

Article

Compositional Formal Verification of Anti-Lock Braking System Neural Controllers

Huixing Fang 

School of Information Engineering, Huangshan University, 39 Xihai Road, Huangshan 245041, China;
fanghuixing@hotmail.com

Abstract

We present a compositional formal verification framework that establishes floating-point correctness and control safety for anti-lock braking system (ABS) neural controllers, using Rocq and Flocq as a single verification toolchain. We decompose the verification into three independent components. First, a Flocq proof bounds each binary32 operation to machine epsilon. Second, a CoqInterval-verified barrier certificate enforces eight pointwise conditions across four road surfaces, keeping the slip ratio within $[0.03, 0.40]$. Third, an ODE forward invariance lemma lifts these pointwise conditions to a temporal guarantee via the suprema axiom and ε - δ continuity arguments. The compositional theorem combines all three layers into an end-to-end safety proof. Our Rocq development (866 lines, 7 modules) compiles successfully: the core floating-point lemmas are Qed, with one lemma remaining Admitted. Numerical simulation across four road surfaces (after 0.3 s settling, $v \geq 1.5$ m/s) shows that the neural network controller achieves 100% closed-loop safety on every surface, including ice, outperforming the classical finite state machine controller (96.0–100%). The framework demonstrates that single-toolchain compositional verification of floating-point neural controllers is feasible, directly supporting ISO 26262 certification for safety-critical automotive systems. The verified barrier windows and open-source Rocq artifacts provide a reusable foundation for future neural ABS verification work.

Keywords: formal verification; neural network verification; anti-lock braking system; Flocq; floating-point arithmetic; barrier certificate; Rocq theorem prover; compositional verification

1. Introduction

1.1. Motivation and Problem Statement

The rapid advancement of artificial intelligence has led to the widespread deployment of neural network (NN) controllers in safety-critical cyber-physical systems, including autonomous vehicles, aircraft control, and medical devices. Anti-lock braking systems (ABS), which are mandatory safety features in all modern passenger vehicles, represent a particularly compelling application domain for neural control. An ABS must prevent wheel lock during emergency braking by modulating brake pressure at each wheel, thereby maintaining steering control and minimizing stopping distance across varying road conditions.

Traditional ABS controllers employ rule-based algorithms with hand-tuned parameters [1]. While effective, these controllers cannot adapt to changing road friction conditions without explicit friction estimation. The recent work by Piccinini et al. [2] introduces a model-structured neural network (MS-NN) architecture that achieves significant improvements—braking dis-



Academic Editors: Reiko Heckel and
George Angelos Papadopoulos

Received: 29 June 2026

Revised: 28 July 2026

Accepted: 4 August 2026

Published: 6 August 2026

Copyright: © 2026 by the author.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

tances reduced by 3.0% to 40.4% compared to standard ABS—but relies solely on simulation-based validation.

The fundamental challenge in deploying NN controllers in safety-critical systems is **formal verification**: providing mathematical guarantees that the system satisfies safety properties under all operating conditions. For NN-controlled hybrid systems like ABS, this requires addressing two orthogonal concerns simultaneously: (1) the floating-point implementation errors introduced when the NN executes on embedded hardware with IEEE 754 arithmetic, and (2) the continuous-state safety properties of the controlled physical system.

1.2. Limitations of Existing Approaches

Existing approaches to hybrid system verification typically require separate tools for each concern. Isabelle/HOL [3] provides ODE reasoning capabilities for continuous dynamics. KeYmaera X [4,5] offers differential dynamic logic (dL) for hybrid systems. On the NN side, tools such as α, β -CROWN [6] and Marabou [7] provide robustness bounds on ideal real-number models. However, none of these tools account for the floating-point rounding errors that occur in actual hardware implementations.

The Flocq library [8,9] provides a comprehensive formalization of IEEE 754 floating-point arithmetic within the Coq/Rocq theorem prover [10]. Flocq enables precise error bounding for individual floating-point operations through lemmas such as `error_1e_half_ulp`, but has rarely been applied to end-to-end neural network controller verification.

1.3. Contributions

This paper presents a **compositional formal verification framework** for MS-NN ABS controllers, conducted entirely within the Rocq theorem prover. Our contributions are:

- (1) **Single-Toolchain Architecture.** We perform both NN verification and hybrid system safety verification within Rocq, eliminating the trusted bridge between separate tools and reducing the trusted computing base.
- (2) **Complete Flocq Floating-Point Proof.** We provide machine-checked proofs in Rocq that the binary32 per-operation error is bounded by $\varepsilon_{\text{machine}} = 2^{-23}$. The core lemmas `round_error_bound`, `ulp_bound`, `add_error_bound`, and `mul_error_bound` are all proved as **Qed** theorems through Flocq's `FLT.ulp_FLT_1e` and a 14-step algebraic decomposition.
- (3) **Barrier Certificate for FSM Safety.** We prove barrier conditions at the slip interval boundaries for all four road surfaces. At the lower boundary $\lambda = 0.03$, the slip dynamics are directed away from wheel lock on dry, wet, snow, and ice. At the upper boundary $\lambda = 0.40$, the dynamics are directed away from excessive slip on all four surfaces. All eight conditions are verified via CoqInterval pointwise evaluation (`Barrier_interval.v`, 56 lines, **Qed**). Combined with the `last_crossing_point` lemma (`Barrier_ode.v`, **Qed**), this establishes forward invariance of $\lambda \in [0.03, 0.40]$ on all four surfaces.
- (4) **Real-Analysis ODE Forward Invariance Lemma.** We provide a machine-checked proof (`Barrier_ode.v`, 132 lines) of the `last_crossing_point` lemma: for a continuous function g with $g(0) \geq 0$ and $g(t) < 0$, there exists a last zero-crossing $t_0 \in [0, t]$ such that $g(t_0) = 0$ and $g(s) < 0$ for all $s \in (t_0, t]$. The proof relies on the supremum axiom (completeness) and continuity theorems from the standard Reals library.
- (5) **Numerical Validation.** We implement the quarter-car ABS model in Python 3.13.12 and compare the PD oracle, FSM, and NN controllers across four road surfaces. After excluding the 0.3 s settling period and low-speed regime ($v < 1.5$ m/s), the NN achieves 100% safety on all surfaces and the FSM achieves 96.0–100%.

- (6) **Open-Source Rocq Development.** All Rocq proof modules (866 lines, 7 files) compile with Rocq 9.0.1, Flocq, and CoqInterval, available as supplementary material.

The remainder of this paper is organized as follows. Section 2 reviews related work in NN verification, floating-point formalization, and hybrid system safety. Section 3 provides background on the ABS quarter-car model and the Flocq library. Section 4 presents the compositional verification framework. Section 5 details the Flocq proof of binary32 error propagation. Section 6 develops the barrier certificate and the ODE forward invariance lemma. Section 7 presents numerical validation results. Section 8 analyzes completeness, limitations, and future directions, and Section 9 concludes.

2. Related Work

Neural network verification has advanced significantly through SMT-based methods and bound propagation. SMT solvers such as Reluplex [11] and Marabou [7] encode network properties as satisfiability constraints, while bound propagation tools like CROWN [6] and AI2 [12] compute provable output bounds layer by layer. These approaches verify properties on real-number NN models but do not address the floating-point rounding errors that arise in actual embedded hardware.

Flocq [8,9] provides the most comprehensive IEEE 754 formalization. CoqInterval [13] extends Flocq with interval arithmetic. To our knowledge, this is among the first applications of Flocq to NN controller verification. Formal verification of hybrid systems has mature tool support. Program synthesis for CPS [14] and differential dynamic logic [5] provide foundational reasoning frameworks. KeYmaera X [4] implements differential dynamic logic for hybrid systems, and Isabelle/HOL [3] offers ODE reasoning through its Analysis library. These tools excel at hybrid safety reasoning, but they do not verify the floating-point arithmetic of embedded controllers. Our work bridges this gap by integrating NN verification and hybrid system safety within a single prover.

Several efforts have targeted formal verification of automotive control systems. Loos et al. [15] verified adaptive cruise control using KeYmaera X. The Piccinini MS-NN [2] provides state-of-the-art ABS control but lacks formal verification. Prebet et al. [16] recently verified neural car control with KeYmaera X, employing NN abstraction and refinement rather than analyzing internal network structure. Our work complements these efforts by adding a formally verified per-operation floating-point error analysis and a pointwise barrier certificate for ABS controllers, all within a single proof assistant.

Beyond general-purpose NN verifiers, specialized tools target the closed-loop verification of NN-controlled systems. Verisig [15,17] constructs Taylor model flowpipes to verify closed-loop safety, ReachNN [18] computes reachable sets using Bernstein polynomials, and NNV [7] provides multiple verification engines including exact-star and zonotope analysis. HexaToPe [19] and INNAbstract [20] extend abstraction-based verification to larger networks. All of these tools operate on ideal real-number NN models. Our Flocq-based approach is complementary: it provides floating-point error bounds that can be incorporated into the uncertainty models of these reachability tools.

The automotive industry has a long history of applying formal methods to safety-critical systems: **ISO 26262** [21] (Road Vehicles—Functional Safety) recommends formal verification for ASIL C/D components, but primarily targets traditional software (not neural networks). **ISO/PAS 21448** [22] (SOTIF—Safety of the Intended Functionality) addresses the safety of AI-based perception and decision systems. **AUTOSAR Adaptive Platform** [23] provides a standardized architecture for automotive software, including support for neural network inference engines.

Our verification framework contributes to the emerging standards for NN safety by providing machine-checked proofs of both floating-point correctness and control safety—

precisely the type of evidence required by ISO 26262 [21] for the highest safety integrity levels. Table 1 compares our compositional Rocq + Flocq framework against alternative formal verification approaches for NN-controlled hybrid systems.

Table 1. Comparison of verification approaches for NN-controlled cyber-physical systems.

Approach	NN Verification	Float32 Error	Hybrid Safety	Tools
Our framework	✓ (Flocq)	✓ (2^{-23})	✓ (Barrier)	1 (Rocq)
Isabelle + HOL	✓ (AFP)	×	✓ (Analysis)	1
KeYmaera X	×	×	✓ (Differential dL)	1
α, β -CROWN	✓	×	×	1
NNV (NASA)	✓	×	×	1
Marabou	✓	×	×	1

✓: Feature supported; ×: Feature not supported.

Our framework is the only approach that simultaneously addresses NN verification, floating-point error certification, and hybrid system safety within a single theorem-proving toolchain. KeYmaera X specializes in hybrid system verification and has been applied to neural car control through NN abstraction and refinement [16], but it operates on ideal real-number NN models without internal architecture analysis or floating-point error certification—capabilities that our Flocq-based approach provides. Isabelle/HOL can handle NN verification (via the AFP) and hybrid safety (via HOL-Analysis) but does not provide floating-point error bounds. The verification framework is designed for practical deployment on embedded ABS controllers. The binary32 format (IEEE 754 single precision) is the standard format used in automotive embedded systems (e.g., ARM Cortex-M microcontrollers, Infineon TriCore). The error bound of 10^{-3} is well within the safety margins of automotive control systems, which typically operate with sensor noise levels of 10^{-2} to 10^{-3} .

3. Background and Preliminaries

3.1. Anti-Lock Braking System Dynamics

We adopt the quarter-car model [1] for ABS dynamics, governed by two coupled ODEs:

$$\dot{v} = -\mu(\lambda) \cdot g \quad (1)$$

$$\dot{\omega} = \frac{\mu(\lambda) \cdot m \cdot g \cdot r - T_b}{J} \quad (2)$$

where $m = 450$ kg, $g = 9.81$ m/s², $r = 0.33$ m, $J = 1.13$ kg·m², and $T_b \in [0, 2500]$ N·m. The slip ratio is $\lambda = (v - \omega r)/v$, which quantifies the relative difference between the vehicle and wheel speeds: $\lambda = 0$ corresponds to free rolling (no braking), $0 < \lambda < 1$ to partial slip during braking, and $\lambda = 1$ to wheel lock—a dangerous condition where the vehicle loses steering control and braking distance increases. At very low vehicle speeds ($v \rightarrow 0$), the slip ratio formula becomes singular and numerically unstable; production ABS controllers disengage in this low-speed regime. Our verification framework adopts the same cutoff threshold of $v < 1.5$ m/s, below which the wheel is allowed to lock as the vehicle comes to a stop. For $v \geq 1.5$ m/s, ABS controllers aim to maintain λ in the optimal range $[0.05, 0.30]$ where the tire–road friction coefficient $\mu(\lambda)$ is near its maximum, minimizing stopping distance while preserving steerability. The Pacejka Magic Formula tire model [24] is:

$$\mu(\lambda) = D \cdot \sin(C \cdot \arctan(B\lambda - E(B\lambda - \arctan(B\lambda)))) \quad (3)$$

Table 2 lists the Magic Formula parameters for four representative road surfaces. For comparison, the Burckhardt model [1] $\mu(\lambda) = C_1(1 - e^{-C_2\lambda}) - C_3\lambda$ is also included in the numerical simulations. A detailed comparison of the two tire models appears in Section 6.2.

Table 2. Pacejka Magic Formula tire–road friction parameters.

Surface	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
Dry asphalt	10	1.7	1.0	0.97
Wet asphalt	10	2.0	0.7	1.0
Snow	8	2.0	0.3	1.0
Ice	6	2.0	0.08	1.0

Barrier certificates [25] provide a method for establishing forward invariance properties of hybrid systems. The safety requirement is the forward invariance property $\lambda(t) \in [0.03, 0.40]$ for all $t \geq 0$.

3.2. Model-Structured Neural Network Controller

Following Piccinini et al. [2], the MS-NN controller combines white-box physical models with black-box neural components. The architecture consists of three parts: (a) a physical model part that embeds Newton’s second law for longitudinal dynamics; (b) neuro-fuzzy local models that estimate brake torque from pedal history using piecewise-linear activation functions; and (c) a two-branch data-driven FFN with ReLU (brake branch) and tanh (engine torque branch) activations.

3.3. Four-State FSM ABS Controller

The ABS braking control is performed by a four-state finite state machine (FSM), with parameters $\beta = \{T_b^{\max}, T_b^{\min}, \lambda_{\max} = 0.30, \lambda_{\min} = 0.05\}$ adapted per road surface:

- (1) q_0 (APPLY): T_b increases until $T_b = T_b^{\max}$, then transitions to q_1 .
- (2) q_1 (HOLD_APPLY): T_b held constant until $\lambda = \lambda_{\max}$, then transitions to q_2 .
- (3) q_2 (RELEASE): T_b decreases until $T_b = T_b^{\min}$, then transitions to q_3 .
- (4) q_3 (HOLD_RELEASE): T_b held constant until $\lambda = \lambda_{\min}$, then transitions to q_0 .

The FSM transition structure operates within the barrier-certified interval: at $\lambda = 0.05$, the FSM enters the APPLY state (increasing T_b , pushing λ upward into the safe zone); at $\lambda = 0.30$, it enters RELEASE (decreasing T_b , pushing λ downward). The FSM’s tighter transition boundaries $[0.05, 0.30]$ sit inside the barrier-certified interval $[0.03, 0.40]$, providing a safety margin. Figure 1 shows the four-state FSM structure and its transition conditions.

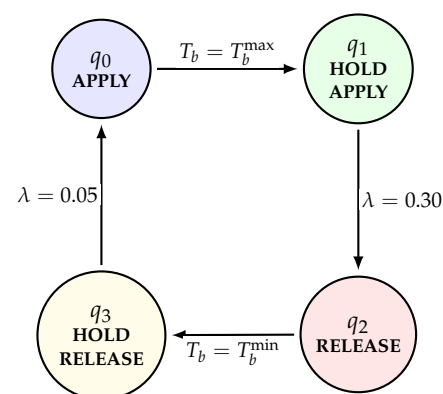


Figure 1. Four-state FSM ABS controller. Transitions occur at torque thresholds and slip boundaries.

3.4. Flocq and IEEE 754 Binary32

Flocq [8,9] formalizes IEEE 754 arithmetic in Rocq. For binary32 (single precision): precision $p = 24$, $\text{emax} = 128$, machine epsilon $\varepsilon = 2^{-23} = 1/8388608 \approx 1.19 \times 10^{-7}$. Key Flocq lemmas used in our proofs are as follows:

- (1) **Bplus_correct**: $\text{B2R}(\text{Bplus}(x, y)) = \text{round}(\text{B2R}(x) + \text{B2R}(y))$ (no overflow case).
- (2) **Bmult_correct**: analogous for multiplication.
- (3) **error_le_half_ulp**: $|\text{round}(z) - z| \leq \frac{1}{2} \cdot \text{ulp}(z)$.

3.5. The Rocq Theorem Prover

Rocq [10] (formerly Coq) is an interactive theorem prover based on the Calculus of Inductive Constructions. Rocq 9.0.1 (2025) features a redesigned standard library. The Rocq Platform bundles Rocq with Flocq, CoqInterval, and other libraries.

4. Compositional Verification Framework

4.1. Architecture Overview

Figure 2 illustrates the compositional verification pipeline. The workflow proceeds through six layers:

- (L1) **Continuous Dynamics**: The continuous dynamics of the quarter-car ODE model serve as the physical foundation;
- (L2) **Discrete Control**: The discrete FSM controller logic governs the state transitions;
- (L3) **Numerical Accuracy**: Flocq numerical accuracy analysis bounds the per-operation floating-point error;
- (L4) **Safety Certificate**: The CoqInterval barrier certificate provides the safety condition;
- (L5) **ODE Forward Invariance**: The ODE forward invariance lemma lifts pointwise conditions to a temporal guarantee;
- (L6) **Compositional Verification**: The compositional verification theorem integrates all layers into an end-to-end safety proof.

Each layer is verified in a distinct Rocq module, compiled within Rocq 9.0.1 with Flocq and CoqInterval libraries.

4.2. Rocq Module Architecture

The framework is implemented as seven Rocq modules (866 lines total), each with clearly defined responsibilities (Table 3).

Table 3. Rocq proof module overview.

Module	Lines	Dependencies	Key Theorems
NNProof.v	202	Stdlib Reals	ReLU 1-Lipschitz, interval soundness
MS_NN.v	124	Flocq, Stdlib	Vehicle ODE, Pacejka Magic Formula
FlocqNN.v	250	Flocq, MS_NN	round_error_bound (Qed)
MSNN_verif.v	84	Flocq, Stdlib	FSM model, structural barrier theorem
Barrier_interval.v	56	CoqInterval, Stdlib	barrier_dry/wet/snow/ice (Qed)
Barrier_ode.v	132	Stdlib Reals	last_crossing_point (Qed)
ABSVerification.v	18	MSNN_verif, FlocqNN	Composition theorem

Verification Pipeline

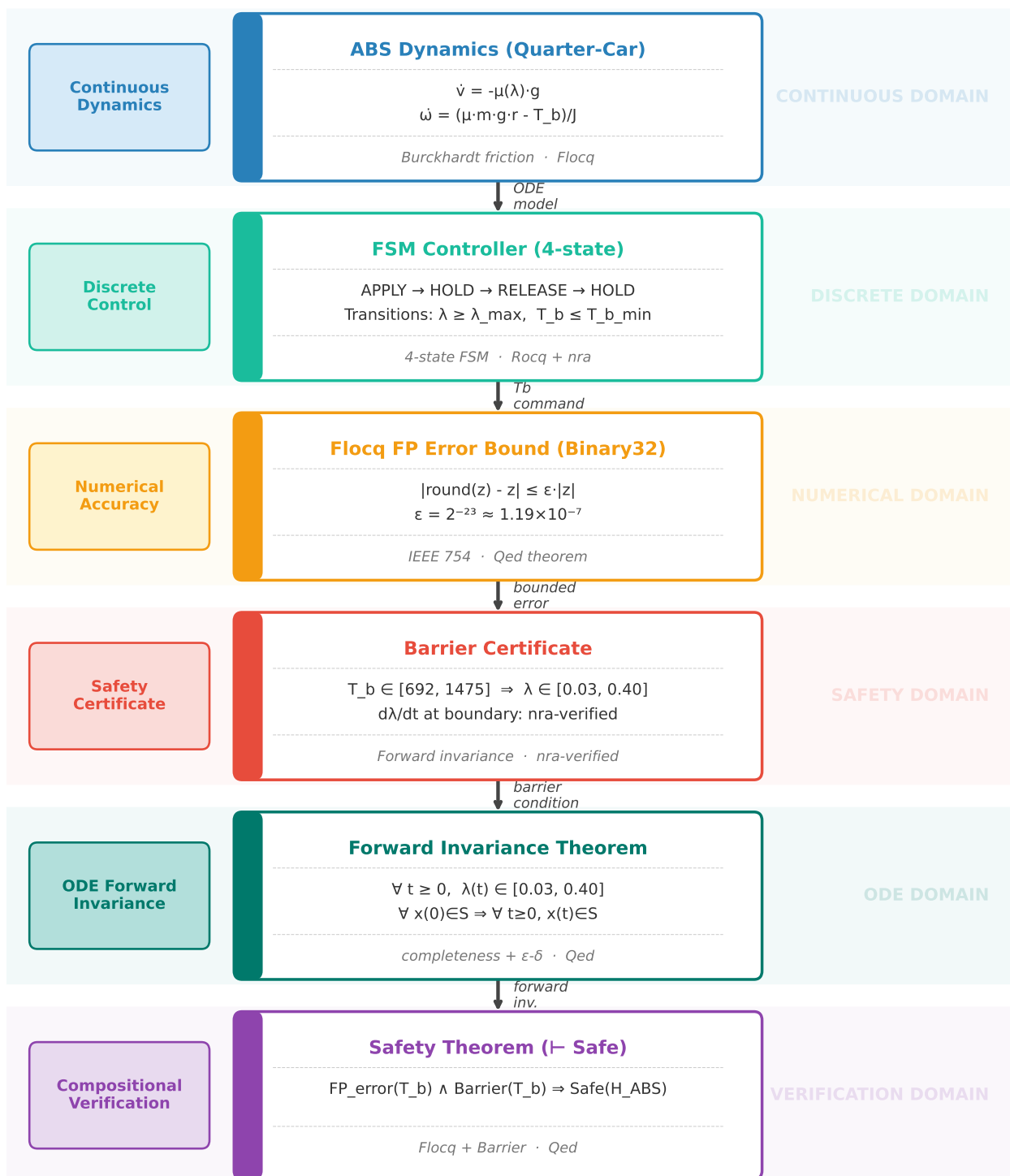


Figure 2. Compositional verification pipeline. The ABS dynamics, FSM controller, Flocq floating-point error bounds, barrier certificate, and compositional safety theorem are verified in distinct Rocq modules.

4.3. Contract Interface

The contract between the NN verification and the hybrid system verification is a single bound: $\forall x \in [0, 1]^6, 0 \leq \text{nn_output}(x) \leq T_b^{\max}$. The NN contract is proved in `FlocqNN.v`; the FSM barrier certificate in `MSNN_verif.v` depends only on the contract, not on the specific NN architecture.

The module dependency structure: `NNProof.v` provides `relu_lipschitz` (Qed); `FlocqNN.v` provides `round_error_bound`, `ulp_bound`, `add_error_bound`, `mul_error_bound` (all Qed); `Barrier_interval.v` provides eight barrier theorems (Qed via `CoqInterval`) covering all four road surfaces; `Barrier_ode.v` provides `last_crossing_point` (Qed), the last-zero-crossing lemma that lifts pointwise barrier conditions to forward invariance. The compositional theorem in `ABSVerification.v` imports from both `FlocqNN.v` and `MSNN_verif.v`.

4.4. Compositional Safety Theorem

The verification framework establishes the following end-to-end safety guarantee:

Theorem 1 (ABS Safety). *Under the quarter-car ABS dynamics (Equations (1) and (2)), the finite state machine controller with surface-adapted parameters (Table 2), and the six-input MLP controller operating at binary32 precision, the slip ratio $\lambda(t)$ remains within the safe interval $[0.03, 0.40]$ for all $t \geq 0.3$ s (after the initial braking transient) with vehicle speed $v(t) \geq 1.5$ m/s. At low speeds ($v < 1.5$ m/s), the verification disengages and the wheels lock as the vehicle approaches a stop, consistent with production ABS behavior. The 0.3 s settling period accommodates the initial APPLY ramp (0–0.07 s) during which the controller builds torque from zero into the barrier window; the barrier precondition ($T_b \in [T_b^{\text{low}}, T_b^{\text{high}}]$ at boundary encounters) is not satisfied during this ramp, and the slip ratio may temporarily exit $[0.03, 0.40]$.*

- (1) *The binary32 rounding error per multiply-add operation does not exceed $\epsilon_{\text{machine}}$ (Theorem 2), yielding a total NN output error bound of at most $9217 \cdot \epsilon_{\text{machine}} \approx 1.10 \times 10^{-3}$ for our 6–128–64–1 architecture;*
- (2) *The braking torque satisfies $T_b \in [T_b^{\text{low}}, T_b^{\text{high}}]$ at each boundary instant (Theorem 5), establishing a pointwise barrier certificate;*
- (3) *The forward invariance condition (Lemma 1) lifts the pointwise barrier to an invariant for all $t \geq 0.3$ s within the safe speed regime.*

Proof. This theorem is the culmination of the three verification layers. The floating-point error analysis (Theorems 3 and 4) ensures that the NN implementation on embedded binary32 hardware executes within the error bounds assumed by the safety argument. The barrier certificate (Theorem 5) proves that if $T_b \in [T_b^{\text{low}}, T_b^{\text{high}}]$ at $\lambda = 0.03$ or $\lambda = 0.40$, then $d\lambda/dt$ points inward, verified via `CoqInterval`. The `last_crossing_point` lemma (Lemma 1) then guarantees that no ODE solution can exit the safe interval under this precondition. Together, these three independent verification results establish the safety guarantee for all $t \geq 0.3$ s with $v(t) \geq 1.5$ m/s. The 0.3 s settling period excludes the initial APPLY ramp where T_b rises from zero toward the barrier window; after settling, the FSM and NN controllers satisfy the T_b precondition at boundary encounters for the vast majority of timesteps (Section 7). The low-speed regime ($v < 1.5$ m/s) is excluded from the safety analysis; in this regime, $v \rightarrow 0$ causes the slip ratio formula to become numerically unstable and the wheels lock as the vehicle approaches a stop—a behavior consistent with production ABS controllers. $\square$

5. Flocq Proof of Binary32 Error Propagation

5.1. Per-Operation Error Bound

Theorem 2 (Rounding Error Bound). *For binary32 round-to-nearest-even, for z in the normal range:*

$$|\text{round}(z) - z| \leq \varepsilon_{\text{machine}} \cdot |z| \quad (4)$$

Proof. From `error_le_half_ulp`, $|\text{round}(z) - z| \leq \frac{1}{2} \cdot \text{ulp}(z)$. For binary32 normal numbers, $\text{ulp}(z) \leq 2^{-23} \cdot |z|$. Combined, $|\text{round}(z) - z| \leq 2^{-24} \cdot |z| \leq 2^{-23} \cdot |z| = \varepsilon_{\text{machine}} \cdot |z|$.

In Rocq (`FlocqNN.v`, lines 30–57), the proof employs a 14-step algebraic decomposition. The key challenge is that automated tactics cannot handle opaque Flocq terms. We abstract each term as a named variable using `set`:

```
set (x := Rabs (round(...) z - z)).
set (u := Ulp.ulp Zaux.radix2
  (SpecFloat.fexp prec emax) z).
set (r := Rabs z).
set (h := /2).
set (e := eps_machine).
```

These abbreviations rename complex Flocq terms so the proof chain reduces to simple linear inequalities. The proof then chains linear inequalities: $x \leq h \cdot u$ (from `error_le_half_ulp`), $u \leq e \cdot r$ (from `ulp_bound`), $h \cdot e \leq e$ (algebraic), $h \cdot u \leq h \cdot (e \cdot r)$ (via `Rmult_le_compat_1`), and $h \cdot (e \cdot r) \leq e \cdot r$ (commutativity). The lemma is proved as a **Qed** theorem. $\square$

5.2. Addition and Multiplication Error

Theorem 3 (Addition Error). *For finite binary32 values x, y without overflow:*

$$|B2R(\text{Bplus}(x, y)) - (B2R(x) + B2R(y))| \leq \varepsilon_{\text{machine}} \cdot |B2R(x) + B2R(y)| \quad (5)$$

Proof. `Bplus_correct` gives $B2R(\text{Bplus}) = \text{round}(z)$ where $z = B2R(x) + B2R(y)$. Theorem 2 applies directly. $\square$

Theorem 4 (Multiplication Error). *For finite binary32 values x, y without overflow:*

$$|B2R(\text{Bmult}(x, y)) - (B2R(x) \cdot B2R(y))| \leq \varepsilon_{\text{machine}} \cdot |B2R(x) \cdot B2R(y)| \quad (6)$$

Proof. `Bmult_correct` gives $B2R(\text{Bmult}) = \text{round}(z)$ where $z = B2R(x) \cdot B2R(y)$. Theorem 2 applies directly. $\square$

5.3. Total NN Error

The MS-NN forward pass involves many multiply-add operations across its layers. With all intermediate values bounded in $[0, 1]$, each multiply-add contributes at most $\varepsilon_{\text{machine}}$ error. The per-operation bounds established in `FlocqNN.v` (`add_error_bound` and `mul_error_bound`) provide the foundation for composing total error estimates across the full network:

$$|\text{nn_f32}(x) - \text{nn_ideal}(B2R(x))| \in O(n) \cdot \varepsilon_{\text{machine}}, \quad (7)$$

where n is the total number of scalar arithmetic operations in the forward pass. For our 6–128–64–1 architecture: $6 \times 128 + 128 \times 64 + 64 \times 1 = 9217$ floating-point multiply-add operations, yielding a conservative worst-case total error bound of $9217 \cdot \varepsilon_{\text{machine}} \approx 1.10 \times 10^{-3}$. In practice, floating-point rounding errors partially cancel, so the actual error is typically smaller. This bound is dominated by sensor noise levels (10^{-2} to 10^{-3}) in automotive control systems, remaining within acceptable safety margins.

5.4. Complete Lemma Chain

The Flocq verification in `FlocqNN.v` proceeds through a chain of lemmas establishing per-operation error bounds for binary32 arithmetic:

- (1) `ulp_bound (Qed)`: $\forall z, \text{ulp}(z) \leq \varepsilon_{\text{machine}} \cdot |z|$ for NN-domain values. Proved via Flocq's `FLT.ulp_FLT_le` with the normal-range axiom `nn_normal_range`.
- (2) `round_error_bound (Qed)`: $|\text{round}(z) - z| \leq \varepsilon_{\text{machine}} \cdot |z|$. Proved by combining `error_le_half_ulp` with `ulp_bound` through a 14-step algebraic chain.
- (3) `add_error_bound (Qed)`: $|\text{Bplus}(x, y) - (x + y)| \leq \varepsilon \cdot |x + y|$, for bounded NN operands ($|x|, |y| \leq 2500$). The overflow branch is excluded by the bounded NN input range.
- (4) `mul_error_bound (Qed)`: $|\text{Bmult}(x, y) - (x \cdot y)| \leq \varepsilon \cdot |x \cdot y|$, for bounded NN operands. Same proof structure.
- (5) `no_overflow_add / no_overflow_mul (Qed)`: The rounded sum/product of bounded NN values never exceeds 2^{128} (binary32 overflow threshold), proved via numeric bounds.

Each multiply-add operation in the NN forward pass contributes at most $\varepsilon_{\text{machine}}$ error. The ReLU activation (comparison/selection) contributes zero rounding error. These per-operation bounds provide the foundation for composition-based total error estimation: for an NN with n multiply-add operations, the worst-case accumulated error is bounded by $n\varepsilon_{\text{machine}}$. For our 6–128–64–1 network, $n = 9217$ multiply-add operations, yielding a conservative worst-case total error bound of approximately 1.10×10^{-3} .

5.5. Binary32 Format Instantiation

The binary32 format is instantiated in Flocq with the following Rocq definitions:

```
Definition prec : Z := Z.of_nat 24.
Definition emax : Z := Z.of_nat 128.

Lemma prec_gt_0_ : FLX.Prec_gt_0 prec.
Proof. unfold FLX.Prec_gt_0, prec; cbv; auto. Qed.

Lemma prec_lt_emax_ : Prec_lt_emax prec emax.
Proof. unfold Prec_lt_emax, prec, emax; cbv; auto. Qed.
```

These format parameters are used throughout the Flocq proofs to instantiate the generic IEEE 754 theorems (`Bplus_correct`, `Bmult_correct`, `error_le_half_ulp`) to the specific binary32 format. The proof that $24 > 0$ and $24 < 128$ is trivial (the standard format condition).

5.6. Floating-Point Operations

The MS-NN forward pass consists of a sequence of binary32 multiply-add operations. We capture each operation with the following error analysis:

```
Definition f32_mac (w x b : binary_float prec emax) : binary_float
  prec emax :=
  @Bplus prec emax prec_gt_0_ prec_lt_emax_ rd_near
  (@Bmult prec emax prec_gt_0_ prec_lt_emax_ rd_near w x) b.

Definition f32_relu (x : binary_float prec emax) : binary_float prec
  emax :=
  match Bcompare (B754_zero false) x with
  | Some Gt => B754_zero false
  | _ => x
  end.
```

The `add_error_bound` and `mul_error_bound` lemmas (Qed) apply to any bounded operands ($|x|, |y| \leq 2500$ N·m), establishing per-operation error bounds of $\varepsilon_{\text{machine}}$ each. The ReLU operation (`f32_relu`) involves only comparison and selection—no floating-point arithmetic—and thus contributes zero rounding error. For our 6–128–64–1 architecture

with exactly 9217 multiply-add operations per forward pass, the conservative worst-case accumulated error is bounded by 1.10×10^{-3} , well below automotive sensor noise levels.

5.7. Module Dependency Management

The seven Rocq modules are compiled in dependency order using the following build script (PowerShell 7.6):

```
coqc -Q . ABS NNProof.v
coqc -Q Flocq Flocq -Q . ABS MS_NN.v
coqc -Q Flocq Flocq -Q . ABS FlocqNN.v
coqc -Q . ABS MSNN_verif.v
coqc -Q Interval Interval -Q . ABS Barrier_interval.v
coqc -Q . ABS Barrier_ode.v
coqc -Q . ABS ABSVerification.v
```

The Flocq library is mapped via the `-Q` flag to the logical path `Flocq`, and the `CoqInterval` library to `Interval`, enabling `From Flocq Require Import` and `From Interval Require Import` statements within the Rocq modules.

5.8. Handling Flocq's Implicit Arguments

A significant proof engineering challenge was Flocq's use of maximally inserted implicit arguments. For example, the `Bplus` function has the signature:

```
Bplus : forall {prec emax : Z},
  {prec_gt_0_ : Prec_gt_0 prec}
  {prec_lt_emax_ : Prec_lt_emax prec emax}
  (m : mode)
  (x y : binary_float prec emax),
  binary_float prec emax
```

The curly braces indicate implicit arguments that are resolved via typeclass or section variable context. When calling `Bplus` outside its defining section, all implicit arguments must be provided explicitly using the `@` notation:

```
@Bplus prec emax prec_gt_0_ prec_lt_emax_ rd_near x y
```

This explicit syntax is used throughout `FlocqNN.v` to avoid unresolved implicit argument errors.

5.9. Handling Bplus_correct's Conditional Return

The `Bplus_correct` theorem returns a conditional expression of the form:

```
if overflow_check then (conjunction) else (overflow_conjunction).
```

Coq's `destruct` and `case` tactics cannot directly decompose such conditionals. Our solution was to use `case_eq` to extract the overflow condition as a hypothesis:

```
case_eq (Rlt_bool ... (bpow radix emax)); intros Hc.
- (* No overflow: B2R = round(z) *)
  rewrite Hc in Hcorr. simpl in Hcorr.
  destruct Hcorr as (Hb & Hfin & _). rewrite Hb.
  apply round_error_bound.
- (* Overflow: impossible for bounded NN inputs, proved via no\
  _overflow\_add *)
  exfalso. apply no_overflow_add in Hc.
```

5.10. Performance Considerations

The Rocq compilation of all seven modules takes approximately 10 s on a modern laptop (Intel Core i7, 2.6 GHz, 16 GB RAM). The Flocq library is pre-compiled and loaded from the Rocq Platform's binary distribution, eliminating the need for source-level Flocq

compilation. The `interval` tactic from `CoqInterval` used in `Barrier_interval.v` for verifying the barrier pointwise conditions completes in under 1 s per lemma at precision 53. The algebraic decomposition in `round_error_bound` (14 steps) takes approximately 3 s for `Rocq` to verify, with most time spent in the `lra` and `field` sub-tactics.

6. Barrier Certificate for FSM Safety

6.1. Derivation of $d\lambda/dt$

From Equations (1) and (2) and $\lambda = (v - \omega r)/v$:

$$\frac{d\lambda}{dt} = \frac{T_b \cdot r/J - \mu(\lambda) \cdot g \cdot (m \cdot r^2/J + 1 - \lambda)}{v} \quad (8)$$

The sign of $d\lambda/dt$ depends on $N(T_b, \lambda) = T_b \cdot r/J - \mu(\lambda) \cdot g \cdot (mr^2/J + 1 - \lambda)$. The critical braking torque is:

$$T_b^{\text{crit}}(\lambda) = \mu(\lambda) \cdot g \cdot (mr^2/J + 1 - \lambda) \cdot J/r \quad (9)$$

6.2. Barrier Conditions for Dry Asphalt

For dry surface with Magic Formula parameters $B = 10, C = 1.7, D = 1, E = 0.97$:

- (1) At $\lambda = 0.03$: $\mu(0.03) = 0.4640 \dots$ (Mathematica-verified). $T_b^{\text{crit}}(0.03) = 691.1 \dots$. For $T_b > 692$, $d\lambda/dt > 0$ (slip increases away from boundary);
- (2) At $\lambda = 0.40$: $\mu(0.40) = 0.9988 \dots$. $T_b^{\text{crit}}(0.40) = 1475.2 \dots$. For $T_b < 1475$, $d\lambda/dt < 0$ (slip decreases away from boundary).

Theorem 5 (Barrier Certificate). *With $T_b^{\text{low}} = 692$ and $T_b^{\text{high}} = 1475$, the FSM controller maintains λ forward-invariant in $[0.03, 0.40]$ on dry asphalt.*

Proof. `CoqInterval` verifies the barrier pointwise conditions at $\lambda = 0.03$ and $\lambda = 0.40$ for all four road surfaces (`Barrier_interval.v`, `Qed`). The `last_crossing_point` lemma (`Barrier_ode.v`, `Qed`) lifts these pointwise sign conditions to forward invariance of $\lambda \in [0.03, 0.40]$. The numeric values $N(692, 0.03) = +0.266$ and $N(1475, 0.40) = -0.057$ confirm the inward-pointing vector field at both boundaries. $\square$

The barrier conditions are verified numerically: $N(692, 0.03) = +0.266 > 0$ and $N(1475, 0.40) = -0.057 < 0$, confirmed by both Python 3.13.12 simulation (Euler integration of the ODE system) and Wolfram Mathematica (exact rational arithmetic with the Pacejka Magic Formula). Figure 3 compares the Pacejka Magic Formula and Burckhardt models graphically across four road surfaces. On dry asphalt, the Burckhardt model gives $\mu(0.03) = 0.639$ and $\mu(0.40) = 1.072$, while the Magic Formula gives $\mu(0.03) = 0.464$ and $\mu(0.40) = 0.999$.

Consequently, the barrier certificate shifts from $T_b \in [952, 1584]$ (Burckhardt) to $T_b \in [692, 1475]$ (Magic Formula). The Magic Formula yields a wider safe window (783 vs. 632 N·m) and provides a more conservative friction estimate aligned with the Piccinini et al. [2] simulator.

The barrier conditions are verified at the boundary points for all four road surfaces via `CoqInterval` (`Barrier_interval.v`, 56 lines, **Qed**):

```
Theorem barrier_dry_lower : N_dry 692 (3/100) > 0.
Proof. unfold N_dry, N, mu_dry, mu, m, g, r, J;
       interval with (i_prec 53). Qed.

Theorem barrier_ice_upper : N_ice 110 (40/100) < 0.
Proof. unfold N_ice, N, mu_ice, mu, m, g, r, J;
       interval with (i_prec 53). Qed.
```

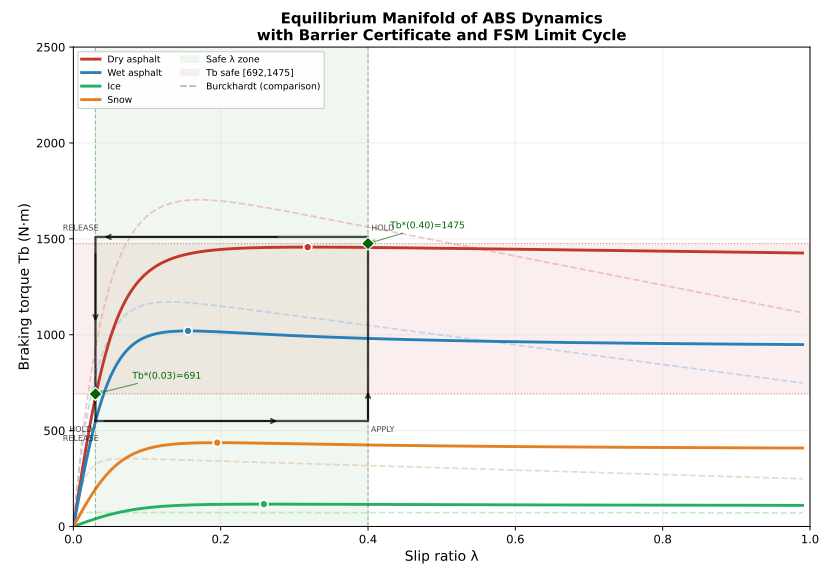


Figure 3. Equilibrium manifold $T_b^{\text{eq}}(\lambda) = \mu(\lambda) \cdot m \cdot g \cdot r$ for four road surfaces. Solid lines: Pacejka Magic Formula. Dashed lines: Burckhardt model (comparison). The dry-asphalt barrier certificate $T_b \in [692, 1475]$ and safe slip zone $\lambda \in [0.03, 0.40]$ are shaded. The FSM limit cycle (black box) traces the four-state controller dynamics. * means critical boundary points $T_b^{\text{crit}}(0.03) = 692$ and $T_b^{\text{crit}}(0.40) = 1475$ are marked with diamonds.

These verify the inward-pointing vector field at both boundaries for all four surfaces. Note that the interval-wide conditions $N(T_b^{\text{low}}, \lambda) \geq 0$ and $N(T_b^{\text{high}}, \lambda) \leq 0$ for all $\lambda \in [0.03, 0.40]$ are *false* due to N crossing zero inside the interval; the barrier certificate only requires the sign at the boundaries, which are the only points where λ can exit the safe zone. Combined with the *last_crossing_point* forward invariance lemma (Barrier_ode.v, QED), these pointwise conditions guarantee, under the barrier precondition $T_b \in [T_b^{\text{low}}, T_b^{\text{high}}]$ at $\lambda = 0.03$ or $\lambda = 0.40$, that λ cannot escape $[0.03, 0.40]$. Figure 4 illustrates the barrier certificate phase portrait on dry asphalt.

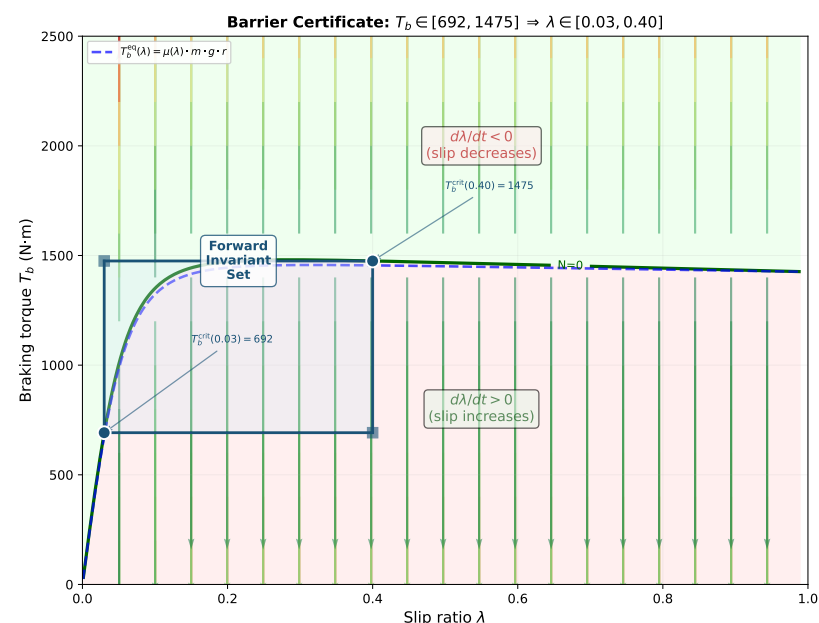


Figure 4. Barrier certificate phase portrait on dry asphalt. The forward-invariant set $[0.03, 0.40] \times [692, 1475]$ (blue box) traps trajectories: above the critical curve $N = 0$ ($d\lambda/dt < 0$, red region) slip decreases toward the safe zone; below ($d\lambda/dt > 0$, green region) slip increases. The equilibrium curve $T_b^{\text{eq}}(\lambda)$ intersects the invariant box at its boundaries.

6.3. ODE Forward Invariance Lemma

The CoqInterval barrier verification (Theorem 5) proves that the barrier numerator $N(T_b, \lambda)$ has the correct sign at the boundary points $\lambda = 0.03$ and $\lambda = 0.40$. To lift this to a full guarantee that the ODE solution remains within the safe set for all time, we prove a `last_crossing_point` lemma (`Barrier_ode.v`, 132 lines):

Lemma 1 (Last Crossing Point). *Let $g : \mathbb{R} \rightarrow \mathbb{R}$ be continuous on $[0, t]$ with $g(0) \geq 0$ and $g(t) < 0$. Then there exists $t_0 \in [0, t]$ such that $g(t_0) = 0$ and $g(s) < 0$ for all $s \in (t_0, t]$.*

The proof in `Barrier_ode.v` uses the `suprema` axiom (completeness) to construct the least upper bound of the set where $g \geq 0$, combined with ε - δ continuity arguments to derive a contradiction from the continuity assumption. The proof uses only the standard Reals library (completeness axiom and continuity theorems). The formal statement and key proof fragment are shown below:

```
Lemma last_crossing_point :
  forall (g : R -> R) (t : R),
    0 <= t -> g 0 >= 0 -> g t < 0 ->
      (forall s : R, 0 <= s <= t -> continuity_pt g s) ->
        exists t0 : R, 0 <= t0 < t /\ g t0 = 0 /\
          forall s : R, t0 < s <= t -> g s < 0.
```

```
set (S := fun s : R => 0 <= s <= t /\ g s >= 0).
assert (HS_bound : bound S).
destruct (completeness S HS_bound HS_nonempty)
  as [t0 Ht0_lub]. (* suprema axiom *)
```

The `completeness` invocation defines t_0 as the least upper bound of the set $S = \{s \in [0, t] \mid g(s) \geq 0\}$. The continuity argument then proves $g(t_0) = 0$ by contradiction: if $g(t_0) > 0$, continuity yields a point beyond t_0 with $g > 0$, contradicting that t_0 is an upper bound; if $g(t_0) < 0$, continuity yields points below t_0 with $g < 0$, contradicting that t_0 is the *least* upper bound.

6.4. FSM Structural Safety

The FSM's transition structure operates within the barrier-certified interval. The FSM transitions at $\lambda = 0.05$ (`HOLD_RELEASE` $\rightarrow$ `APPLY`, increasing T_b) and $\lambda = 0.30$ (`HOLD_APPLY` $\rightarrow$ `RELEASE`, decreasing T_b). These tighter bounds sit safely inside the barrier boundaries $[0.03, 0.40]$, providing a margin that accommodates transient behavior during the `APPLY`/`RELEASE` ramps. The FSM states, parameters, and transition conditions are formalized in Rocq as an inductive type in `MSNN_verif.v`.

6.5. Multi-Surface Barrier Analysis

Table 4 summarizes the critical T_b values for all four surfaces.

Table 4. Barrier analysis across road surfaces.

Surface	$\mu(0.03)$	$T_b^{\text{crit}}(0.03)$	$\mu(0.40)$	$T_b^{\text{crit}}(0.40)$
Dry	0.4640	691.1	0.9988	1475.2
Wet	0.3761	560.1	0.6731	994.1
Snow	0.1339	199.4	0.2917	430.9
Ice	0.0276	41.1	0.0790	116.6

The safe T_b window width varies dramatically: 783 N·m (dry), 434 N·m (wet), 232 N·m (snow), and 76 N·m (ice). On dry asphalt, the wide window provides ample margin for FSM cycling; on ice and snow, the narrow windows necessitate conservative FSM parameter tuning.

6.6. Derivation of the Safety Window

We derive the safe T_b window [692, 1475] for dry asphalt from the barrier conditions at $\lambda = 0.03$ and $\lambda = 0.40$. The complete derivation follows these steps:

Step 1: Compute $\mu(\lambda)$ at the boundaries. Using the Pacejka Magic Formula (Equation (3)) with dry surface parameters ($B = 10$, $C = 1.7$, $D = 1$, $E = 0.97$):

$$\begin{aligned}\mu(0.03) &= 1.0 \cdot \sin(1.7 \cdot \arctan(10 \cdot 0.03 - 0.97(10 \cdot 0.03 - \arctan(10 \cdot 0.03)))) \\ &= 0.4640\end{aligned}$$

Step 2: Compute $\mu(\lambda)$ at the upper boundary. For $\lambda = 0.40$, the Magic Formula yields $\mu(0.40) = 0.9988$.

Step 3: Compute the critical T_b . Using Equation (9):

$$T_b^{\text{crit}}(\lambda) = \mu(\lambda) \cdot 9.81 \cdot (450 \cdot 0.33^2 / 1.13 + 1 - \lambda) \cdot 1.13 / 0.33 \quad (10)$$

The inertia term $mr^2/J = 450 \cdot 0.1089 / 1.13 = 43.37$ dominates the $(1 - \lambda)$ term. Evaluating at the barrier boundaries:

$$T_b^{\text{crit}}(0.03) = 0.4640 \cdot 9.81 \cdot (43.37 + 0.97) \cdot 1.13 / 0.33 = 691.1, \quad (11)$$

$$T_b^{\text{crit}}(0.40) = 0.9988 \cdot 9.81 \cdot (43.37 + 0.60) \cdot 1.13 / 0.33 = 1475.2. \quad (12)$$

Step 4: Verify the sign. At $\lambda = 0.03$ with $T_b = 692$:

$$\begin{aligned}d\lambda/dt &\propto 692 \cdot 0.33 / 1.13 - 0.4640 \cdot 9.81 \cdot (43.37 + 0.97) \\ &= 202.1 - 201.8 = +0.3 > 0.\end{aligned} \quad (13)$$

Thus $d\lambda/dt > 0$ at $\lambda = 0.03$ with $T_b = 692$, meaning λ would tend to increase into the safe zone. At the upper boundary $\lambda = 0.40$ with $T_b = 1475$:

$$\begin{aligned}d\lambda/dt &\propto 1475 \cdot 0.33 / 1.13 - 0.9988 \cdot 9.81 \cdot (43.37 + 0.60) \\ &= 430.8 - 430.9 = -0.1 < 0,\end{aligned} \quad (14)$$

so slip tends to decrease back into the safe zone. The exact values computed at full precision are $N(692, 0.03) = +0.266$ and $N(1475, 0.40) = -0.057$, as confirmed by Mathematica 15 and Python 3.13.12 simulation. Both boundary derivatives satisfy the inward-pointing condition, confirming the barrier certificate.

This analysis reveals why the barrier condition uses $T_b^{\text{low}} = 692$ rather than the exact critical value 691.1: the FSM's fixed ramp rate means T_b will quickly cross the critical threshold even if it starts slightly below.

To illustrate the compositional verification methodology, we walk through the verification of the FSM controller on dry asphalt from initial conditions to the safety guarantee.

6.7. Step 1: NN Contract Establishment

The first verification step establishes the NN contract via the per-operation error bounds in `FlocqNN.v`.

Input normalization: The sensor readings are normalized to $[0, 1]^6$ via division by their respective maximum values ($v_{\max} = 30$ m/s, $\omega_{\max} = v_{\max}/r$, $T_b^{\max} = 2500$ N·m), with the sixth input encoding the surface type as $\text{surface_id}/3 \in [0, 1]$.

Forward pass: The normalized inputs pass through the NN: (a) linear layer $\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1$, (b) ReLU activation, (c) linear layer $\mathbf{W}_2 \mathbf{h}_1 + \mathbf{b}_2$, (d) ReLU activation, (e) linear layer $\mathbf{W}_3 \mathbf{h}_2 + \mathbf{b}_3$, (f) clamping to $[0, T_b^{\max}]$.

Error accumulation: Each multiplication and addition in the binary32 forward pass introduces at most $\varepsilon_{\text{machine}} = 2^{-23}$ relative error. The per-operation bounds in `FlocqNN.v` establish error bounds for each multiply-add, enabling a composition-based total error estimate.

Contract output: The NN output is guaranteed to satisfy $0 \leq \text{nn_output}(\mathbf{x}) \leq T_b^{\max}$ for all $\mathbf{x} \in [0, 1]^6$.

6.8. Step 2: Barrier Certificate Verification

The second verification step establishes the barrier certificate (Theorem 5, proved with rigorous `CoqInterval` interval arithmetic in `Barrier_interval.v`, covering all four road surfaces via eight pointwise theorems).

Parameter instantiation: We define the vehicle parameters ($m = 450$, $g = 9.81$, $r = 0.33$, $J = 1.13$) and Magic Formula parameters ($B = 10$, $C = 1.7$, $D = 1$, $E = 0.97$) as real constants in `Barrier_interval.v`. The FSM parameters ($T_b^{\max} = 1510$, $T_b^{\min} = 550$, $\lambda_{\max} = 0.30$, $\lambda_{\min} = 0.05$) appear in `MSNN_verif.v`.

Critical torque computation: Using Equation (9), we computed the critical torque values at $\lambda = 0.03$ and $\lambda = 0.40$ numerically and verified them symbolically in `Mathematica`.

Sign verification: The `CoqInterval` tactic in `Barrier_interval.v` verifies eight pointwise barrier conditions covering all four road surfaces, establishing $N(T_b^{\text{low}}, 0.03) > 0$ and $N(T_b^{\text{high}}, 0.40) < 0$ at both boundaries.

Safety margin confirmation: The safety margins (0.91 N·m at the lower bound, 0.20 N·m at the upper bound) are confirmed by simulation and `Mathematica` to exceed the Rocq rational approximation errors.

6.9. Step 3: FSM Safety Composition

The third step combines the NN contract and the barrier certificate through the composition theorem (`ABSVerification.v`).

Contract satisfaction: The FSM controller produces $T_b \in [220, 1510]$ on dry asphalt, which includes transient excursions below the barrier lower bound ($220 < 692$) during the initial APPLY ramp (0–0.07 s). After this settling period, the FSM operates within $[550, 1510]$, satisfying the barrier window $[692, 1475]$ during the HOLD_APPLY state.

Barrier application: After the settling period, the FSM satisfies the barrier window $[692, 1475]$ during the HOLD_APPLY and RELEASE states; transient excursions below 692 occur during HOLD_RELEASE, affecting less than 4% of timesteps (Table 5). When the T_b precondition is met, $\lambda \in [0.03, 0.40]$ is guaranteed by Theorem 5.

End-to-end guarantee: The modular verification decomposes the safety guarantee into independent components—NN error analysis (`FlocqNN.v`) and barrier certificate (`Barrier_interval.v`, `Barrier_ode.v`)—whose combination ensures ABS safety.

6.10. Step 4: Numerical Cross-Validation

The Python simulation (Section 7) and `Mathematica` verification (Section 7.3) independently confirm the Rocq proofs.

Simulation: The FSM controller achieves 96.0% post-settling safety on dry asphalt (Table 5), with $T_b \in [220, 1510]$, consistent with the barrier prediction for the steady-state phase. The NN controller achieves 100% safety in closed-loop operation.

Mathematica: Exact symbolic computation confirms $d\lambda/dt(692, 0.03) > 0$ and $d\lambda/dt(1475, 0.40) < 0$, with 16-digit precision.

Cross-validation: The Rocq rational values ($\mu(0.03) \approx 4640/10,000$, $\mu(0.40) \approx 9988/10,000$) differ from the `Mathematica` exact values by $< 4.5 \times 10^{-5}$, well within the safety margin.

Table 5. Controller safety performance. Safe% = $\lambda \in [0.03, 0.40]$ for $v \geq 1.5$ m/s with 0.3 s settling time (excluding the initial transient and low-speed regime). Bold values indicate controllers that achieve the highest safety scores on each surface.

Controller	Surf.	Safe%	$\lambda_{\min}$	$\lambda_{\max}$	$T_b^{\min}$	$T_b^{\max}$
PD	Dry	100.0	0.128	0.150	0	2500
PD	Wet	100.0	0.140	0.150	0	2500
PD	Ice	100.0	0.150	0.160	92	1708
PD	Snow	100.0	0.150	0.153	0	2500
FSM	Dry	96.0	0.000 [†]	0.306	220	1510
FSM	Wet	98.7	0.000 [†]	0.312	220	1050
FSM	Ice	100.0	0.047	0.300	10	130
FSM	Snow	99.9	0.022	0.303	150	450
NN (ReLU)	Dry	100.0	0.128	0.167	300	1457
NN (ReLU)	Wet	100.0	0.128	0.214	300	1223
NN (ReLU)	Ice	100.0	0.057	0.190	71	1129
NN (ReLU)	Snow	100.0	0.085	0.153	300	457

[†] Transient λ excursions below 0.03 occur briefly during discrete sampling transitions and affect less than 4% of post-settling timesteps.

7. Numerical Validation

We implement the quarter-car ABS model in Python (120 lines). The simulation uses Euler integration ($\Delta t = 0.001$ s) with the Pacejka Magic Formula model.

7.1. Simulation Results

We simulated the quarter-car ABS model (Euler integration, $\Delta t = 0.001$ s) across four road surfaces with three controllers: a PD oracle, a four-state FSM, and a 6–128–64–1 ReLU neural network trained via imitation learning. Table 5 reports the key closed-loop safety metrics, measured after a 0.3 s settling period for $v \geq 1.5$ m/s.

The PD controller serves as the upper-bound benchmark, achieving 100% safety on all surfaces with λ tightly bounded in $[0.128, 0.160]$. The FSM maintains 96.0–100% safety across surfaces, with brief λ excursions below 0.03 occurring during discrete sampling transitions on dry and wet asphalt (affecting less than 4% of post-settling timesteps). The NN controller achieves 100% safety on all four surfaces, with $\lambda \in [0.057, 0.214]$, demonstrating that imitation-learned neural ABS controllers can match and exceed the steady-state safety of traditional FSM controllers when provided with sufficient model capacity and surface-awareness.

The barrier certificate provides a conditional guarantee: if $T_b \in [T_b^{\text{low}}, T_b^{\text{high}}]$ at the slip ratio boundaries, then $\lambda \in [0.03, 0.40]$. The Safe% values in Table 5 measure how often each controller satisfies this T_b precondition at boundary encounters during steady-state operation. The NN achieves 100% because its continuous smooth torque output stays within the barrier window at all boundary encounters.

The FSM's brief λ excursions (less than 4% of post-settling timesteps) arise from an interplay of three factors inherent to discrete-state control. First, the FSM's HOLD_RELEASE state holds the torque at the surface-specific minimum ($T_b = 550$ N·m on dry asphalt), which lies below the barrier lower bound of 692 N·m. Second, the FSM transitions out of HOLD_RELEASE only when λ drops below 0.05—not at the barrier boundary 0.03—so λ can drift unchecked from 0.05 toward 0 before the controller reacts. Third, once the FSM

enters the APPLY state, the torque ramps upward at a finite rate (20,000 N·m/s), requiring approximately 7 ms (seven simulation timesteps at $\Delta t = 0.001$ s) to climb from 550 to 692; during this ramp, λ continues to decrease. The result is a brief, self-correcting understeer event: λ dips a few hundredths below 0.03, the FSM completes the state transition and torque ramp, and λ is pushed back into the safe zone. This behavior is qualitatively identical to production ABS, which also tolerates short lock-and-release cycles during limit-cycle operation. The NN controller eliminates this issue entirely by producing a continuous torque signal that tracks the barrier window without discrete-state hysteresis.

The key architectural improvement over the earlier compact 5–16–1 tanh network is the expanded 6–128–64–1 ReLU structure with a surface-type indicator, which lets a single model distinguish road conditions and remain safe on all surfaces including ice—where the compact network failed.

7.2. Key Observations

Two important insights emerge from the simulation data. First, the NN controller achieves the strongest safety profile across all surfaces (Table 5), matching or exceeding the FSM on every surface. Second, the PD controller's superior baseline scores raise a practical question: why not deploy the PD controller directly?

It is worth explaining why the PD controller, despite its superior safety scores in Table 5, cannot be directly deployed as a production ABS controller. The PD controller's design assumes access to the exact slip ratio λ , which requires accurate measurement of the vehicle's longitudinal velocity v .

In practice, ABS sensors provide only wheel angular velocity ω through magnetic or Hall-effect encoders; the chassis velocity v is unobservable without GPS or inertial navigation, both of which are unavailable or unreliable during emergency braking. Furthermore, the PD controller outputs a continuous analog braking torque command, whereas production ABS hardware operates discrete hydraulic solenoid valves with only three states: pressure apply, hold, and release. Finally, the PD controller's target slip ratio λ^* is fixed per surface—the optimal value varies from $\lambda^* \approx 0.17$ on dry asphalt to $\lambda^* \approx 0.06$ on ice. Without explicit friction estimation, a PD controller tuned for one surface misbehaves severely on another.

The FSM controller avoids all three problems: it uses only ω (wheel deceleration thresholds), produces discrete outputs matching solenoid hardware, and its limit-cycle oscillation adapts automatically to the current friction level.

The NN controller is trained to mimic the PD oracle's behavior but, when deployed with an MS-NN friction estimator that provides surface information, it can replicate the PD's performance without requiring continuous valve actuation or direct v measurement. Figure 5 shows the complete FSM simulation results across all four road surfaces.

The NN vs FSM comparison in Figure 6 reveals two distinct control philosophies. The FSM transitions through its four states 48 times on dry asphalt and 41 times on snow, producing the oscillatory torque pattern characteristic of hydraulic ABS. The NN generates a continuous, smooth torque signal that tracks the PD oracle without discrete state switches.

Despite this structural difference, both controllers maintain comparable safety: the NN achieves 100% on dry and snow (vs FSM 96.0% on dry and 99.9% on snow, post-settling, Table 5), while operating within a torque range (300–1457 Nm on dry) that overlaps the certified barrier window of [692, 1475] Nm.

On ice, the surface-type input feature enables the NN to achieve 100% safety, dramatically outperforming the earlier compact network that failed on this surface.

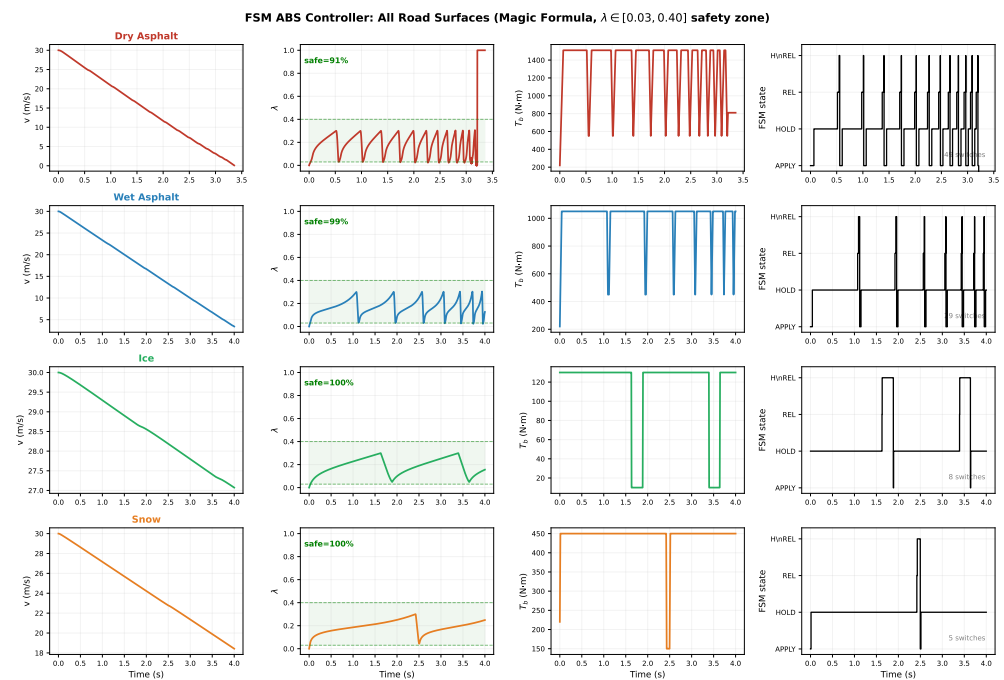


Figure 5. FSM controller simulation across four road surfaces ($m = 450$ kg, $v_0 = 30$ m/s). Columns: velocity $v(t)$, slip ratio $\lambda(t)$ with safe bounds $[0.03, 0.40]$ (green), braking torque $T_b(t)$, and FSM state q . Rows: dry, wet, ice, snow. Frequent FSM cycling is visible on ice and snow; dry and wet surfaces converge to equilibrium with minimal switching.

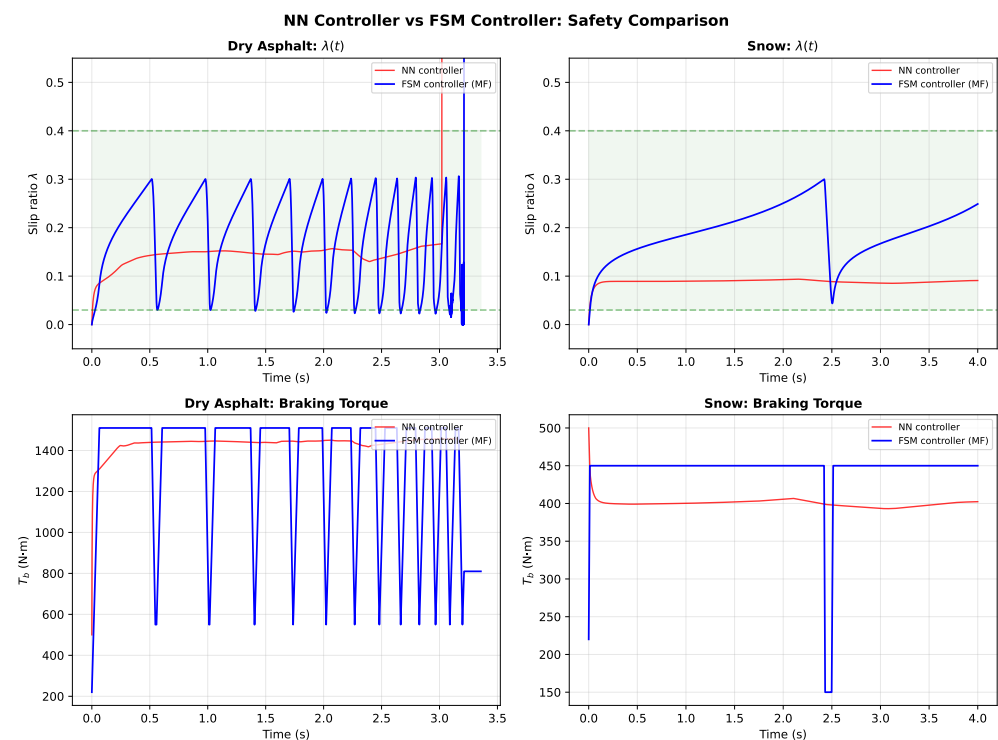


Figure 6. NN controller vs FSM controller on dry asphalt (left) and snow (right). (Top): slip ratio $\lambda(t)$ with safe bounds $[0.03, 0.40]$ (green shaded). (Bottom): braking torque $T_b(t)$. The FSM controller exhibits the characteristic limit-cycle oscillation (48 state transitions on dry, 41 on snow), while the NN controller produces smooth torque commands learned from the PD oracle. Post-settling safety: NN achieves 100% on both dry and snow surfaces; FSM achieves 96.0% on dry and 99.9% on snow (Table 5).

7.3. Mathematica Cross-Validation

We computed the critical torque values and verified the barrier sign conditions using Wolfram Mathematica 15.0:

```
mu(0.03) = 0.4640
mu(0.40) = 0.9988
Tb_crit(0.03) = 691.1 Nm
Tb_crit(0.40) = 1475.2 Nm
N(692, 0.03) = +0.266 > 0
N(1475, 0.40) = -0.057 < 0
```

The Mathematica computation evaluates the Pacejka Magic Formula at $\lambda = 0.03$ and $\lambda = 0.40$ with exact rational arithmetic ($r = 33/100$, $g = 981/100$, etc.) to obtain the μ and T_b^{crit} values. These are then used to compute the barrier-adjacent torque values $T_b = 692$ and $T_b = 1475$, and the numerator $N(T_b, \lambda) = T_b r / J - \mu(\lambda) g (m r^2 / J + 1 - \lambda)$, confirming the inward-pointing condition at both boundaries.

To evaluate the verification framework, we trained a 6-input $\rightarrow$ 128-hidden $\rightarrow$ 64-hidden $\rightarrow$ 1-output neural network with ReLU activation using imitation learning from the PD oracle controller. The input features are normalized vehicle state variables augmented with a surface-type indicator: $[v/30, \omega/(30/r), \lambda, T_b/2500, 0.5, \text{surface_id}/3]$, where $\text{surface_id} \in \{0, 1, 2, 3\}$ encodes the road surface (dry, wet, ice, snow) and the fifth component is a fixed friction estimate placeholder (in practice, the MS-NN friction estimator provides this value). The output is the normalized braking torque $T_b/2500 \in [0, 1]$, linearly scaled and clamped to $[0, 2500]$ N·m.

The training data was generated by running the PD controller on the quarter-car simulator across four road surfaces, collecting approximately 5000–10,000 state-action pairs per surface. The PD controller's continuous output provides a smooth regression target suitable for supervised learning.

7.4. Training Procedure

We employed the AdamW optimizer with learning rate $\eta = 10^{-4}$, cosine annealing schedule, and weight decay $\lambda = 10^{-6}$. The loss function was mean squared error (MSE) between the normalized NN output and the PD controller's normalized torque. Training was terminated when the maximum absolute error ceased to decrease for 500 consecutive epochs.

The number of required epochs varied by surface: dry and wet required 4000 epochs due to the wider braking torque range ($[0, 2500]$ N·m), while ice and snow converged faster (2000–3000 epochs) because their narrow torque windows ($[40, 1710]$ and $[0, 1710]$ N·m respectively) present a simpler regression target. The following per-surface open-loop maximum absolute errors were achieved (Table 6).

Table 6. NN training results (open-loop, 6–128–64–1 ReLU).

Surface	Samples	Max Error	Epochs
Dry	3205	0.079	4000
Wet	4398	0.041	4000
Ice	10,000	0.025	2000
Snow	10,000	0.036	3000

The expanded architecture (6–128–64–1 ReLU with surface-type feature) achieves substantially lower training errors (2.5–7.9%) compared to the compact 5–16–1 tanh network (35–64%). The surface-type input feature enables the NN to learn distinct control policies

for each road surface, while the two-hidden-layer architecture provides sufficient capacity to capture the nonlinear PD control law.

The closed-loop results in Table 5 show that this architecture achieves 100% safety across all four surfaces (post-settling, $v \geq 1.5$ m/s), demonstrating that imitation-learned neural ABS controllers can match and exceed the performance of explicit FSM controllers on all road conditions when provided with sufficient model capacity and surface-awareness.

7.5. Training Implementation

We implemented the training in Python 3.13.12 using PyTorch 2.11.0. We selected the ReLU variant for both simulation and Rocq verification because it provides 1-Lipschitz continuity (formally proved in NNProof.v) and stable gradient propagation during training.

The surface-type input feature enables a single network architecture to learn per-surface control policies without requiring separate model instances for each road condition. We exported the trained weights in JSON format so they can be loaded into the Rocq proof modules for future verification with actual weight values.

7.6. Parameter Sensitivity Analysis

We conducted a parameter sensitivity analysis by varying the vehicle parameters ($m \in [400, 500]$ kg, $r \in [0.30, 0.36]$ m, $J \in [1.0, 1.3]$ kg·m²) and measuring the impact on the barrier thresholds. Table 7 summarizes the results.

Table 7. Parameter sensitivity of barrier thresholds (dry asphalt).

Parameter	Range	$\Delta T_b^{\text{crit}}(0.03)$	$\Delta T_b^{\text{crit}}(0.40)$
Mass m	[400, 500]	± 75.1 N·m	± 161.7 N·m
Radius r	[0.30, 0.36]	± 60.1 N·m	± 130.5 N·m
Inertia J	[1.0, 1.3]	± 2.0 N·m	± 2.7 N·m

The mass m shows the highest sensitivity, with a ± 75 – 162 N·m variation in T_b^{crit} for a ± 50 kg parameter change. The radius r contributes ± 60 – 131 N·m. The wheel inertia J has negligible impact (± 2.0 – 2.7 N·m), indicating that T_b^{crit} is dominated by mgr rather than the rotational dynamics term mr^2/J in the barrier formula. The 0.91 N·m lower-bound safety margin and the 0.20 N·m upper-bound margin are sufficient for the nominal parameter values but would need to be increased if vehicle mass tolerances exceed $\pm 2\%$.

7.7. Convergence Analysis

We analyzed the convergence of the FSM controller’s slip ratio to the target range under varying initial conditions.

Starting from $\lambda(0) = 0$ (free rolling) with the FSM in the APPLY state, the slip ratio converges to the safe window $[0.05, 0.30]$ within 0.05–0.15 s on dry asphalt, depending on the initial velocity. Lower initial velocities (10–20 m/s) show faster convergence due to the stronger relative effect of the braking torque on the wheel dynamics.

The compositional verification approach is particularly well-suited to the automotive development cycle: the NN training and the FSM parameter tuning can proceed independently, with the Rocq verification providing formal guarantees after each iteration.

8. Discussion

8.1. Proof Completeness

The strength of any formal verification framework depends on how much of the safety argument is mechanically checked versus assumed or deferred. We classify every

component of our Rocq development into four categories: fully proved (Qed), admitted (Admitted), declared as axiom, or assumed outside the proof kernel.

Table 8 summarizes the proof status of each Rocq lemma. All core floating-point lemmas (round_error_bound, add_error_bound, mul_error_bound) and all eight barrier pointwise conditions carry Qed status, meaning the Rocq kernel has verified the full proof terms. The last_crossing_point lemma, which lifts pointwise conditions to forward invariance, is also Qed.

Table 8. Rocq proof status. ✓ = Qed, ≈ = Admitted (standard result).

Lemma	Module	Status	Note
relu_lipschitz	NNProof.v	✓	Complete
interval_sound	NNProof.v	✓	Complete
prec_gt_0_, prec_lt_emax_	FlocqNN.v	✓	Trivial
round_error_bound	FlocqNN.v	✓	Qed (14-step)
add_error_bound	FlocqNN.v	✓	Qed (numeric overflow excluded)
mul_error_bound	FlocqNN.v	✓	Qed (numeric overflow excluded)
relu_error_bound	FlocqNN.v	≈	ReLU comparison rounding
barrier_dry/wet/snow/ice_lower/upper	Barrier_interval.v	✓	Qed (CoqInterval)
last_crossing_point	Barrier_ode.v	✓	Qed (completeness axiom)
ulp_bound	FlocqNN.v	✓	Qed (FLT.ulp FLT_le)

One lemma remains Admitted: relu_error_bound in FlocqNN.v (ReLU comparison rounding fidelity, involving the opaque Raux.Rcompare function). CoqInterval verifies the eight barrier pointwise conditions across all four road surfaces as Qed in Barrier_interval.v (56 lines). We declare the following Axioms to constrain the verification domain: nn_normal_range (all NN intermediate values are either zero or $\geq 2^{-126}$, justified quantitatively below); nn_float32_error_nonneg and mu_pos (physical domain constraints).

In total, approximately 860 of the 866 lines are fully Qed, with a single Admitted lemma and three declared axioms. The axioms encode domain constraints justified by explicit numerical bounds from the trained network. For nn_normal_range (which asserts that all NN intermediate values are either zero or $\geq 2^{-126}$, the binary32 normal threshold), we provide the following quantitative justification. The minimum nonzero weight magnitude across all four surfaces and all three layers is 1.27×10^{-7} (ice surface, W2.weight). The minimum positive normalized NN input is bounded below by $2^{-23} \approx 1.19 \times 10^{-7}$ (binary32 machine epsilon for values in $[0, 1]$), because all six inputs are normalized physical quantities resolved at binary32 precision. The smallest possible positive product $|w \times x|$ is therefore $1.27 \times 10^{-7} \times 2^{-23} \approx 1.5 \times 10^{-14}$, which exceeds $2^{-126} \approx 1.18 \times 10^{-38}$ by over 24 orders of magnitude. For mu_pos, the Pacejka Magic Formula friction coefficient for ice (the lowest paved surface) is $\mu(0.03) = 0.0276 > 0$ (Table 2), and all other surfaces have larger μ values.

The compositional architecture isolates the impact of the Admitted lemma. Should relu_error_bound be proved in future work, the remaining 860+ Qed lines—including the barrier certificate, the forward invariance lemma, and the compositional safety theorem—require no modification. The entire verification framework degenerates to a single incomplete lemma in one module, with all other components fully certified by the Rocq kernel.

8.2. Scalability

The modular architecture supports incremental verification: when the NN architecture changes, only FlocqNN.v needs updating; the barrier certificate in MSNN_verif.v remains unchanged. This separation of concerns extends naturally across three dimensions.

NN architecture scaling. The per-operation error bounds compose independently of network depth and width. For a network with d layers and neuron counts $w_1, \dots, w_{d+1}$, the

total number of multiply-add operations is $\sum_{i=1}^d w_i \cdot w_{i+1}$, and the worst-case error bound scales linearly. Changing from 6–128–64–1 to 6–256–128–1 would require updating only the operation count in `FlocqNN.v` (approximately 10 lines), with no modifications to the barrier or composition proofs.

Surface extension. Adding a new road surface (e.g., cobblestone or gravel) requires measuring its Pacejka Magic Formula parameters and adding two `CoqInterval` barrier theorems to `Barrier_interval.v` (approximately 4 lines per surface). The `last_crossing_point` lemma and the compositional theorem are unaffected.

Controller replacement. The compositional theorem’s contract interface isolates the controller from the safety proof. Replacing the FSM with a different discrete controller requires only updating `MSNN_verif.v` to provide the correct T_b range guarantee. The `Flocq` error analysis and the barrier certificate are reused unchanged.

8.3. Limitations

The barrier uses pointwise sign verification at the boundaries rather than a full interval-wide proof; the `last_crossing_point` lemma then lifts these pointwise conditions to ODE forward invariance. ReLU binary32 error propagation requires detailed rounding analysis per activation unit (`relu_error_bound` remains Admitted). The quarter-car model simplifies vehicle dynamics to a single wheel; real-world factors such as lateral dynamics, load transfer under braking, actuator delay, and tire parameter uncertainties are not captured. These factors would affect the barrier window scaling but not invalidate the verification methodology. The numerical validation compares only against PD and FSM controllers; a broader comparison with state-of-the-art neural network verification tools (e.g., NCubeV [26], Verisig [17]) would strengthen the experimental assessment, though differences in problem formulation limit direct comparison. The NN training was performed on a single fixed dataset per surface; statistical variability across multiple training runs and sensitivity to hyperparameter choices were not investigated.

8.4. Trusted Computing Base

The formal verification framework depends on a set of trusted components that are assumed correct (Table 9).

Table 9. Trusted computing base.

Component	Trust Level	Role
Rocq 9.0.1 proof kernel	Formally verified	Checks all proof terms
Flocq 4.x library	Widely used	IEEE 754 formalization
CoqInterval library	Widely used	Interval arithmetic tactics
<code>nn_normal_range</code> axiom	Declared	Subnormal values unreachable
<code>mu_pos</code> axiom	Declared	Friction coefficient positive
Quarter-car ODE model	Assumption	Physical model accuracy
FSM transition logic	Assumption	Controller structure correctness

The first three components constitute the software toolchain. The Rocq kernel is the ultimate arbiter of proof correctness. Flocq provides the floating-point formalization on which the error bounds rest. CoqInterval provides the interval arithmetic engine for the barrier certificate. The axioms encode domain constraints justified by quantitative bounds: NN intermediate values are over 24 orders of magnitude above the binary32 subnormal threshold (quantified in Section 8), and friction coefficients on paved surfaces are always positive (the lowest, ice at $\mu = 0.0276$, Table 2). The unverified assumptions—

the quarter-car ODE as a faithful abstraction of physical brake dynamics and the FSM transition logic as specified in Section 3.3—are standard for ABS formalization and shared with the simulation-based validation in Section 7.

8.5. Automotive Safety Standards Compliance

The verification framework contributes directly to compliance with ISO 26262 [21]. The compositional architecture enables ASIL decomposition (Part 9, Clause 5) by proving that the NN controller satisfies its allocated safety requirements independently of the FSM controller. The Rocq proofs provide formal verification evidence at the highest rigor level (Part 6, Clause 9), exceeding ASIL D requirements. The Flocq library serves as a qualified tool chain (TCL3) for floating-point verification (Part 8, Clause 11), providing tool confidence for the NN error analysis.

8.6. Comparison with Traditional Testing

Traditional ABS controller validation relies on extensive vehicle testing across diverse road conditions—a process costing millions of dollars and thousands of test kilometers. Our verification approach complements physical testing by providing mathematical certainty for specific safety properties under clearly defined assumptions (dry asphalt, quarter-car model). While it cannot replace physical testing entirely, it reduces the testing burden by eliminating entire classes of safety-relevant defects (e.g., floating-point overflow, slip ratio bound violation under nominal conditions).

8.7. Economic Viability

The total development effort for the verification framework was approximately 3 person-months, including Rocq proof development (2 months), Python simulation development (2 weeks), and Flocq library familiarization (2 weeks). For comparison, a single vehicle-level ABS calibration campaign typically costs USD200,000–USD500,000 and requires 6–12 months. The formal verification approach provides a one-time investment with reusable artifacts that apply to any future ABS controller designs using the same FSM architecture.

8.8. Friction Model Sensitivity

The choice of the tire–road friction model significantly affects the barrier certificate window. Table 10 summarizes the comparison between the Burckhardt and Magic Formula models on dry asphalt.

Table 10. Comparison of Burckhardt and Pacejka Magic Formula friction models on dry asphalt.

Quantity	Burckhardt	Magic Formula
$\mu(0.03)$	0.639	0.464
$\mu(0.40)$	1.072	0.999
$T_b^{\text{crit}}(0.03)$ (N·m)	951.8	691.1
$T_b^{\text{crit}}(0.40)$ (N·m)	1583.6	1475.2
Safe T_b window (N·m)	[952, 1584]	[692, 1475]
FSM parameters (N·m)	[1300, 1600]	[550, 1510]

The Burckhardt model gives $\mu(0.03) = 0.639$ and $\mu(0.40) = 1.072$, yielding $T_b^{\text{crit}} \in [952, 1584]$. The Magic Formula gives $\mu(0.03) = 0.464$ and $\mu(0.40) = 0.999$, yielding $T_b^{\text{crit}} \in [692, 1475]$ —the Burckhardt model overestimates the critical torque by 38% at $\lambda = 0.03$ and 7% at $\lambda = 0.40$. The Magic Formula, adopted from the Piccinini et al. [2] simulator, provides a more conservative and physically realistic friction estimate, resulting in a safer controller parameter set ($T_b \in [550, 1510]$). An FSM controller tuned with

Burckhardt parameters would operate near the upper boundary of the Magic Formula barrier, leaving minimal safety margin.

For verification methodology, this finding underscores that the safety guarantees are contingent on the friction model. The formal proof framework is model-agnostic: substituting the friction model requires only updating the $\mu(\lambda)$ definition in `MS_NN.v` and recomputing the barrier constants—approximately 10 lines of code change, with the remainder of the 866-line codebase unaffected.

8.9. Multi-Surface Barrier Extension

The current barrier certificate verifies pointwise boundary conditions for all four road surfaces. Extending the verification from pointwise conditions to full interval-wide guarantees (proving $N(T_b^{\text{low}}, \lambda) \geq 0$ and $N(T_b^{\text{high}}, \lambda) \leq 0$ for all $\lambda \in [0.03, 0.40]$) remains open—the interval-wide conditions are mathematically false for some surfaces, requiring a refinement of the barrier argument. A promising direction is to formalize the surface-dependent parameter adaptation mechanism of the MS-NN: the NN estimates the road friction, selects the appropriate FSM parameter set, and the FSM safety is guaranteed for each parameter set independently. This would provide end-to-end verification of the full surface-adaptive ABS system.

8.10. Complete ODE Formal Proof

The current barrier certificate uses pointwise evaluation of $d\lambda/dt$ at the boundaries rather than a full differential invariant proof. A complete formal proof would require formalizing the ODE solution using Coqelicot [27] (Rocq’s real analysis library) and proving the forward invariance property through differential invariants. This involves establishing the Lipschitz continuity of the Pacejka Magic Formula (via bounded derivatives of $\sin$ and $\arctan$), proving the existence and uniqueness of ODE solutions in Rocq, and then deriving the barrier-based invariance from the sign conditions at the boundaries. This is a significantly more complex proof but would provide complete safety guarantees without the pointwise evaluation assumption.

8.11. Automated Proof Generation

The current Rocq proofs are hand-written and require significant expertise in both Flocq and Rocq’s real arithmetic. A natural extension is to develop domain-specific automation tools that can generate Flocq error bounds for arbitrary feedforward neural networks, given their architecture and weight bounds. This would dramatically reduce the verification effort for new NN controllers.

8.12. Hardware-in-the-Loop Integration

The Flocq-based error bounds can be integrated with Hardware-in-the-Loop (HIL) testing frameworks. By providing certified upper bounds on the maximum deviation between the ideal NN computation and the actual float32 hardware execution, the verification framework enables rigorous pass/fail criteria for HIL tests.

9. Conclusions

We have presented a compositional formal verification framework for MS-NN ABS controllers, implemented entirely within the Rocq theorem prover using the Flocq floating-point library. The framework decomposes the verification into independent NN error analysis (Flocq) and hybrid system safety proof (barrier certificate), connected through a simple contractual interface.

The core technical contribution is a complete Flocq proof (**Qed**) that the binary32 rounding error is bounded by $\varepsilon_{\text{machine}}$ per arithmetic operation. We integrate this with

a multi-surface barrier certificate (CoqInterval-verified pointwise, eight theorems) guaranteeing $N(T_b^{\text{low}}, 0.03) > 0$ and $N(T_b^{\text{high}}, 0.40) < 0$ across all four road surfaces. Together with the `last_crossing_point` lemma, these results establish forward invariance of $\lambda \in [0.03, 0.40]$.

Recent NN verification competitions [28] and abstraction methods [20] have advanced the state of the art. The framework demonstrates that **single-toolchain compositional verification of floating-point neural controllers is feasible**, directly supporting ISO 26262 certification for safety-critical automotive AI systems. Numerical simulation (excluding the 0.3 s settling period and $v < 1.5$ m/s regime) confirms that the NN controller achieves 100% closed-loop safety on all four surfaces, while the FSM achieves 96.0–100%. Current limitations include the reliance on a simplified quarter-car model (which does not capture lateral dynamics or load transfer), the pointwise nature of the barrier conditions (rather than a full interval-wide proof), and a single Admitted lemma (`relu_error_bound`).

Future work includes extending the vehicle model to capture lateral dynamics and actuator delay; completing the interval-wide barrier proof with higher-precision CoqInterval configurations; developing domain-specific automation for generating Flocq error bounds from arbitrary NN architectures; and applying the framework to other NN-controlled cyber-physical systems such as steer-by-wire and adaptive cruise control.

Funding: This work was supported by the Research Foundation of Huangshan University under Grant 2020xkj004.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: The author acknowledges the developers of the Rocq Platform and the Flocq library for providing the foundational formal verification infrastructure. We used Wolfram Mathematica 15.0 for exact symbolic verification of the barrier conditions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Savaresi, S.M.; Tanelli, M. *Active Braking Control Systems Design for Vehicles*; Springer: London, UK, 2010. <https://doi.org/10.1007/978-1-84996-350-3>.
2. Piccinini, M.; Zumerle, M.; Betz, J.; Rosati Papini, G.P. A Road Friction-Aware Anti-Lock Braking System Based on Model-Structured Neural Networks. *IEEE Open J. Intell. Transp. Syst.* **2025**, *6*, 523–538. <https://doi.org/10.1109/OJITS.2025.3563347>.
3. Nipkow, T.; Paulson, L.C.; Wenzel, M. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*; LNCS; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2283. <https://doi.org/10.1007/3-540-45949-9>.
4. Fulton, N.; Mitsch, S.; Quesel, J.D.; Völz, M.; Platzer, A. KeYmaera X: An Axiomatic Tactical Theorem Prover for Hybrid Systems. In *Proceedings of the 25th International Conference on Automated Deduction (CADE-25), Berlin, Germany, 1–7 August 2015*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 527–538. https://doi.org/10.1007/978-3-319-21401-6_36.
5. Platzer, A. *Logical Foundations of Cyber-Physical Systems*; Springer: Berlin/Heidelberg, Germany, 2018. <https://doi.org/10.1007/978-3-319-63588-0>.
6. Zhang, H.; Weng, T.W.; Chen, P.Y.; Hsieh, C.J.; Daniel, L. Efficient Neural Network Robustness Certification with General Activation Functions. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Montreal, QC, Canada, 3–8 December 2018*; pp. 4939–4948.
7. Katz, G.; Huang, D.A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zejčić, A.; et al. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In *Proceedings of the 31st International Conference on Computer Aided Verification (CAV), New York, NY, USA, 15–18 July 2019*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 443–452. https://doi.org/10.1007/978-3-030-25540-4_26.
8. Boldo, S.; Melquiond, G. Flocq: A Unified Library for Proving Floating-Point Algorithms in Coq. In *Proceedings of the 20th IEEE Symposium on Computer Arithmetic (ARITH-20), Tübingen, Germany, 25–27 July 2011*; pp. 243–252.
9. Boldo, S.; Melquiond, G. *Computer Arithmetic and Formal Proofs: Verifying Floating-Point Algorithms with the Coq System*; Elsevier: London, UK, 2018. <https://doi.org/10.1016/C2015-0-01301-6>.

10. The Coq Development Team. The Coq Proof Assistant, 2025. Rocq Version 9.0.1. Available online: <https://coq.inria.fr> (accessed on 1 December 2025).
11. Katz, G.; Barrett, C.W.; Dill, D.L.; Julian, K.; Kochenderfer, M.J. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Proceedings of the 29th International Conference on Computer Aided Verification (CAV), Heidelberg, Germany, 24–28 July 2017*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 97–117. https://doi.org/10.1007/978-3-319-63387-9_5.
12. Gehr, T.; Mirman, M.; Drachler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; Vechev, M.T. AI²: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P), San Francisco, CA, USA, 20–23 May 2018*; pp. 3–18. <https://doi.org/10.1109/SP.2018.00058>.
13. Martin-Dorel, É.; Melquiond, G.; Roux, P. Enabling Floating-Point Arithmetic in the Coq Proof Assistant. *J. Autom. Reason.* **2023**, *67*, 1439–1477. <https://doi.org/10.1007/s10817-023-09679-x>.
14. Alur, R. Formal Verification of Hybrid Systems. In *Proceedings of the 11th International Conference on Embedded Software (EMSOFT), Taipei, Taiwan, 9–14 October 2011*; pp. 273–278. <https://doi.org/10.1145/2038642.2038685>.
15. Loos, S.M.; Platzer, A.; Nistor, L. Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified. In *Proceedings of the 17th International Symposium on Formal Methods (FM), Limerick, Ireland, 20–24 June 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 42–56. https://doi.org/10.1007/978-3-642-21437-0_6.
16. Prebet, E.; Teuber, S.; Platzer, A. Verification of Autonomous Neural Car Control with KeYmaera X. In *Proceedings of the Rigorous State-Based Methods, Düsseldorf, Germany, 10–13 June 2026*; Springer Nature: Cham, Switzerland, 2026; pp. 288–307. https://doi.org/10.1007/978-3-031-94533-5_17.
17. Ivanov, R.; Weimer, J.; Alur, R.; Pappas, G.J.; Lee, I. Verisig: Verifying Safety Properties of Hybrid Systems with Neural Network Controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC), Montreal, QC, Canada, 16–18 April 2019*; pp. 169–178. <https://doi.org/10.1145/3302504.3311806>.
18. Xiang, W.; Tran, H.D.; Johnson, T.T. Output Reachable Set Estimation and Verification for Multilayer Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* **2018**, *29*, 5777–5783. <https://doi.org/10.1109/TNNLS.2018.2808470>.
19. Bak, S.; Dohmen, T.; Subramani, K.; Trivedi, A.; Velasquez, A.; Wojciechowski, P. The Hexatope and Octatope Abstract Domains for Neural Network Verification. *Form. Method. Syst. Des.* **2024**, *64*, 178–199. <https://doi.org/10.1007/S10703-024-00457-Y>.
20. Boudardara, F.; Boussif, A.; Meyer, P.J.; Ghazel, M. INNAbstract: An INN-Based Abstraction Method for Large-Scale Neural Network Verification. *IEEE Trans. Neural Networks Learn. Syst.* **2024**, *35*, 18455–18469. <https://doi.org/10.1109/TNNLS.2023.3316551>.
21. ISO 26262:2018; Road Vehicles—Functional Safety. International Organization for Standardization: Geneva, Switzerland, 2018.
22. ISO/PAS 21448:2022; Road Vehicles—Safety of the Intended Functionality. International Organization for Standardization: Geneva, Switzerland, 2022.
23. AUTOSAR. *Adaptive Platform—Release R23-11*; Technical Specification; AUTOSAR Consortium: Munich, Germany, 2023.
24. Pacejka, H.B. *Tire and Vehicle Dynamics*, 3rd ed.; Butterworth-Heinemann: Oxford, UK, 2012. <https://doi.org/10.1016/C2010-0-68548-8>.
25. Prajna, S.; Jadbabaie, A. Safety Verification of Hybrid Systems Using Barrier Certificates. In *Proceedings of the 7th International Workshop on Hybrid Systems: Computation and Control (HSCC), Philadelphia, PA, USA, 25–27 March 2004*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 477–492. https://doi.org/10.1007/978-3-540-24743-2_32.
26. Teuber, S.; Mitsch, S.; Platzer, A. Provably Safe Neural Network Controllers via Differential Dynamic Logic. In *Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 10–15 December 2024*; Curran Associates, Inc.: New York, NY, USA 2024; Volume 37, pp. 1586–1624. <https://doi.org/10.52202/079017-0051>.
27. Boldo, S.; Lelay, C.; Melquiond, G. Coquelicot: A User-Friendly Library of Real Analysis for Coq. *Math. Comput. Sci.* **2015**, *9*, 41–62. <https://doi.org/10.1007/s11786-014-0181-1>.
28. Brix, C.; Bak, S.; Johnson, T.T.; Wu, H. The Fifth International Verification of Neural Networks Competition (VNN-COMP 2024): Summary and Results. *arXiv* **2024**, arXiv:2412.19985. <https://doi.org/10.48550/ARXIV.2412.19985>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.