traditional Lyapunov methods struggle to compute expectations analytically when dealing with high-dimensional Brownian motion noise. To handle this uncertainty, synthesis frameworks incorporate Itô calculus to fit stochastic Lyapunov functions capable of satisfying the negative definiteness of the infinitesimal generator via neural networks, thereby providing mean-square asymptotic stability guarantees for Stochastic Differential Equations (SDEs) [24].
- **Switched and Hybrid Systems:** For nonlinear systems that discretely switch between different dynamic modes, a single Lyapunov function typically does not exist. Researchers utilize neural networks to construct multiple Lyapunov functions, jointly optimizing dwell-time parameterization conditions via reinforcement learning or gradient descent. This method yields the control law for an unknown switching nonlinear system and provides a common stability region that guarantees global hybrid stability [25].

6.2 High-Dimensional and Decentralized Multi-Agent Systems

As the number of agents increases and state dimensions grow, centralized verification faces dimensionality issues. Therefore, research focus has shifted to distributed and decentralized neural network architectures.

- **Dimensionality Reduction Decomposition and Small-Gain Theory:** For large-scale interconnected systems, training a global neural network is impractical. Researchers utilize the small-gain theorem to develop compositional vector neural Lyapunov functions. This method decomposes the overall state space into multiple low-dimensional, locally verifiable subsystems. Each subsystem is handled by an independent lightweight neural network, and global stability is proven by analyzing the nonlinear interconnection gains between subsystems [26, 27].
- **Graph Neural Networks (GNN) and Swarm Intelligence:** In decentralized multi-agent systems, the communication topology among agents is time-varying. To address this, GNNs are embedded into the synthesis process. Leveraging their permutation invariance and local aggregation properties, GNNs can extract graph topology features. Local Lyapunov networks and distributed control protocols are co-parameterized via GNNs to ensure that Unmanned Aerial Vehicles (UAVs), satellite constellations, and Autonomous Underwater Vehicles (AUVs) form collision-free stable formation consensus under local communication constraints [28–30].

Although GNNs provide scale-free deployment for decentralized architectures, an analytical gap remains. In these decentralized neural architectures, it is currently difficult to establish formal bounds against worst-case scenarios. For example, when facing adversarial topology switching or Byzantine agents, local neural Lyapunov functions are difficult to compositionally prove in formal verifiers through algebraic graph theory. Recently, learning-based Lyapunov methods have shown significant potential in addressing these uncertainties specifically for aerospace systems, such as robust UAV swarm coordination and spacecraft formation consensus.

6.3 Analytical PDEs, Geometry, and Optimal Control

Deep learning architectures are integrated to map mathematical theories to physical deployment. Physics-Informed Neural Networks (PINNs) frame synthesis construction as an unconstrained optimization over the Zubov partial differential equation to estimate regions of attraction [31]. While Implicit Neural Representations (SIRENs) capture high-frequency variations to yield sharper empirical level sets [32], the formal verification of architectures with periodic activation functions remains unresolved. For optimal control, DeepReach parameterizes Hamilton-Jacobi-Bellman value functions to compute "reach-avoid" sets for high-dimensional systems [33]. Koopman operator theory lifts nonlinear dynamics into high-dimensional linear spaces to simplify structural synthesis [34], while contraction theory facilitates the synthesis of differential Lyapunov metrics for trajectory tracking [35].

Table 1. Comparison of main approaches for neural Lyapunov functions.

Methodology	Key Paper(s)	Function Form	Training / Verification	Core Advantage	Key Limitation
Empirical Approx.	Serpen [7]; Noroozi et al. [8]	NN (MLP)	Trained based on sampled trajectories.	Simple, model-free, data-driven.	No formal guarantees; suffers from the verification gap.
CEGIS + SMT/MIP	Chang et al. [4]; Abate et al. [9]	NN (Standard or Polynomial)	Learner (NN) + Verifier (SMT/MIP).	End-to-end automated formal guarantees.	NP-hard verification bottleneck; lack of scalability.
CEGIS + SOS	Zhao et al. [13]	Polynomial NN	Learner (NN) + Verifier (SOS/LMI).	Formal guarantees; avoids exact SMT computation.	Reintroduces the restriction to polynomials.
By-Construction	Gaby et al. [17]	Structured NN	Offset term plus positive definite scaffold.	Positive definite by design; reduces verification load.	Derivative condition still requires computational load.
Stable-by-Design	Kolter and Manek [18]; Min et al. [19]	ICNN / Projected Dynamics	Jointly learn dynamics and certificate.	Inherent stability; reduces verification burden (but limits model expressivity).	Restricts the expressive parameterization of system dynamics.
PINN / Zubov	Liu et al. [31]	PINN	Minimizes PDE (Zubov) residuals.	Directly estimates exact ROA.	Formal verification of high-frequency PDE solvers remains unresolved.
Symbolic (Pre-train)	Alfarano et al. [5]	Analytical Symbolic	Transformer (Seq2Seq).	High interpretability; discovers global functions.	Over-reliance on large-scale datasets; not applicable to local stability.
Symbolic (RL)	Zou et al. [20]	Analytical Symbolic	Transformer + Risk-Seeking RL.	Interpretability; no dataset requirement.	Sample inefficiency; lacks convergence guarantees in discontinuous spaces.

6.4 Robustness and Safety-Critical Control (CLF and CBF)

While nominal stability provides a theoretical baseline, physical deployment exposes systems to bounded disturbances and unmodeled dynamics. Early neural synthesis frameworks focused on nominal global asymptotic stability, often lacking mechanisms to guarantee Input-to-State Stability (ISS) or to synthesize Control Lyapunov Functions (CLFs). The concept of CLF extends traditional Lyapunov methods by directly incorporating the control input, allowing for the simultaneous synthesis of the verifiable certificate and a stabilizing control law.

To address this issue, progress has shifted toward embedding Lipschitz continuity and ISS guarantees directly into neural architectures [28, 36]. Researchers enforce Lipschitz bounds on network layers and learned dynamic models. By doing so, the synthesis of incremental ISS (δ-ISS) certificates and robust CLFs can be framed as an optimization problem. This method mathematically bounds the $\mathcal{L}_\infty$ gain from external disturbances to state deviations, embedding disturbance attenuation directly into the training loss [37].

Building upon these robust CLFs, mathematical stability frameworks naturally integrate Neural Control Barrier Functions (CBFs) to formulate a unified safety-critical control framework. While CLFs ensure asymptotic stability (reaching a goal state), CBFs enforce predefined safe sets to guarantee forward invariance (avoiding obstacles) [38–40]. In reinforcement learning, these jointly verified CLF-CBF certificates act as differentiable optimization shields [12, 41] or structural penalties [42] to guarantee safe exploration. To bridge the sim-to-real gap, current pipelines deploy distributionally robust chance constraints [37], Gaussian processes [38], randomized smoothing [36], end-to-end safe reinforcement learning via barrier functions [43], and online adaptive control laws [28, 44] to bound epistemic uncertainties.

7 Conclusion and Future Outlook

7.1 Summary of Development Evolution

This review synthesizes the evolutionary trajectory of computational Lyapunov function synthesis.

The field transitioned from Stage 1 (pre-2005) utilizing SOS polynomial templates with dimensionality limits [6], to Stage 2 (2005-2008) employing empirical approximations without formal guarantees [7, 8]. Stage 3 (2019-2021) introduced CEGIS frameworks providing exact SMT verification but suffering from scalability bottlenecks [4, 9]. To reduce the verification burden, Stage 4 (2019-2025) developed by-construction architectures and scalable bounds [13, 17]. Finally, Stage 5 (2024-2025) shifted toward the symbolic discovery of analytical formulas using Transformers and generative AI to enhance interpretability [5, 20].

7.2 Methodological Comparison

Table 1 summarizes the main methods, delineating their verification mechanisms and structural limitations.

7.3 Challenging and Open-Ended Questions

While the integration of machine learning has proven the feasibility of automated Lyapunov function synthesis, scaling these methods to industrial and complex

physical systems requires addressing several mathematical and computational challenges. Future research should address the following open-ended questions:

- **Mitigating SDP Dimensionality via Sparsity and Composition:** Transforming neural outputs into polynomial templates merely shifts the verification bottleneck to Semidefinite Programming (SDP). For systems beyond moderate dimensions ($n > 10$), the Gram matrix size scales combinatorially as $\binom{n+d}{d}$, growing rapidly with both state dimension and polynomial degree. To overcome this curse of dimensionality, future neural-SOS frameworks must integrate algebraic simplifications, such as chordal sparsity and facial reduction. Furthermore, rather than pursuing a monolithic global certificate, architectures should pivot to synthesizing piecewise neural Lyapunov functions, utilizing switching logic to mathematically compose local stability guarantees across the entire state space.
- **Stabilizing Symbolic Discovery with Neurosymbolic Priors:** Applying model-free reinforcement learning directly to symbolic regression exposes the search process to discontinuous reward distributions. A mutation in mathematical operators (e.g., from x^2 to x^3) can cause Lie derivatives to diverge, resulting in sample inefficiency. To stabilize this process, the field must transition toward neurosymbolic architectures. Specifically, large generative models should serve as structural priors to propose candidate formulas, while look-ahead algorithms such as Monte Carlo Tree Search (MCTS) conduct the structural pruning and verification, avoiding the variance issues of pure policy gradients in discrete mathematical spaces.
- **Formal Bounds for Dynamic Multi-Agent Topologies:** While Graph Neural Networks (GNNs) offer empirical zero-shot scalability to decentralized swarms, they lack mathematical mechanisms to compute robust stability bounds against worst-case scenarios. The analytical challenge lies in formally verifying decentralized neural Lyapunov functions under topology switching and Byzantine node failures. This requires seamlessly fusing local dissipativity storage functions directly with time-varying algebraic graph theory within the neural verification loop.
- **Real-Time Computational Feasibility and Sim-to-Real Deployment:** Neural certificates validated in idealized continuous simulations fail during hardware deployment due to non-parametric disturbances (e.g., unmodeled friction, aerodynamic drag). Theoretical guarantees are insufficient if neural control laws cannot be evaluated within the microsecond constraints of real-time embedded hardware. Future work must focus on bounding the Lipschitz constants of deep networks to provide robust margins, ensuring that the synthesized certificates remain valid under real-world computational latency and sensor noise.
- **Mechanical Verification via Interactive Theorem Provers:** Current verification relies on SMT solvers (which obscure their internal logic) or SOS relaxations (depending on numerical solver tolerances). To eliminate the risk of numerical artifacts, the field must shift from numerical verification to formal logical deduction. Future generative architectures should output not only analytical candidate functions but also their corresponding proof scripts. Connecting neural synthesizers directly to interactive theorem provers (such as Lean 4 or Coq) to mechanically certify positive definiteness and derivative conditions remains an unresolved objective for safety-critical control.

Data Availability Statement

Not applicable.

Funding

This work was supported without any funding.

Conflicts of Interest

Lanhao Zhao served as an Editorial Board Member of the *Aerospace Engineering Communications* at the time of manuscript submission. To ensure the integrity of the peer-review process, Lanhao Zhao was not involved in the editorial handling, peer review, or decision-making process for this manuscript, which was handled independently by another editor. The remaining author declares no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Khalil, H. K. (2002). *Nonlinear systems* (3rd ed.). Prentice Hall. ISBN: 978-0-13-067389-3.

[2] Wu, Y. Y., Zhang, Y., & Wu, A. G. (2022). Finite-time attitude stabilization of rigid spacecrafts based on control Lyapunov functions. *IET Control Theory & Applications, 16*(7), 663-673. [CrossRef]

[3] Li, B., Wen, S., Yan, Z., Wen, G., & Huang, T. (2023). A survey on the control lyapunov function and control barrier function for nonlinear-affine control systems. *IEEE/CAA Journal of Automatica Sinica, 10*(3), 584-602. [CrossRef]

[4] Chang, Y. C., Roohi, N., & Gao, S. (2019). Neural lyapunov control. *Advances in neural information processing systems, 32*. https://proceedings.neurips.cc/paper_files/paper/2019/hash/2647c1dba23bc0e0f9cdf75339e120d2-Abstract.html

[5] Alfarano, A., Charton, F., & Hayat, A. (2024). Global lyapunov functions: a long-standing open problem in mathematics, with symbolic transformers. *Advances in Neural Information Processing Systems, 37*, 93643-93670.

[6] Ahmadi, A. A., Krstic, M., & Parrilo, P. A. (2011, December). A globally asymptotically stable polynomial vector field with no polynomial Lyapunov function. In *2011 50th IEEE Conference on Decision and Control and European Control Conference* (pp. 7579-7580). IEEE. [CrossRef]

[7] Serpen, G. (2005, July). Empirical approximation for Lyapunov functions with artificial neural nets. In Proceedings. *2005 IEEE International Joint Conference on Neural Networks, 2005.* (Vol. 2, pp. 735-740). IEEE. [CrossRef]

[8] Noroozi, N., Karimaghaee, P., Safaei, F., & Javadi, H. (2008). Generation of Lyapunov functions by neural networks. In *Proceedings of the World Congress on Engineering* (Vol. 2008). https://www.iaeng.org/publication/WCE2008/WCE2008_pp61-65.pdf

[9] Abate, A., Ahmed, D., Giacobbe, M., & Peruffo, A. (2020). Formal synthesis of Lyapunov neural networks. *IEEE Control Systems Letters, 5*(3), 773-778. [CrossRef]

[10] Dai, H., Landry, B., Yang, L., Pavone, M., & Tedrake, R. (2021). Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*. [CrossRef]

[11] Wu, J., Clark, A., Kantaros, Y., & Vorobeychik, Y. (2023). Neural lyapunov control for discrete-time systems. *Advances in neural information processing systems, 36*, 2939-2955.

[12] Dawson, C., Gao, S., & Fan, C. (2023). Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics, 39*(3), 1749-1767. [CrossRef]

[13] Zhao, H., Qi, N., Ren, M., Liu, B., Shi, S., & Yang, Z. (2025). Learning-enabled Polynomial Lyapunov Function Synthesis via High-Accuracy Counterexample-Guided Framework. In *Proceedings of the Computer Vision and Pattern Recognition Conference* (pp. 10275-10284). [CrossRef]

[14] Yang, L., Dai, H., Shi, Z., Hsieh, C. J., Tedrake, R., & Zhang, H. (2024). Lyapunov-stable neural control for state and output feedback: A novel formulation. *arXiv preprint arXiv:2404.07956*. [CrossRef]

[15] Tran, H. D., Yang, X., Manzanas Lopez, D., Musau, P., Nguyen, L. V., Xiang, W., ... & Johnson, T. T. (2020, July). NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *International conference on computer aided verification* (pp. 3-17). Cham: Springer International Publishing. [CrossRef]

[16] Richards, S. M., Berkenkamp, F., & Krause, A. (2018, October). The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on robot learning* (pp. 466-476). PMLR.

[17] Gaby, N., Zhang, F., & Ye, X. (2022, December). Lyapunov-net: A deep neural network architecture for Lyapunov function approximation. In *2022 IEEE 61st Conference on Decision and Control (CDC)* (pp. 2091-2096). IEEE. [CrossRef]

[18] Kolter, J. Z., & Manek, G. (2019). Learning stable deep dynamics models. *Advances in neural information processing systems, 32*.

[19] Min, Y., Richards, S. M., & Azizan, N. (2023, December). Data-driven control with inherent lyapunov stability. In *2023 62nd IEEE Conference on Decision and Control (CDC)* (pp. 6032-6037). IEEE. [CrossRef]

[20] Zou, H., Feng, J., Zhao, H., & Shi, Y. (2025). Analytical lyapunov function discovery: An rl-based generative approach. *arXiv preprint arXiv:2502.02014*. [CrossRef]

[21] Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems, 33*, 6840-6851.

[22] Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., & Le, M. (2022). Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*. [CrossRef]

[23] Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., ... & Fawzi, A. (2024). Mathematical discoveries from program search with large language models. *Nature, 625*(7995), 468-475. [CrossRef]

[24] Zhang, J., Zhu, Q., & Lin, W. (2022). Neural stochastic control. *Advances in neural information processing systems, 35*, 9098-9110.

[25] Dey, B. S., Basu, A., & Jagtap, P. (2026). Input-to-State Stabilizing Neural Controllers for Unknown Switched Nonlinear Systems within Compact Sets. *arXiv preprint arXiv:2601.14643*. [CrossRef]

[26] Grüne, L. (2021). Computing Lyapunov functions using deep neural networks. *Journal of Computational Dynamics, 8*(2), 131-152. [CrossRef]

[27] Liu, J., Meng, Y., Fitzsimmons, M., & Zhou, R. (2024, July). Compositionally verifiable vector neural Lyapunov functions for stability analysis of interconnected nonlinear systems. In *2024 American Control Conference (ACC)* (pp. 4789-4794). IEEE. [CrossRef]

[28] Dawson, C., Qin, Z., Gao, S., & Fan, C. (2022, January). Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning* (pp. 1724-1735). PMLR.

[29] Jena, A., Huang, T., Sivaranjani, S., Kalathil, D., & Xie, L. (2022). Distributed learning of neural Lyapunov functions for large-scale networked dissipative systems. *arXiv preprint arXiv:2207.07731*. [CrossRef]

[30] Qin, Z., Zhang, K., Chen, Y., Chen, J., & Fan, C. (2021). Learning safe multi-agent control with decentralized neural barrier certificates. *arXiv preprint arXiv:2101.05436*. [CrossRef]

[31] Liu, J., Meng, Y., Fitzsimmons, M., & Zhou, R. (2025). Physics-informed neural network Lyapunov functions: PDE characterization, learning, and verification. *Automatica, 175*, 112193. [CrossRef]

[32] Sitzmann, V., Martel, J., Bergman, A., Lindell, D., & Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Advances in neural information processing systems, 33*, 7462-7473.

[33] Bansal, S., & Tomlin, C. J. (2021, May). Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1817-1824). IEEE. [CrossRef]

[34] Zinage, V., & Bakolas, E. (2023). Neural koopman lyapunov control. *Neurocomputing, 527*, 174-183. [CrossRef]

[35] Tsukamoto, H., Chung, S. J., & Slotine, J. J. E. (2021). Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview. *Annual Reviews in Control, 52*, 135-169. [CrossRef]

[36] Long, K., Yi, Y., Cortés, J., & Atanasov, N. (2023, June). Distributionally robust Lyapunov function search under uncertainty. In *Learning for Dynamics and Control Conference* (pp. 864-877). PMLR.

[37] Xiong, Z., Eappen, J., Qureshi, A. H., & Jagannathan, S. (2022, October). Model-free neural lyapunov control for safe robot navigation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5572-5579). IEEE. [CrossRef]

[38] Berkenkamp, F., Turchetta, M., Schoellig, A., & Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems, 30*. https://proceedings.neurips.cc/paper_files/paper/2017/hash/766ebcd59621e305170616ba3d3dac32-Abstract.html

[39] Robey, A., Hu, H., Lindemann, L., Zhang, H., Dimarogonas, D. V., Tu, S., & Matni, N. (2020, December). Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)* (pp. 3717-3724). IEEE. [CrossRef]

[40] Hu, H., Yang, Y., Wei, T., & Liu, C. (2025, January). Verification of Neural Control Barrier Functions with Symbolic Derivative Bounds Propagation. In *Conference on Robot Learning* (pp. 1797-1814). PMLR.

[41] Choi, J., Castaneda, F., Tomlin, C. J., & Sreenath, K. (2020). Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. *arXiv preprint arXiv:2004.07584*. [CrossRef]

[42] So, O., Serlin, Z., Mann, M., Gonzales, J., Rutledge, K., Roy, N., & Fan, C. (2024, May). How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. In *2024 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 11532-11539). IEEE. [CrossRef]

[43] Cheng, R., Orosz, G., Murray, R. M., & Burdick, J. W. (2019, July). End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, No. 01, pp. 3387-3395). [CrossRef]

[44] Umlauft, J., & Hirche, S. (2017, July). Learning stable stochastic nonlinear dynamical systems. In *International Conference on Machine Learning* (pp. 3502-3510). PMLR.

Lanhao Zhao received the B.S. degree in automation from Shandong University of Science and Technology, Taian, China, in 2018, and the M.S. degree in systems science from Qingdao University, Qingdao, China, in 2022. He is currently pursuing the Ph.D. degree in control science and engineering at Beijing University of Technology, Beijing, China. His research interests include multi-agent systems and complex networks. He serves as a reviewer for journals including IEEE TCST, TASE, and TMM, and is a member of the Popularization Working Committee of the Chinese Association of Automation (CAA). (Email: lugh56.2007@163.com)

Shuai Liu received the B.E. degree in automation from Qufu Normal University, Rizhao, China, in 2017, the M.S. degree in control science and engineering from Qingdao University, Qingdao, China, in 2020, and the Ph.D. degree in electronic information from Dalian University of Technology, Dalian, China, in 2026. Since 2026, he has been with the School of Mathematics and Information Science, Yantai University, Yantai, China. His current research interests include multi-agent systems, networked control systems, and distributed algorithm design.