Iterated Narrow Networks: Turing Completeness at Width 4 on Convex Input Spaces and the Complexity of Verification

Olivier Bournez
Institut Polytechnique de Paris, LIX
91128 Palaiseau, France
olivier.bournez@lix.polytechnique.fr

Johanne Cohen
LISN, CNRS, Université Paris-Saclay
91190 Gif-sur-Yvette, France
johanne.cohen@lisn.fr

Adrian Wurm
Computer Science Institute
BTU Cottbus-Senftenberg
D-03046 Cottbus, Germany
wurm@b-tu.de

Abstract

We study the iterates of a fixed ReLU network N of width 4, that is, the discrete-time system $x_{k+1} = N(x_k)$ on $[0,1]^4$. Equivalently, x_k is the output of a weight-tied network made of k copies of N, or of a recurrent network with no external input, started at x_0. We prove that every deterministic Turing machine M admits such a network N_M. The network can be constructed in polynomial time from M. Each iteration has depth $\mathrm{poly}(|M|)$. The iterates of N_M simulate M exactly on every encoded tape. The number of iterations is proportional to the running time of M. Moreover, the fourth coordinate irreversibly flags every point of the convex cube $[0,1]^4$ whose simulation reads an invalid digit. This flag is the certificate that makes hardness reductions sound on *convex* input regions, the regime of practical verifiers. Convex regions were previously inaccessible because of a continuity obstruction inherent to ReLU networks. The correspondence has three consequences. First, reachability for iterated width-4 ReLU networks spans the full complexity hierarchy along the single axis of the time bound: it is NP-complete for a unary bound, NEXP-complete for a binary bound and RE-complete (undecidable) with no bound. Second, two extra neurons of width suffice to extend the hardness to every common smooth activation: at width 6, the robust verification problem (a promise problem with $\delta/2\delta$ gap) is NP-complete for $\tanh$, sigmoid, GELU, Swish, ELU, and any polynomial-time-computable continuous non-polynomial activation with a polynomial-time-computable Lipschitz constant. Third, the whole spectrum transfers to discretized Neural ODEs in dimension 5.

1 Introduction

We study the iterates of a fixed ReLU network of constant width: the discrete-time system

$$x_{k+1} \;=\; N(x_k), \qquad x_0 \in [0,1]^4,$$

where $N\colon [0,1]^4 \to [0,1]^4$ is a feedforward ReLU network of width 4. In other words, the object of study is a recurrent network that takes no per-step input: its input is its initial state x_0, and its outputs are its states x_k. Iterating one fixed block is the pattern behind deep equilibrium models [Bai et al., 2019] and discretized Neural ODEs [Chen et al., 2018, Kidger, 2022]. Residual and highway

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

networks [He et al., 2016, Srivastava et al., 2015] follow the same pattern with untied weights. We ask what the iterates of a width-4 block can compute, and what it costs to verify their behavior.

The correspondence. We prove that iterated width-4 ReLU networks are Turing complete on *convex* input regions, with iteration count playing the role of time. For every Turing machine (TM) M, we construct in polynomial time a single network $N_M\colon [0,1]^4 \to [0,1]^4$ of width 4 and depth $\mathrm{poly}(|M|)$ per iteration. Its iterates simulate M exactly on encoded tapes, and irreversibly flag every point of $[0,1]^4$ whose simulation reads an invalid digit (Theorem 3). Instantiated on a universal TM U, the construction yields a single fixed block N_U that simulates every machine on every input. The consequence is a hardness ladder along the single axis of the time bound (Theorem 7). At fixed width 4, bounded-time reachability is NP-complete for a unary bound, NEXP-complete for a binary bound and undecidable with no bound. NP-hardness of ReLU verification has been known since Katz et al. [2017]; the ladder shows that restricting the width does not restore tractability, since width 4 already suffices (Section 2.2). The hardness extends at width 6 to every common smooth activation (Theorem 10) and, through the correspondence of Duan et al. [2024], to discretized Neural ODEs at dimension 5 (Theorem 50).

The obstruction, in plain terms. A TM tape is an infinite binary string, and to feed it to a network we must store it in a single real number. Writing the tape in base 2 fails twice over: $0.0\overline{1}_2 = 0.1_2$, so two different tapes can be the same number, and reading off the first bit is a discontinuous operation. Writing it in base 4 using only the digits 1 and 3, a device going back to Siegelmann and Sontag [1991], repairs both defects. The encoding becomes injective, and the two possible leading digits are separated by a fixed gap, so a continuous piecewise-affine gadget can read the first digit. The price is that the valid encodings form a Cantor-like set $\mathbb{DD}_{1,3}$ of measure zero, whereas verification asks a question about a full-dimensional convex region. The two domains cannot be reconciled by any continuous preprocessing: any continuous map from a convex set into $\mathbb{DD}_{1,3}$ is constant (Proposition 11), and ReLU networks compute only continuous piecewise-affine maps. Since continuous encoding is impossible, our network *detects*: a fourth coordinate accumulates, one base-4 digit per iteration, a continuous certificate that irreversibly flags every input whose simulation reads an invalid digit. This online certificate is the ingredient that is new at the level of network construction. It is what makes every verification reduction below sound on convex regions.

Four theorems. Formal statements are in Section 2, proofs in the appendices.

(Th.3) Turing completeness of iterated width-4 networks on convex input spaces. For every TM M, a single polynomial-time-constructible width-4 ReLU network simulates M when iterated, exactly on encoded inputs, and irreversibly flags every point of the convex cube $[0,1]^4$ whose simulation reads an invalid digit.

(Th.7) Verification spectrum. Bounded-time reachability for iterated width-4 ReLU networks is NP-complete with unary time bound, NEXP-complete with binary time bound and RE-complete (undecidable) with no bound.

(Th.10) Robust verification at width 6 for smooth activations. For every continuous, non-polynomial, C^1-at-a-point, polynomial-time-computable activation ρ, the δ-robust verification problem (a promise problem with $\delta/2\delta$ gap) for width-6 ρ-networks is NP-complete.

(Th.50) Neural ODE consequences at dimension 5 (App.H). Through the correspondence of Duan et al. [2024] between vanilla feedforward networks and splitting-method discretizations, the whole spectrum transfers to the discretized reachability of Neural ODEs at dimension 5.

Positioning. Table 1 situates the results; its mode column records whether a statement concerns a feedforward network or the iterates of one (a recurrent network). Siegelmann–Sontag Siegelmann [1999], Siegelmann and Sontag [1991, 1994] established TM universality for recurrent networks; in the variant relevant here, where the input is placed in the initial state, their construction requires fractal-encoded inputs and does not optimize width. (Their results with per-step input sequences carry no such restriction, but answer a different question.) Bounded-width *approximation* by feedforward networks was developed by Hanin and Sellke [2017], Park et al. [2020], Telgarsky [2016]. Bournez et al. [2025] contains two distinct results: a width-3 network simulating one TM step exactly on the fractal encoding set, and a separate width-12 uniform approximation theorem for arbitrary continuous piecewise-affine maps on convex domains. Theorem 3 improves neither width. It establishes a

different property: a single width-4 network whose iterates are exact on encoded inputs and certify non-encoded inputs, uniformly over the whole convex cube $[0,1]^4$ (Section 3.1 explains why neither prior result yields this). Verification of ReLU networks was previously known to be NP-hard via SMT-style encodings Katz et al. [2017]; (Th.7) lifts this to the full NP / NEXP / RE spectrum along the time bound, and (Th.10) extends it to smooth activations at width 6. Training hardness is known up to $\exists\mathbb{R}$-completeness Blum and Rivest [1988], Abrahamsen et al. [2021], Froese et al. [2022]. The question we study is also the narrow-network analogue of a line of work on how expressive power grows with the number of sequential steps: Merrill and Sabharwal [2024] show that transformers gain expressive power as the number of chain-of-thought steps grows, and ours is the same trade-off in a different architecture class, pushed to its limit. Finally, the Neural ODE consequence (Th.50) uses the discrete-to-continuous correspondence Chen et al. [2018], Duan et al. [2024].[1]

	mode	input domain	guarantee	width	activation
Siegelmann–Sontag [1991, 1994, 1999]	recurrent	fractal $\mathbb{DD}_{1,3}$	exact TM simulation	not optimized	saturated linear
Hanin and Sellke [2017] Park et al. [2020]	feedforward	compact $\mathbb{R}^d$	uniform approx.	$d{+}1$ to $d{+}4$	ReLU
Bournez et al. [2025], main thm.	feedforward	convex $[0,1]^d$	uniform approx. of CPAMs	12	ReLU
Bournez et al. [2025], Thm. 23	iterated	fractal $\mathbb{DD}_{1,3}$	exact TM step; no claim off fractal	3	ReLU
Bournez et al. [2026]	iterated	fractal + convex	NP-hardness via 3-SAT	3, 4	ReLU
this paper, Th.3	iterated	convex $[0,1]^4$	**exact TM simulation + certificate**	4	ReLU
this paper, Th.10	both	convex $[0,1]^d$	robust verif. NP-c.	6	smooth ρ

Table 1: Positioning. The mode column distinguishes feedforward statements from statements about the iterates of a fixed block. (Th.3) improves neither width of Bournez et al. [2025]: it is the first exact TM simulation whose soundness extends to a full-dimensional convex input region, which is what the NEXP and undecidability statements require.

2 Model, problems, and main theorems

2.1 Networks, iteration, and verification

The model: iterated networks. The object of study in Theorems 3 and 7 is the discrete-time system $x_{k+1} = N(x_k)$ obtained by iterating a fixed feedforward ReLU network $N\colon [0,1]^4 \to [0,1]^4$ of width 4. In other words, it is a weight-tied recurrent network that takes no per-step input: its input is its initial state and its outputs are its states. We write $N^{[t]} := N \circ N \circ \cdots \circ N$ for the t-fold composition. Which representation a result concerns is stated with the result: the unary-bound NP-completeness also holds for feedforward width-4 networks, by unrolling (Corollary 8), while the NEXP and RE statements concern the iterated representation, whose unrolled depth would be exponential or unbounded.

We consider neural networks with rational weights, with the "enhanced neuron" convention (e.g. Kidger and Lyons [2020]): a neuron is a map $\psi \circ \rho \circ \phi$ with ϕ, ψ affine and ρ the activation. We write $\mathcal{NN}^{\rho}_{n,m,k}$ for the class of feedforward networks with n inputs, m outputs, ρ activation, and bounded width k (i.e. every hidden layer has at most k neurons).

Verification problems. Our hardness results concern the following decision problems.

Definition 1 (Reachability)**.** The *reachability* problem takes as input a ρ-network $N\colon [0,1]^d \to [0,1]^d$ and two convex regions $\mathrm{Init}, \mathrm{Acc} \subseteq [0,1]^d$ specified by rational linear inequalities, and asks whether $\exists\, x \in \mathrm{Init},\ \exists\, t \geq 0,\ N^{[t]}(x) \in \mathrm{Acc}$.

Here, the network is presented as a weighted graph, where the encoding of each hidden layer is the adjacency matrix, but every entry is the numerator-denominator-pair describing the rational weight.

[1] Companion work Bournez et al. [2026] establishes NP-completeness at widths 3 and 4 via fractal preprocessing from 3-SAT, with #P/PP corollaries on the fractal encoding set. The TM-simulation, NEXP, undecidability and smooth-activation contributions of the present paper are disjoint; the only overlap is Corollary 8 (NP-completeness of width-4 verification on convex input), which here arises as a corollary of Theorem 3 (a strictly stronger primitive whose further consequences are illustrated by Theorems 7, 10 and 50 and follow more generally from standard complexity theory Sipser [1997], Arora and Barak [2009]).

This formulation follows the standard one in the neural-network verification literature (see e.g. Katz et al., 2017, Wurm, 2023a).

Definition 2 (Bounded-time reachability)**.** The *T-bounded reachability* problem takes the same input plus a time bound $T \in \mathbb{N}$ (encoded either in unary or in binary) and asks whether $\exists\, x \in \mathrm{Init},\ \exists t \leq T,\ N^{[t]}(x) \in \mathrm{Acc}$.

When $\rho = \mathrm{ReLU}$, both problems admit NP-membership for unary T: by the small-witness property for piecewise-affine systems (Lemma 36, classical), it suffices to guess a rational x of polynomial bit complexity and simulate T steps with exact rational arithmetic. For binary T, no such polynomial-time simulation exists; we recover NEXP-membership in Section 4.

2.2 Why reachability, why convex regions, why iterates

Robustness certification is exactly the reachability question we study: an ℓ_∞ ball around a data point $\bar{x}$ is an axis-aligned box, hence a convex polytope specified by rational linear inequalities, and the certification query is literally $\exists x \in \mathrm{Init}\colon N(x) \in \mathrm{Acc}$, with Acc the region where the decision is wrong. This is the VNN-LIB and VNN-COMP specification format [Brix et al., 2023], decided by complete verifiers such as Reluplex, Marabou and α, β-CROWN [Katz et al., 2017, Wang et al., 2021]. Convex initial regions are therefore imposed by the application. Iteration matters wherever one block is applied repeatedly, above all in closed-loop verification of a neural controller, where the reachable set over k control steps is exactly $N^{[k]}(\mathrm{Init})$. The practical reading of our results is that certification cost scales along depth, or number of steps, and not along width: at width 4, relaxing only the time bound climbs $\mathrm{PTIME} \to \mathrm{NP} \to \mathrm{NEXP} \to$ undecidable, and the remaining route to tractability is a restriction on the dynamics or on the parameters (Section 6).

2.3 Encoding Turing machines

We encode Turing-machine (TM) configurations using base-4 digits $\{1, 3\}$, a Cantor-style encoding in the lineage of Siegelmann and Sontag [1991, 1994]. Allowing all digits would create ambiguities like $0.1\overline{3}_4 = 0.2_4$; restricting to digits 1 and 3 makes the presentation of a number unique, and resolves the discontinuity issues inherent to ReLU iteration. A word $u = u_1 \cdots u_n \in \{0,1\}^n$ maps to $\mathrm{enc}_{\{1,3\}}(u) = \sum_{i=1}^{n} (2u_i + 1)\, 4^{-i} \in [0, 1]$. A configuration $c = (u, q, v)$ (current state q, tape to the left of the head u (reversed), tape from the head onward v) is encoded as

$$\mathrm{enc}_M(c) \;=\; \big(\mathrm{enc}_{\{1,3\}}(u),\ \mathrm{enc}_M(q, v),\ 0,\ 0\big) \;\in\; [0,1]^4,$$

where $\mathrm{enc}_M(q, v)$ co-encodes the state and the right tape into a single base-4 string with $\mathrm{enc}_M(q)$ as a constant-length prefix followed by $\mathrm{enc}_{\{1,3\}}(v)$, based on a packing borrowed from Bournez et al. [2025] that saves one coordinate over the naive encoding $(\mathrm{enc}_{\{1,3\}}(u), \mathrm{enc}_{\{1,3\}}(v), \mathrm{enc}_M(q), \cdot, \cdot)$ which would force width 5. The third coordinate is a scratch register used by the simulator; the fourth is a validity register, both initially 0 (see Section 4). We write Q for the state set of M, $\mathrm{enc}_M(Q) \subseteq [0, 1]$ for the prefix-encoded set of states, $\mathbb{DD}_{1,3} \subseteq [0, 1]$ for the Cantor-like set of points whose base-4 expansion uses only digits $\{1, 3\}$, π_i for the i-th coordinate projection, step_M for the map sending a configuration of M to the next one, and $c_t(\tau) := \mathrm{step}_M^t(c_0(\tau))$ for the configuration at step t on an infinite tape $\tau \in \{0,1\}^\omega$, with $c_0(\tau)$ the initial configuration. The *state prefix* of x_2 consists of its first $|q|_4 = \lceil \log_4 |Q| \rceil$ base-4 digits, and the *post-state-prefix* (or tape-suffix) portion of x_2 is the remainder. We say that a digit of x_1, or of the tape-suffix of x_2, is *visited during the first k iterations* of the network of Theorem 3 if it is exposed as the leading digit of its register at some iteration $j < k$. When x encodes a configuration, the visited digits are exactly the tape cells read by M's head.

2.4 Turing completeness of iterated width-4 networks on convex input spaces

Theorem 3 (Turing completeness on convex input spaces)**.** *For every deterministic Turing machine M over the binary alphabet, one can construct in polynomial time a single* ReLU *network $N_M\colon [0,1]^4 \to [0,1]^4$ of width 4 and depth $D_M = \mathrm{poly}(|M|)$, with a constant $C_M \in \mathbb{N}$, such that:*

(i) (Faithful simulation, all time.) For every $\tau \in \{0,1\}^\omega$ and every $t \geq 0$, $N_M^{[C_M\, t]}\big(\mathrm{enc}_M(c_0(\tau))\big) = \mathrm{enc}_M\big(\mathrm{step}_M^t(c_0(\tau))\big)$. (For finite inputs w, take $\tau = w \cdot \tau'$ for any suffix τ'; M's behavior up to time T depends only on the first $\leq T$ symbols.)

(ii) (Online validity certificate.) The fourth coordinate $\pi_4(N_M^{[k]}(x))$ is non-decreasing in k and bounded in $[0,1]$. Moreover, for $x = (x_1, x_2, x_3, x_4)$ with $x_4 = 0$, we have $\pi_4(N_M^{[k]}(x)) = 0$ iff every leading base-4 digit of x_1 and of the post-state-prefix portion of x_2 visited by M's head during the first k iterations lies in $\{1, 3\}$, and the state prefix of x_2 stays in $\mathrm{enc}_M(Q)$ along the trajectory. In particular, $\pi_4(N_M^{[k]}(x)) = 0$ for all $k \geq 0$ whenever x_1 and the v-part of x_2 lie in $\mathbb{DD}_{1,3}$ and the state prefix of x_2 encodes a valid state.

Part (i) alone holds at width 3 (Theorem 12, due to Bournez et al., 2025).

Remark 4 (Online certificate; single fixed network). Part (ii) is an *online* certificate: the fourth coordinate is updated one digit per iteration via a width-2 sub-gadget, so N_M is uniformly polynomial in $|M|$, independent of input length. (A finite ReLU network cannot have its zero set equal to $\mathbb{DD}_{1,3}$ exactly, hence a check spread over the iterations.) Taking $M = U$ a universal Turing machine, $\mathrm{enc}_U(\langle M', w\rangle)$ embeds an arbitrary pair (M', w) into a single point of $[0,1]^4$, and N_U becomes a single fixed network simulating any TM M' on any input w. Width 4 is the price of the pair (i)+(ii), and we claim no lower bound for the pair.

The combination of (i) and (ii) is what makes the verification reductions work: property (i) gives soundness on encoded inputs, and property (ii) on the rest of the convex region. Plugging in halting problems of varying time complexity yields the hardness spectrum below.

Remark 5 (Convex inputs are the range of a quantifier, not a tolerance). Theorem 3 is not a robustness property: part (i) requires an exact encoding, and nothing is claimed about inputs near an encoding beyond part (ii). The convex region is the range of the existential quantifier in the verification problems, where it is imposed by the application (Section 2.2); part (ii) is what holds on the rest of that region.

Remark 6 (No certificate hypothesis in the complexity results). Certifying, in the sense of part (ii), is a capability of the witness network, not a hypothesis on a model class: Definitions 1 and 2 and Theorem 7 and Corollaries 8 and 14 quantify over *all* width-4 ReLU networks. Remark 34 makes the certified set precise.

2.5 Verification spectrum of iterated width-4 networks

Theorem 7 (Verification spectrum at width 4)**.** *For iterated width-4 ReLU networks $N\colon [0,1]^4 \to [0,1]^4$ (the system of Section 2.1), T-bounded reachability is:*

(i) NP*-complete when T is encoded in unary;*
(ii) NEXP*-complete when T is encoded in binary;*
(iii) RE*-complete (hence undecidable) with no time bound.*

The hardness statements hold even when Init *and* Acc *are axis-aligned hyperrectangles with rational endpoints.*

Corollary 8 (Verification of feedforward width-4 ReLU networks)**.** *Verification of* feedforward *width-4 ReLU networks on convex $\mathrm{Init}, \mathrm{Acc} \subseteq [0,1]^4$ (deciding whether $\exists x \in \mathrm{Init} : N(x) \in \mathrm{Acc}$, a single application of N) is* NP*-complete.*

2.6 Robust verification at width 6 for smooth activations

Why "robust". Exact verification of ρ-networks for smooth ρ such as $\tanh$ quickly leaves classical complexity classes: the relevant decision problems involve transcendental constraints whose decidability rests on contested hypotheses [Macintyre et al., 1996, Wurm, 2023a, 2024]. The standard remedy is robust verification, which allows a fixed tolerance with a promise gap.

Definition 9 (δ-robust verification)**.** Given a ρ-network $N\colon [0,1]^d \to [0,1]^d$, convex regions $\mathrm{Init}, \mathrm{Acc} \subseteq [0,1]^d$, and a rational $\delta > 0$, the *δ-robust verification problem* is the **promise** problem:

- Output YES if $\exists\, x \in \mathrm{Init}$ with $N(x) \in \mathrm{Acc}^{+\delta}$;
- Output NO if $\forall\, x \in \mathrm{Init}$, $N(x) \notin \mathrm{Acc}^{+2\delta}$;
- Promise: the input falls into one of these two cases.

Here $\mathrm{Acc}^{+\delta} = \{y \in \mathbb{R}^d : dist(y, \mathrm{Acc}) \leq \delta\}$ and $dist$ is Euclidean distance. The T-bounded variant considers $N^{[t]}(x)$ for $t \leq T$.

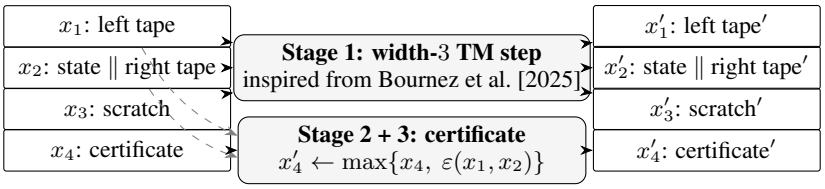

Figure 1: One iteration of the width-4 network of Theorem 3: x_1 holds the left tape, x_2 co-encodes the state and the right tape, x_3 is a scratch register, and x_4 accumulates the certificate ε via the max update, with $\varepsilon(x_1, x_2) = 0$ iff the leading digits and the state prefix are valid (Remark 33).

Theorem 10 (Robust verification at width 6)**.** *Let $\rho\colon \mathbb{R} \to \mathbb{R}$ satisfy:*

- *(H1) ρ is continuous and non-polynomial;*
- *(H2) ρ is C^1 at some point with non-zero derivative;*
- *(H3) ρ is polynomial-time computable, and admits a polynomial-time-computable Lipschitz constant on every bounded interval.*

Then δ-robust verification and δ-robust T-bounded reachability (with T unary) for ρ-networks of width 6 are NP-complete[2].

Coverage of (H1)–(H3). (H1)–(H2) is the relaxation of differentiability of Kidger and Lyons [2020]; with (H3) they cover ReLU, leaky ReLU, $\tanh$, sigmoid, GELU, Swish, ELU, softplus, $\arctan$, the Gaussian and every analytic non-polynomial activation. Constructive $\rho \to \mathrm{ReLU}$ bit-complexity bounds are provided by Zhang et al. [2024]; the remaining cases follow from interval evaluation under (H3).

3 Proof sketch of Theorem 3

The proof of Theorem 3 has three modular components, illustrated in Figure 1: a width-3 subnetwork that simulates one Turing-machine step on the $\{1, 3\}$-base-4 encoding domain (*Stage 1*, at width 3, based on ideas from Bournez et al. [2025]); a width-2 subnetwork that produces a continuous encoding-validity certificate (*Stage 2*); and a combinator that propagates the certificate across iterations (*Stage 3*). The full network has width 4. We first identify the obstruction, then describe each stage; complete proofs are in Appendices B to D.

3.1 The continuity obstruction, and why exactness is forced

Why exactness, in one sentence. Each simulated tape shift multiplies a coordinate by 4 (Lemma 31), so a per-step error ϵ becomes about $4^k\epsilon$ after k iterations: an approximation accurate for k steps needs precision 4^{-k}, hence a network whose size grows with the time horizon. Theorem 7(ii)–(iii) quantify over exponential and unbounded k with one fixed finite network, so no approximation-based construction can reach them.

The obstruction. The existing ReLU-based simulations operate on the fractal set $\mathbb{DD}_{1,3}$ (totally disconnected, Lebesgue measure 0), whereas verification asks for a trajectory from a *convex* initial region $\mathrm{Init} \subseteq [0, 1]^4$. The two domains are incompatible in a strong sense.

Proposition 11 (No continuous encoding)**.** *Every continuous map from a convex subset of $\mathbb{R}^n$ into $\mathbb{DD}_{1,3}$ is constant.*

Proof. A convex set is connected, the continuous image of a connected set is connected, and $\mathbb{DD}_{1,3}$ is totally disconnected, so the image is a single point. □

ReLU networks realize only continuous piecewise-affine maps Arora et al. [2016], so no subnetwork can map Init onto valid encodings, and none can gate the simulation on the discontinuous predicate "is x a valid encoding?". Bournez et al. [2025] circumvent this by approximating the validity predicate

[2]Both are promise problems with a $\delta/2\delta$ gap; "NP-complete" is shorthand for "NP-hard for the YES instances of the promise and admitting an NP verifier on the promise," i.e. formally $\mathrm{promiseNP}$-complete in the sense of Goldreich [2006].

uniformly, at width 12 and at the cost of exact iteration; the companion work Bournez et al. [2026] restricts inputs by 3-SAT preprocessing, which yields NP-hardness but no TM simulation. We instead realize a *continuous* certificate $\varepsilon(x)$ that vanishes on valid encodings, is positive elsewhere and is non-decreasing along trajectories: any input which was never a valid encoding is flagged and stays flagged. The certificate occupies one extra coordinate, hence width 4.

Stage 1: width-3 TM step on encoded inputs. TM simulation at some fixed width has been known since Siegelmann and Sontag [1991]; Bournez et al. [2025] reach width 3, which we make fully explicit in Appendix B and use as a black box.

Theorem 12 (Width-3 step simulation, inspired from Bournez et al. [2025])**.** *For every Turing machine M over $\{0,1\}$, there exists a* ReLU *network $g_M \in \mathcal{NN}^{\mathrm{ReLU}}_{3,3,3}$ of depth $\mathrm{poly}(|M|)$ such that, for every valid encoded configuration $\mathrm{enc}_M(c) \in \mathbb{DD}_{1,3} \times \mathbb{DD}_{1,3_Q} \times \{0\}$, $g_M\big(\mathrm{enc}_M(c)\big) = \mathrm{enc}_M\big(\mathrm{step}_M(c)\big)$, where $\mathbb{DD}_{1,3_Q} \subseteq [0,1]$ denotes the set of co-encodings of a state prefix in $\mathrm{enc}_M(Q)$ followed by an $\mathbb{DD}_{1,3}$-tape suffix, and the third coordinate is a scratch register reset to 0 between steps.*

Theorem 12 treats the encoding domain as a set of exact rational points: each TM transition is an explicit affine map on the relevant cylinder of $\mathbb{DD}_{1,3} \times \mathbb{DD}_{1,3_Q}$, and the gaps between cylinders allow these maps to be assembled at width 3. Outside the encoded domain, g_M has no specified behavior, which is what forces the certificate: it is what lets us talk about verification on convex inputs rather than on the fractal encoding set.

Stage 2: continuous encoding-validity certificate. The new ingredient is a per-step error gadget $\mathrm{err}\colon [0,4] \to [0,1]$, continuous and piecewise affine, which vanishes exactly on $[1,2] \cup [3,4]$ and is positive elsewhere. It can be implemented as a sum of two tents by the nested ReLU expression

$$\mathrm{err}(x) = \mathrm{ReLU}\Big(3 - x - \mathrm{ReLU}\big(5 - 2x - \mathrm{ReLU}\big(2 - x + \mathrm{ReLU}(1 - x)\big)\big)\Big).$$

Applied to $4x_j$, err vanishes iff the leading base-4 digit of x_j lies in $\{1,3\}$ (scheduling inside the width-4 network: Appendix C).

Proposition 13 (Per-step encoding-validity gadget)**.** *The function $\mathrm{err}_4(x) := \mathrm{err}(4 \cdot x)$ is continuous, non-negative, computable in constant depth inside the width-4 register network of Theorem 3, and vanishes precisely on the leading-digit $\{1,3\}$-cylinders of $[0,1]$.*

Stage 3: combining the stages, and proof of Theorem 3. The full width-4 network N_M assembles, at each iteration, a *per-step certificate update* followed by Stage 1's TM step:

1. Apply the per-step error gadget err to $4x_1$ and to the leading digits of the tape-suffix portion of x_2 (Proposition 13; the state-prefix portion of x_2 is checked by the same gadget composed with a constant-depth slicing, Appendix C). Update $x_4' \leftarrow \max\{x_4, \varepsilon(x_1, x_2)\}$, where $\varepsilon(x_1, x_2)$ is the maximum of the digit-error and state-distance terms (Remark 33). The maximum is computed as $a + \mathrm{ReLU}(b - a)$ at width 2.
2. Apply Stage 1, the width-3 step simulation g_M, to (x_1, x_2, x_3) to obtain (x_1', x_2', x_3'). The fourth register x_4' is propagated unchanged through Stage 1 by an identity wire.

Each iteration of N_M has constant depth on top of g_M; thus the per-iteration depth of N_M is $D_M = \mathrm{poly}(|M|)$, independent of any input length or time horizon.

Scheduling and width-4 bound. Each step is a sequence of register-layer updates (Definition 15), each modifying one of $(x_1, \ldots, x_4)$: the error gadgets are width-2 and sequentialized so that x_4 serves as accumulator while x_1, x_2 are read-only and x_3 holds scratch data; Stage 1 then updates x_1, x_2, x_3 and preserves x_4 through an identity wire (register form of register width 4, Remark 42).

Proof of Theorem 3. ε is non-negative and bounded by 1; the $\max$ update keeps $x_4(k) := \pi_4(N_M^{[k]}(x))$ non-decreasing and in $[0,1]$. If x_1 and the tape-suffix of x_2 lie in $\mathbb{DD}_{1,3}$, and the state-prefix of x_2 is in $\mathrm{enc}_M(Q)$, every leading digit visited stays in $\{1,3\}$ and the state stays in $\mathrm{enc}_M(Q)$ along the trajectory (since g_M preserves valid encodings by Theorem 12); ε vanishes at every iteration, $x_4 = 0$ throughout, and $N_M^{[t]}(x) = \mathrm{enc}_M(c_t)$ by induction, proving (i). Conversely, when a visited leading digit falls outside $[1,2] \cup [3,4]$ or the state-prefix leaves $\mathrm{enc}_M(Q)$, the per-step

error is positive at that iteration; x_4 becomes positive and stays so. Stage 2 checks *before* Stage 1 reads, so the simulator never operates on a forbidden digit while $x_4 = 0$. □ (A fully worked example is given in Appendix D.4.)

4 Verification spectrum of iterated width-4 networks

4.1 Proof of Theorem 7

Reductions: a uniform recipe. For each part of Theorem 7, we reduce a classical TM complexity problem Π to width-4 reachability. The reduction is identical across the three parts, so we describe it once. Let Π ask whether some Turing machine M accepts some input within some (problem-dependent) time bound. By Theorem 3, applied to M, we obtain a width-4 ReLU network N_M with the simulation and certificate properties. Recall that the second coordinate x_2 co-encodes the state (constant-length prefix $\mathrm{enc}_M(q)$ of length $|q|_4 = \lceil \log_4 |Q| \rceil$) and the right tape (Section 2). Define

$$\mathrm{Init} \;=\; \big\{x \in [0,1]^4 \;:\; \mathrm{enc}_M(q_0) \le 4^{|q|_4} x_2 \le \mathrm{enc}_M(q_0) + 1,\; x_3 = 0,\; x_4 = 0\big\},$$

i.e. x_2 has the prefix $\mathrm{enc}_M(q_0)$ of the initial state, the remaining digits of x_2 encode an arbitrary right tape, x_1 is unrestricted in $[0,1]$ (left tape), and the scratch and validity registers are pinned to 0. Define $\mathrm{Acc} \;=\; \big\{x \in [0,1]^4 \;:\; \mathrm{enc}_M(q_F) \le 4^{|q|_4} x_2 \le \mathrm{enc}_M(q_F) + 1,\; x_4 \le 0\big\}$, i.e. the state-prefix of x_2 encodes the final state q_F. Both Init and Acc are convex and specified by rational linear inequalities (axis-aligned in x_2, x_3, x_4). The reduction maps (M, T) (or (M) in the unbounded case) to $(N_M, \mathrm{Init}, \mathrm{Acc}, T)$ (or $(N_M, \mathrm{Init}, \mathrm{Acc})$). Correctness is by Theorem 3: a trajectory enters Acc iff $x_4 = 0$ throughout, i.e. every step is locally valid in the sense of Lemma 35, and the state-prefix of x_2 encodes q_F at some iteration $\le T$. Equivalently, the trajectory enters Acc iff M accepts on some infinite binary tape $\tau \in \{0,1\}^\omega$ within the time bound, which matches the NP / NEXP / RE hardness sources by padding: acceptance on some infinite tape in T steps reduces to acceptance on some length-T prefix in T steps (see Lemma 37).

It remains only to choose Π for each part and verify the upper bound.

Part (i): unary T. Take Π to be "does M accept some $w \in \{0,1\}^n$ within T steps?" with T unary, which is NP-complete by Cook–Levin applied to a TM with the standard polynomial slowdown Arora and Barak [2009]. Hardness follows from the recipe above. NP-membership follows from Lemma 36: a non-deterministic verifier guesses a rational $x \in \mathrm{Init}$ of polynomial bit complexity and simulates N_M exactly for $T \cdot C_M$ steps. With T unary, total time is $\mathrm{poly}(|N_M|, T)$.

Part (ii): binary T. Take Π to be SUCCINCT-TM-ACCEPTANCE: "does M accept some w of length at most n in at most T steps?" with n, T binary. This problem is NEXP-complete by a standard padding argument from SUCCINCT-SAT Arora and Barak [2009]. Hardness follows from the recipe above with $T \cdot C_M$ binary as the new time bound. For membership: a non-deterministic verifier guesses $x \in \mathrm{Init}$ and $t \le T \cdot C_M$ in time polynomial in the input size; simulation of N_M for t iterations takes time $\mathrm{poly}(t, |N_M|) = 2^{\mathrm{poly}(\text{input})}$ because t may be exponential in the binary encoding $|T|$. The overall verifier runs in non-deterministic exponential time, hence NEXP-membership.

Part (iii): no time bound. Take Π to be the halting problem on the empty input: "does M halt on the empty word?", which is RE-complete. The recipe gives N_M together with the singleton $\mathrm{Init} = \{\mathrm{enc}_M(c_0(\varepsilon))\}$ (equivalent to a degenerate hyperrectangle with rational endpoints). A trajectory from this single point reaches Acc iff M halts on the empty word. Membership in RE is a standard semi-decision procedure (simulate N_M until the trajectory enters Acc, accept if so). □

A note on the NEXP *witness.* The witness in Part (ii) encodes a TM input of length up to n as the point $x \in \mathrm{Init}$; Remark 38 explains why this lands on NEXP rather than EXPTIME.

4.2 Two corollaries used in the sequel

Two consequences follow at no additional cost. By complementation, the dual problem "does $N(x) \in \mathrm{Acc}$ hold for every $x \in \mathrm{Init}$?" is coNP-complete (this is what complete verifiers like Marabou and α, β-CROWN Katz et al. [2017], Wang et al. [2021] certify), and by a gap construction using the certificate of Theorem 3, approximating $\min_{x \in \mathrm{Init},\, t \le T} N^{[t]}(x)$ within additive error $\eta < 1/4$ inherits the full spectrum.

Corollary 14 (Universal verification and approximation hardness)**.** *For width-4 ReLU networks:* ***(i)*** *universal verification is* coNP*-complete;* ***(ii)*** *approximating worst-case output within η is* NP*-hard (unary T),* NEXP*-hard (binary T), and undecidable (no T).*

5 Robust verification at width 6 for smooth activations

Theorem 7 addresses ReLU. We extend the hardness to every common smooth activation at the cost of two additional units of width, in the robust framework of Section 2.6: exact verification for smooth ρ leaves classical complexity [Macintyre et al., 1996, Wurm, 2024, 2023b], while the $\delta/2\delta$ promise gap [Gao et al., 2012] absorbs the $\rho \to \mathrm{ReLU}$ approximation error and matches the finite-precision regime of numerical verifiers. Theorem 10 shows this relaxation does not help.

5.1 ReLU-register networks and ρ-emulation

The reduction from ReLU width 4 to ρ width 6 is most cleanly stated in a register normal form, in which each hidden layer updates a single coordinate. Every ReLU network of width w can be expressed in this form at a polynomial cost in depth.

Definition 15 (ReLU-register network)**.** A ReLU-*register network* of register width $k \geq 1$ and depth L is a composition $F = A_{\mathrm{out}} \circ R_L \circ \cdots \circ R_1 \circ A_{\mathrm{in}}$, where $A_{\mathrm{in}} \colon \mathbb{R}^d \to \mathbb{R}^k$ and $A_{\mathrm{out}} \colon \mathbb{R}^k \to \mathbb{R}^d$ are affine, and each layer $R_\ell \colon \mathbb{R}^k \to \mathbb{R}^k$ updates a single coordinate $j_\ell \in \{1, \ldots, k\}$ via $(R_\ell(x))_{j_\ell} = \mathrm{ReLU}(a_\ell \cdot x + b_\ell)$, leaving the others unchanged.

We work directly with this register form: it captures all ReLU networks (any standard ReLU network of width w and depth L embeds into a register network of register width $2w$ and depth $O(wL)$; for our purposes, we only need that N_M is natively in register form of register width 4, see Remark 42 in Appendix F.1). The technical core of the section is the following emulation result, adapted from Kidger and Lyons [2020].

Lemma 16 (Per-layer ρ-emulation at width $k+2$)**.** *Let $\rho \colon \mathbb{R} \to \mathbb{R}$ satisfy hypotheses (H1)-(H3) of Theorem 10. For every* ReLU*-register layer $R \colon \mathbb{R}^k \to \mathbb{R}^k$, every compact $K \subseteq \mathbb{R}^k$, and every $\eta > 0$, there exists a ρ-network $\widetilde{R} \in \mathcal{NN}^{\rho}_{k,k,k+2}$ of depth $O(1)$ such that $\sup_{x \in K} \|\widetilde{R}(x) - R(x)\|_\infty \leq \eta$. The construction time is polynomial in the description size of R and in $\log(1/\eta)$.*

Mechanism of the +2 overhead. The width $k+2$ decomposes into k *identity wires* (one per register coordinate, each implemented by one enhanced ρ-neuron via the local-identity approximation of Kidger and Lyons [2020, Lemma 4.1], see Proposition 40) plus a ρ-subnetwork of width 2 that approximates the univariate non-linearity $\mathrm{ReLU}(a \cdot x + b)$ on a bounded range (Kidger and Lyons [2020, Proposition 4.9]). The full proof of Lemma 16 is in Appendix F.2.

Composing per-layer emulators with a standard telescoping argument yields a network-level approximation result.

Lemma 17 (Network-level ρ-emulation)**.** *Let $F \colon \mathbb{R}^d \to \mathbb{R}^d$ be a* ReLU*-register network of register width k and depth L, and let $K \subseteq \mathbb{R}^d$ be compact. For every $\varepsilon > 0$, there exists a ρ-network $\widetilde{F} \in \mathcal{NN}^{\rho}_{d,d,k+2}$ of depth $O(L)$, with $\sup_{x \in K} \|\widetilde{F}(x) - F(x)\|_\infty \leq \varepsilon$, and constructible in time $\mathrm{poly}(|F|, \log(1/\varepsilon))$.*

5.2 Proof of Theorem 10

We prove NP-membership and NP-hardness of δ-robust verification for ρ-networks of width 6. The bounded-time variant (unary T) follows by the same argument with depth scaled by T. Throughout this section and Appendix F, we write R for the acceptance region Acc of Definition 9.

NP-membership. By Lemma 36 (applied to the ReLU-equivalent given by Lemma 16), a nondeterministic verifier can guess a rational $x \in \mathrm{Init}$ of polynomial bit complexity. It then computes $\widetilde{y} \approx N(x)$ to precision $\delta/4$ using polynomial-time evaluation of ρ and exact rational arithmetic for affine layers. By (H3), each ρ evaluation costs $\mathrm{poly}(\log(1/\delta))$ time and the polynomial-time-computable Lipschitz constant $L_\rho = \mathrm{poly}(\text{input})$ on the relevant compact yields total iteration error $\leq T \cdot L_\rho^T \cdot \varepsilon_{\mathrm{step}}$. Choosing $\varepsilon_{\mathrm{step}} = \delta/(4TL_\rho^T)$ requires $\log(1/\varepsilon_{\mathrm{step}}) = \mathrm{poly}(\text{input})$ bits of precision per step. The verifier returns YES iff $\widetilde{y} \in R^{+3\delta/2}$. On promise YES, $dist(\widetilde{y}, R) \leq \delta + \delta/4 = 5\delta/4 \leq 3\delta/2$ and the verifier accepts; on promise NO, $dist(\widetilde{y}, R) > 2\delta - \delta/4 = 7\delta/4 > 3\delta/2$ and the verifier rejects.

NP-hardness via robust reduction from ReLU. Corollary 14 (or directly the proof of Theorem 7) establishes that the δ_0-robust verification problem is NP-hard for ReLU networks at width 4 for some fixed $\delta_0 > 0$ (concretely, the construction yields a separation of $\Omega(1)$ between $N(x) \in R$ and $N(x) \notin R^{+1}$ on the relevant compact). We reduce this to robust ρ verification at width 6.

Given a width-4 ReLU instance $(N, \mathrm{Init}, R, \delta)$, since N_M is natively in register form of register width 4 (Remark 42), apply Lemma 17 on the compact $K = [0,1]^4$ with precision $\varepsilon = \delta/4$ to obtain a width-6 ρ-network $\widetilde{N}$ such that $\sup_K \|\widetilde{N} - N\|_\infty \leq \delta/4$. Output the ρ-instance $(\widetilde{N}, \mathrm{Init}, R^{+\delta}, \delta/4)$.
YES preservation. If the original instance is YES, then $\exists x \in \mathrm{Init}$ with $N(x) \in R^{+\delta}$. Then $dist(\widetilde{N}(x), R) \leq \delta + \delta/4 = 5\delta/4$, so $\widetilde{N}(x) \in (R^{+\delta})^{+\delta/4}$. The output instance is YES.
NO preservation. If the original is NO, then $\forall x \in \mathrm{Init}$, $dist(N(x), R) > 2\delta$, so $dist(\widetilde{N}(x), R) > 2\delta - \delta/4 = 7\delta/4$. We need $\widetilde{N}(x) \notin (R^{+\delta})^{+2(\delta/4)} = R^{+3\delta/2}$, which holds since $7\delta/4 > 3\delta/2$. The output instance is NO.

The reduction is polynomial-time (Lemma 17 is polynomial-time constructible by Lemma 44); promise is preserved; YES/NO map correctly. Hence $\delta/4$-robust verification of ρ-networks at width 6 is NP-hard. NP-membership above gives completeness. □

Two complements. The argument needs T unary: with T binary, the precision required is doubly exponential in the input, and a NEXP-style statement at binary T is open. The approximation hardness of Corollary 14 transfers to width-6 ρ-networks by the same reduction (Appendix F.3).

6 Discussion and open problems

What this paper establishes. The iterates of a single fixed ReLU network of width 4 simulate an arbitrary Turing machine, exactly on encoded inputs and with a certificate on the rest of the convex cube (Theorem 3); their verification spans the hierarchy from NP to undecidable along the time-bound axis (Theorem 7); the hardness extends to all standard smooth activations at width 6 in the robust framework (Theorem 10) and, through Duan et al. [2024], to Neural ODEs at dimension 5 (Theorem 50). The technical novelty over classical Turing-simulation theorems for narrow recurrent networks is a continuous encoding-validity certificate that resolves a continuity obstruction at the cost of one extra unit of width.

Practical implications. Depth, not width, is the axis along which verification cost scales super-polynomially (Section 2.2). This is consistent with the empirical observation that depth dominates the cost of complete verifiers [Katz et al., 2017, Wang et al., 2021], although our worst-case analysis does not by itself explain solver behavior on trained networks.

Open problems. Four questions remain. **(i) Width 2.** Width 1 admits polynomial-time verification, widths 3 and 4 admit NP-completeness via fractal preprocessing Bournez et al. [2026], and width 4 obeys the full spectrum of Theorem 7; width 2 is open. **(ii) Continuous-time simulation.** Theorem 50 holds under splitting-method discretization; whether a Lipschitz piecewise-affine ODE admits TM simulation by its continuous flow, at any dimension, is open (Appendix H.4). **(iii) Trained and constrained networks.** Our construction is essentially expanding: each tape shift multiplies by 4. First, contractivity: non-expansive iterates (spectral norm at most 1, as enforced in deep equilibrium models [Bai et al., 2019] and normalizing flows) cannot create digit resolution, and we believe that bounded-time reachability leaves the NEXP / RE regime under such a constraint. Second, parameterized complexity in the number of unstable neurons, the quantity that governs complete solvers in practice, rather than in width or depth. **(iv) Beyond piecewise-affine activations.** (H1)–(H3) cover the standard smooth activations but not softmax or attention. Extending to Transformer-style architectures is natural.

Acknowledgments and Disclosure of Funding

O. Bournez and J. Cohen acknowledge support from PGMO, and O. Bournez from ANR project OTC ANR-24-CE48-0335. A. Wurm acknowledges partial support from the BTU Graduate Research School. The authors declare no competing interests.

References

Mikkel Abrahamsen, Linda Kleist, and Tillmann Miltzow. Training neural networks $\exists\mathbb{R}$-complete. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, volume 34, pages 18293–18306, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/9813b270ed0288e7c0388f0fd4ec68f5-Abstract.html.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *CoRR*, abs/1611.01491, 2016. doi: 10.48550/arXiv.1611.01491. URL http://arxiv.org/abs/1611.01491.

Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. ISBN 978-0-521-42426-4. doi: 10.1017/cbo9780511804090. URL http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1):35–65, feb 1995. ISSN 0304-3975. doi: 10.1016/0304-3975(94)00228-B.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019. doi: 10.48550/arXiv.1909.01377.

Hugo Bazille, Olivier Bournez, Walid Gomaa, and Amaury Pouly. On the complexity of bounded time reachability for piecewise affine systems. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*, volume 8762 of *Lecture Notes in Computer Science*, pages 20–31. Springer, 2014. doi: 10.1007/978-3-319-11439-2_2. URL https://doi.org/10.1007/978-3-319-11439-2_2.

Vincent D. Blondel, Olivier Bournez, Pascal Koiran, and John N. Tsitsiklis. The stability of saturated linear dynamical systems is undecidable. *Journal of Computer and System Science*, 62(3):442–462, may 2001. doi: 10.1006/jcss.2000.1737. URL https://doi.org/10.1006/jcss.2000.1737.

Avrim Blum and Ronald L. Rivest. Training a 3-node neural network is np-complete. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 1, [NIPS Conference, Denver, Colorado, USA, 1988]*, volume 1, pages 494–501. Morgan Kaufmann, 1988. doi: 10.1016/s0893-6080(05)80010-3. URL http://papers.nips.cc/paper/125-training-a-3-node-neural-network-is-np-complete.

Olivier Bournez and Michel Cosnard. On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science*, 168(2):417–459, nov 1996. ISSN 0304-3975. doi: 10.1016/S0304-3975(96)00086-2. URL https://doi.org/10.1016/S0304-3975(96)00086-2.

Olivier Bournez, Manuel Lameiras Campagnolo, Daniel Silva Graça, and Emmanuel Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *Journal of Complexity*, 23(3):317–335, June 2007. doi: 10.1016/j.jco.2006.12.005. URL http://dx.doi.org/10.1016/j.jco.2006.12.005.

Olivier Bournez, Johanne Cohen, and Adrian Wurm. A universal uniform approximation theorem for neural networks. In Pawel Gawrychowski, Filip Mazowiecki, and Michal Skrzypczak, editors, *50th International Symposium on Mathematical Foundations of Computer Science, MFCS 2025, Warsaw, Poland, August 25-29, 2025*, volume 345 of *LIPIcs*, pages 29:1–29:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. ISBN 978-3-95977-388-1. doi: 10.4230/LIPIcs.MFCS.2025.29. URL https://doi.org/10.4230/LIPIcs.MFCS.2025.29.

Olivier Bournez, Johanne Cohen, Laura Cohen, and Adrian Wurm. Fractal gadgets for neural networks: The complexity of the narrow regime. In *46th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FST/TCS'2026)*, December 2026.

Christopher Brix, Mark Niklas Müller, Stanley Bak, Taylor T. Johnson, and Changliu Liu. First three years of the international verification of neural networks competition (VNN-COMP). *International Journal on Software Tools for Technology Transfer*, 25(3):329–339, 2023. doi: 10.1007/s10009-023-00703-4.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6572–6583, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.

Yifei Duan, Li'ang Li, Guanghua Ji, and Yongqiang Cai. Vanilla feedforward neural networks as a discretization of dynamical systems. *Journal of Scientific Computing*, 101(3):82, 2024. doi: 10.1007/s10915-024-02710-2. URL https://doi.org/10.1007/s10915-024-02710-2.

Vincent Froese, Christoph Hertrich, and Rolf Niedermeier. The computational complexity of relu network training parameterized by data dimensionality. *J. Artif. Intell. Res.*, 74:1775–1790, 2022. doi: 10.1613/jair.1.13547. URL https://doi.org/10.1613/jair.1.13547.

Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. Delta-decidability over the reals. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 305–314. IEEE, IEEE Computer Society, 2012. doi: 10.1109/LICS.2012.41. URL https://doi.org/10.1109/LICS.2012.41.

Oded Goldreich. On promise problems: A survey. In Oded Goldreich, Arnold L. Rosenberg, and Alan L. Selman, editors, *Theoretical Computer Science, Essays in Memory of Shimon Even*, volume 3895 of *Lecture Notes in Computer Science*, pages 254–290. Springer, 2006. doi: 10.1007/11685654_12. URL https://doi.org/10.1007/11685654_12.

Boris Hanin and Mark Sellke. Approximating continuous functions by relu nets of minimal width. *CoRR*, abs/1710.11278, 2017. doi: 10.48550/arXiv.1710.11278. URL http://arxiv.org/abs/1710.11278.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Helge Holden, Kenneth H Karlsen, Knut-Andreas Lie, and Nils Henrik Risebro. Splitting methods for partial differential equations with rough solutions. *European Mathematical Society*, 7:12, 2010.

John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation, Second Edition*. Addison-Wesley, 2nd edition, 2000.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pages 97–117. Springer, 2017.

Patrick Kidger. On neural differential equations. *CoRR*, abs/2202.02435, 2022. doi: 10.48550/arXiv.2202.02435. URL https://arxiv.org/abs/2202.02435.

Patrick Kidger and Terry J. Lyons. Universal approximation with deep narrow networks. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 2306–2327. PMLR, PMLR, 2020. URL http://proceedings.mlr.press/v125/kidger20a.html.

Pascal Koiran, Michel Cosnard, and Max H. Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132(2):113–128, sep 1994. ISSN 0304-3975. doi: 10.1016/0304-3975(94)90229-1. URL https://doi.org/10.1016/0304-3975(94)90229-1.

Angus Macintyre, Alex J Wilkie, and P Odifreddi. On the decidability of the real exponential field. *Kreisel's Mathematics*, 115:451, 1996.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*, 2024. doi: 10.48550/arXiv.2310.07923.

Cristopher Moore. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(3):199–230, 1991. doi: 10.1088/0951-7715/4/2/002.

Christos H Papadimitriou and Mihalis Yannakakis. A note on succinct representations of graphs. *Information and control*, 71(3):181–185, 1986.

Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. *CoRR*, abs/2006.08859, 2020. doi: 10.48550/arXiv.2006.08859. URL `https://arxiv.org/abs/2006.08859`.

Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999. ISBN 978-0-471-98232-6. doi: 10.2307/253980.

Hava T. Siegelmann. *Neural networks and analog computation - beyond the Turing limit*. Progress in theoretical computer science. Birkhäuser, 1999. ISBN 978-0-8176-3949-5.

Hava T. Siegelmann and Eduardo D. Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80, 1991. doi: 10.1016/0893-9659(91)90080-f.

Hava T. Siegelmann and Eduardo D. Sontag. Analog computation via neural networks. *Theoretical Computer Science*, 131(2):331–360, sep 1994. ISSN 0304-3975. doi: 10.1016/0304-3975(94)90178-3. URL `https://doi.org/10.1016/0304-3975(94)90178-3`.

Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997. ISBN 978-0-534-94728-6.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015. doi: 10.48550/arXiv.1505.00387. URL `http://arxiv.org/abs/1505.00387`.

Matus Telgarsky. Benefits of depth in neural networks. In *COLT*, pages 1517–1539, 2016.

Hale F Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 29909–29921, 2021.

Adrian Wurm. Complexity of reachability problems in neural networks. In *International Conference on Reachability Problems*, pages 15–27. Springer, 2023a.

Adrian Wurm. Complexity of reachability problems in neural networks. *CoRR*, abs/2306.05818, 2023b. doi: 10.48550/arXiv.2306.05818. URL `https://doi.org/10.48550/arXiv.2306.05818`.

Adrian Wurm. Robustness verification in neural networks. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 263–278. Springer, 2024.

Shijun Zhang, Jianfeng Lu, and Hongkai Zhao. Deep network approximation: Beyond ReLU to diverse activation functions. *Journal of Machine Learning Research*, 25(35):1–39, 2024.

A Notation and preliminaries

This appendix collects formal definitions used throughout the paper. The neural-network model itself is defined in Section 2.1; here we record additional conventions and the verification problems considered.

Rational networks. Throughout, we restrict attention to *rational* neural networks and inputs: all parameters (weights, biases, activation slopes and hyperplane coefficients of Init and R) are rational numbers. When convenient (here, in Section 5 and in Appendix F) we write R for the acceptance region Acc.

Verification and reachability problems. Fix an activation $\rho\colon \mathbb{R} \to \mathbb{R}$.

Definition 18 (ρ-Verification)**.** The *ρ-Verification* problem is the following decision problem. **Given:** a neural network $N\colon [0,1]^d \to [0,1]^d \in \mathcal{NN}^{\rho}_{d,d,*}$ and two convex regions $\mathrm{Init}, \mathrm{Acc} \subseteq [0,1]^d$ specified by linear inequalities. **Decide:** is there $x \in \mathrm{Init}$ such that $N(x) \in \mathrm{Acc}$?

Definition 19 (ρ-Bounded-Time Recurrent Reachability)**.** The *ρ-Bounded-Time Recurrent Reachability* problem is the decision problem. **Given:** a neural network $N\colon [0,1]^d \to [0,1]^d \in \mathcal{NN}^{\rho}_{d,d,*}$, two convex regions $\mathrm{Init}, \mathrm{Acc} \subseteq [0,1]^d$, and a time bound $T \in \mathbb{N}$ encoded in unary (resp. binary, or unbounded). **Decide:** is there $x \in \mathrm{Init}$ and $t \le T$ with $N^{[t]}(x) \in \mathrm{Acc}$?

The robust variants (δ-robust verification, δ-robust bounded-time reachability) are defined in Definition 9.

B Width-3 simulation of Turing machines (after Bournez et al. [2025])

This appendix recalls the building block of our main theorem: a width-3 ReLU network simulating one step of a Turing machine on a base-4 encoding with digits in $\{1,3\}$. The result is adapted from Bournez et al. [2025].

B.1 Turing machine conventions

We use the following conventions throughout. A *deterministic Turing machine* is a tuple $M = (Q, \Gamma, \delta, q_0, q_F)$, where Q is a finite set of states; $\Gamma = \{1, 3, B\}$ is the tape alphabet, with B denoting the distinguished blank symbol; $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function, where L and R denote a one-cell move of the head to the left and to the right, respectively; $q_0 \in Q$ is the initial state; and $q_F \in Q$ is the unique accepting state. The choice of digits $\{1,3\}$ for the binary input alphabet, instead of $\{0,1\}$, is consistent with the encoding of Section 2.3 and avoids a separate treatment of the digit 0. Thus, a binary input $w \in \{0,1\}^n$ is identified with the tape word $w' \in \{1,3\}^n$ defined by $w'_i = 2w_i + 1$.

A *configuration* of M is a triple (u, q, v) where $q \in Q$ is the current state and $u, v \in \Gamma^{\omega}$ encode the left and right halves of the (bi-infinite) tape, with the head positioned at the leftmost symbol of v. The set of configurations is denoted $\mathcal{C}_M$. The transition function δ induces the single-step map $\mathrm{step}_M\colon \mathcal{C}_M \to \mathcal{C}_M$. A run on input w is the sequence of configurations $c_0, c_1, c_2, \ldots$ where $c_0 = (B^{\omega}, q_0, w \cdot B^{\omega})$ and $c_{t+1} = \mathrm{step}_M(c_t)$. We say M *accepts w in t steps* if c_t has state q_F.

Remark 20. We will sometimes consider M acting on an *infinite binary tape* $\tau \in \{0,1\}^{\omega}$ instead of a finite input; the initial configuration is then $c_0(\tau) = (B^{\omega}, q_0, \tau' \cdot B^{\omega}) \to \tau'$ where $\tau' = (2\tau_1 + 1)(2\tau_2 + 1) \cdots \in \{1,3\}^{\omega}$. Within T steps M visits at most T tape positions, so a finite input w of length $\ge T$ behaves identically to any infinite extension $w \cdot \tau'$.

B.2 Encoding configurations

Let $\mathbb{DD}_{1,3}$ denote the set of base-4 reals with digits in $\{1,3\}$, that is,

$$\mathbb{DD}_{1,3} \;=\; \Big\{ \sum_{i=1}^{\infty} a_i \, 4^{-i} \;:\; a_i \in \{1,3\} \Big\} \;\subseteq\; [0,1].$$

This is a Cantor-like fractal subset of $[0,1]$. The choice of digits in $\{1,3\}$ rather than $\{0,1,2,3\}$ creates uniform gaps between the leading-digit cylinders, which is what allows continuous gadgets to discriminate them. This trick is inspired from Siegelmann and Sontag [1991]: see also discussions and references in Remark 28.

A configuration $C \in \mathcal{C}_M$ of a Turing machine M over alphabet $\Gamma = \{1, 3, B\}$ (with blank B) is a triple (u, q, v) where q is a state and $u, v \in \Gamma^\omega$ are the left and right halves of the tape. Following Bournez et al. [2025], we encode such a configuration into *two* reals by a map

$$\nu \colon \mathcal{C}_M \longrightarrow ([0,1] \cap \overline{\mathbb{DD}_{1,3}})^2, \qquad \nu(C) = \big(\mathrm{enc}_{\{1,3\}}(u),\ \mathrm{enc}_M(q, v)\big),$$

where the second coordinate *co-encodes* state and right tape in base 4: $\mathrm{enc}_M(q, v)$ places a constant-length state prefix $\mathrm{enc}_M(q)$ of length $|q|_4 := \lceil \log_4 |Q| \rceil$ ahead of $\mathrm{enc}_{\{1,3\}}(v)$. This packing keeps the working dimension at 2; the equivalent "naive" encoding (u, v, q) would require 3 dimensions. The width-4 network of Theorem 3 embeds $\nu(C)$ into its first two coordinates, the third being a scratch register and the fourth a validity register (Section 2).

B.3 Auxiliary tool: efficient compilation of one-dimensional CPAMs

We need to write some particular continuous affine maps (CPAM) as particular narrow neural networks.

B.3.1 Some known statements

We review some known statements. We improve some of them, as we need to discuss very small width constructions, and also some effectivity considerations.

Theorem 21 (Arora et al. [2016]). *Every CPAM $f : \mathbb{R}^d \to \mathbb{R}$ can be represented by a $\mathcal{NN}^{\mathrm{ReLU}}_{d,1,*}$.*

Note that this is easily extended to CPAMs with any finite output dimension. However, in practice, the question of the width or depth of the involved ReLU-net that performs a CPAM function is relevant. The article Hanin and Sellke [2017] studies the numbers $w_{\min}(d_{in}, d_{out})$, defined to be the minimal value of width w such that for every continuous function $f : [0,1]^{d_{in}} \to \mathbb{R}^{d_{out}}$ and every $\varepsilon > 0$ there is a ReLU net N with input dimension d_{in}, hidden layer widths at most w, and output dimension d_{out} that ε-approximates f. The main result of Hanin and Sellke [2017] is the following estimate for $w_{\min}(d_{\text{in}}, d_{\text{out}})$.

Theorem 22 ([Hanin and Sellke, 2017, Theorem 1]). *For every $d_{in}, d_{out} \geq 1$, $d_{in} + 1 \leq w_{\min}(d_{in}, d_{out}) \leq d_{in} + d_{out}$.*

The following has been discussed:

Definition 23 (Max-min string, [Hanin and Sellke, 2017, Definition 1]). A function $g : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ is a max-min string of length $L \geq 1$ on d_{in} input variables and d_{out} output variables if there exist affine functions $aff_1, \ldots, aff_L : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ such that

$$g = \sigma_{L-1}(aff_L, \sigma_{L-2}(aff_{L-1}, \ldots, \sigma_2(aff_3, \sigma_1(aff_1, aff_2)) \cdots)$$

where each σ_i is either a coordinate-wise max or a min.

Proposition 24 (Width efficient computation of max-min string, [Hanin and Sellke, 2017, Proposition 2]). *For every max-min string g on d_{in} input variables and d_{out} output variables with length L and every compact $K \subseteq \mathbb{R}^{d_{in}}$, there exists a $\mathcal{NN}^{\mathrm{ReLU}}_{d_{in}, d_{out}, d_{in}+d_{out}}$ and depth L that computes $x \mapsto g(x)$ for every $x \in K$.*

B.3.2 What we actually use

For our purposes, we only need the well-known fact that any one-dimensional continuous piecewise-affine map (CPAM) is expressible as a ReLU expression via the standard hinge representation. The specific gadgets used in the present paper (Lemmas 29 to 31) are constructed explicitly with width-2 implementations that do not appeal to a general width-optimal compilation. They can be seen as (sometimes optimized) particular instances of the following observation.

Lemma 25 (Compilation of bounded CPAMs). *Let $M > 0$ and let $f \colon [0, M] \to \mathbb{R}_{\geq 0}$ be a continuous piecewise affine map with n breakpoints $0 < a_1 < a_2 < \cdots < a_n \leq M$. Then f is computed exactly on $[0, M]$ by a ReLU network of width 2 and depth $2n + 2$ (including one affine output layer).*

Proof. Let $\alpha_0, \dots, \alpha_n$ denote the slopes of f on the subintervals $[0, a_1], [a_1, a_2], \dots, [a_n, M]$, set $\beta_0 := f(0)$, and adopt the convention $a_0 := 0$. The hinge decomposition (see e.g. Arora et al. [2016]) reads

$$f(x) \;=\; \beta_0 + \alpha_0 x \;+\; \sum_{i=1}^{n} (\alpha_i - \alpha_{i-1})\ \mathrm{ReLU}(x - a_i), \qquad x \in [0, M].$$

Define the partial sums $S_j(x) := \beta_0 + \alpha_0 x + \sum_{i=1}^{j} (\alpha_i - \alpha_{i-1}) \,\mathrm{ReLU}(x - a_i)$ for $j \in \{0, \dots, n\}$, so that $S_0(x) = \beta_0 + \alpha_0 x$ and $S_n = f$.

Positivity offset. Each S_j is continuous on the compact $[0, M]$, hence

$$C \;:=\; \max_{0 \le j \le n} \; \max_{x \in [0,M]} \; \max\bigl(0,\, -S_j(x)\bigr)$$

is finite (with explicit bound $C \le |\beta_0| + LM$, where $L := |\alpha_0| + \sum_{i=1}^{n} |\alpha_i - \alpha_{i-1}|$). Set $\widetilde{S}_j(x) := S_j(x) + C \ge 0$ and $u_j(x) := \mathrm{ReLU}(x - a_j)$, so $u_0(x) = x$.

Construction. We build a width-2 network whose state after the j-th *block* of two ReLU layers equals $(u_j(x),\, \widetilde{S}_j(x))$.

Initialization (1 layer). The input $x \in [0, M]$ is lifted by

$$x \;\longmapsto\; \mathrm{ReLU}\begin{pmatrix} x \\ \alpha_0 x + \beta_0 + C \end{pmatrix} \;=\; \begin{pmatrix} u_0(x) \\ \widetilde{S}_0(x) \end{pmatrix},$$

the equality holding because both coordinates are ≥ 0 on $[0, M]$ (the first by hypothesis, the second by choice of C).

Block $j \to j+1$ (2 layers). SUB-LAYER A. Apply

$$\begin{pmatrix} u_j \\ \widetilde{S}_j \end{pmatrix} \;\longmapsto\; \mathrm{ReLU}\begin{pmatrix} u_j - (a_{j+1} - a_j) \\ \widetilde{S}_j \end{pmatrix}.$$

The second coordinate is preserved since $\widetilde{S}_j \ge 0$. The first equals $u_{j+1}(x)$: on $\{x \ge a_j\}$, $u_j(x) = x - a_j$ so $\mathrm{ReLU}(u_j - (a_{j+1} - a_j)) = \mathrm{ReLU}(x - a_{j+1}) = u_{j+1}(x)$; on $\{x < a_j\}$, $u_j(x) = 0$ and $\mathrm{ReLU}(-(a_{j+1} - a_j)) = 0 = u_{j+1}(x)$ (since $x < a_j < a_{j+1}$).

SUB-LAYER B. Apply

$$\begin{pmatrix} u_{j+1} \\ \widetilde{S}_j \end{pmatrix} \;\longmapsto\; \mathrm{ReLU}\begin{pmatrix} u_{j+1} \\ \widetilde{S}_j + (\alpha_{j+1} - \alpha_j)\, u_{j+1} \end{pmatrix} \;=\; \begin{pmatrix} u_{j+1} \\ \widetilde{S}_{j+1} \end{pmatrix}.$$

The first coordinate is preserved since $u_{j+1} \ge 0$. The second argument equals $\widetilde{S}_{j+1}(x)$, which is ≥ 0 by definition of C; hence the ReLU acts as the identity.

Output (1 affine layer). After $2n$ ReLU layers in blocks, the state is $(u_n(x),\, \widetilde{S}_n(x)) = (u_n(x),\, f(x) + C)$. The affine map $(u, v) \mapsto v - C$ yields $f(x)$.

Width and depth. Width is 2 throughout; depth is $1 + 2n + 1 = 2n + 2$. □

B.4 Width-3 step simulation

Theorem 26 (Simulation of TM, [Bournez et al., 2025, Theorem 23])**.** *Let M be a Turing machine with input alphabet $\Sigma = \{1, 3\}$ and tape alphabet $\Gamma = \{1, 3, B\}$, and let $\mathcal{C}_M = \Gamma^\omega \times Q \times \Gamma^\omega$ be its configuration space. There exist a piecewise-affine function $g_M : \mathbb{R}^2 \to \mathbb{R}^2 \in \mathcal{NN}^{\mathrm{ReLU}}_{2,2,3}$ and an encoding map $\nu : \mathcal{C}_M \to ([0,1] \cap \overline{\mathbb{DD}_{1,3}})^2$ such that the diagram below commutes:*

$$\begin{array}{ccc} \mathcal{C}_M & \xrightarrow{\ \vdash\ } & \mathcal{C}_M \\ {\scriptstyle \nu}\downarrow & & \downarrow{\scriptstyle \nu} \\ \mathbb{R}^2 & \xrightarrow{\ g_M\ } & \mathbb{R}^2 \end{array}$$

That is, $g_M(\nu(C)) = \nu(C')$ for every pair of configurations $C \vdash C'$ in $\mathcal{C}_M$.

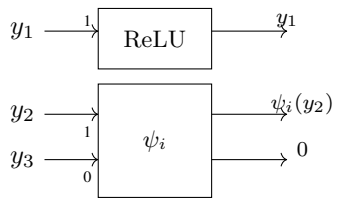

Figure 2: Width-3 block computing $(y_1, y_2, y_3) \mapsto (y_1, \psi_i(y_2), 0)$. Coordinates 2–3 host the width-2 subnetwork of Lemma 25 (staircase + shifted partial sum); coordinate 1 carries y_1 through each layer via $\mathrm{ReLU}(y_1) = y_1$ on $[0, 1]$. A final constant-depth junction outputs $\psi_i(y_2)$ on coordinate 2 and resets coordinate 3 to 0.

Proof. Since the original argument in Bournez et al. [2025] leaves several technical points implicit, we give a self-contained derivation, combining a structural decomposition of the TM step with the one-dimensional compilation of Lemma 25.

Decomposition into one-coordinate atomic updates. The transition $\vdash$ of Bournez et al. [2025] acts on the encoded pair $\nu(C) \in [0, 1]^2$ through a finite sequence of atomic operations: read the leading base-4 digit of the active half-tape (Lemma 30), branch on the state-symbol pair via a constant-depth case split on the finitely many such pairs, write the new symbol, shift the head, update the state prefix. Each atomic operation modifies *exactly one* of the two encoded coordinates $y_1 = \mathrm{enc}_{\{1,3\}}(u)$, $y_2 = \mathrm{enc}_M(q, v)$ and reduces, after the case split, to a continuous piecewise-affine map of one variable with $O_M(1)$ breakpoints (the gadgets of Lemmas 30 and 31 are particular instances). Since the number of atomic operations depends only on M, the construction yields

$$g_M = \phi_K \circ \cdots \circ \phi_1,$$

where K depends on M and each $\phi_i \colon \mathbb{R}^2 \to \mathbb{R}^2$ has the form $(y_1, y_2) \mapsto (y_1,\ \psi_i(y_2))$, or $(y_1, y_2) \mapsto (\psi_i(y_1),\ y_2)$, with ψ_i a 1D CPAM taking values in $[0, 1]$.

One atomic update fits in width 3. Take the first form (the second is symmetric). Lemma 25 provides a width-2 ReLU subnetwork computing ψ_i, internally maintaining a 2-register state (staircase variable and shifted partial sum). We place these two registers in coordinates 2 and 3 of the ambient width-3 network, while coordinate 1 carries y_1 through every ReLU layer via $\mathrm{ReLU}(y_1) = y_1$ (valid since $y_1 \in [0, 1] \subseteq \mathbb{R}_{\geq 0}$). A constant-depth ReLU junction at the end of the block extracts $\psi_i(y_2)$ into coordinate 2, zeros coordinate 3, and preserves y_1 in coordinate 1, restoring the layout $y_1,\ \psi_i(y_2),\ 0)$ required at the entry of ϕ_{i+1}. (See Figure 2 for an illustration).

Concatenation. Stacking the K blocks produces $g_M \in \mathcal{NN}_{2,2,3}^{\mathrm{ReLU}}$ of depth $\sum_{i=1}^{K} O(n_i) = \mathrm{poly}(|M|)$, where n_i is the number of breakpoints of ψ_i. The commutation $g_M \circ \nu = \nu \circ \vdash$ on $\mathcal{C}_M$ holds cylinder-by-cylinder by the explicit case analysis in Bournez et al. [2025]; outside the encoded domain $\nu(\mathcal{C}_M)$, g_M is the continuous piecewise-affine extension determined by the construction, with no claim made. □

Remark 27 (Equivalence with Theorem 12). Theorem 12 in the main body differs from Theorem 26 only in its bookkeeping: the scratch register is exposed as an explicit third input/output coordinate, reset to 0 between TM steps. The conversion from one formulation to the other is purely affine and does not increase the hidden width: prepend an $\mathbb{R}^3 \to \mathbb{R}^3$ map dropping the third input coordinate, and append an $\mathbb{R}^3 \to \mathbb{R}^3$ map padding the output with a zero third coordinate. We use the two-dimensional formulation for the width analysis above and the three-dimensional register form when chaining the simulator with the encoding-validity certificate of Stage 2 in Section 3.

Remark 28. The use of base-4 digits in $\{1, 3\}$ rather than a binary representation is essential: a continuous ReLU map cannot discriminate adjacent base-2 digits, but it can discriminate $\{1, 3\}$-digits because of the gap of size $\frac{1}{2}$ between $\frac{1}{4}$ and $\frac{3}{4}$. See Blondel et al. [2001], Moore [1991], Koiran et al. [1994], Bournez and Cosnard [1996], Siegelmann and Sontag [1991] for related work. Note that the gaps are between the leading-digit cylinders, not between admissible points: $\mathbb{DD}_{1,3}$ is a perfect set, two encodings agreeing on their first m digits differ by at most 4^{-m}, so no fixed margin separates distinct encodings and no rounding step back onto the encoding set exists. This is one more reason

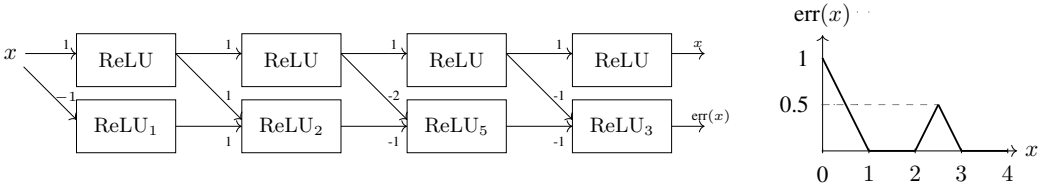

Figure 3: Network N_{err}. Boxes represent ReLU neurons; subscripts indicate biases, edge labels indicate weights, and missing labels correspond to bias 0 and weight 1.

why uniform approximation does not yield exact simulation (the main one is the expansion argument of Section 3.1).

C Encoding-validity gadget at width 4

Theorem 26 requires a validly encoded input from $\mathbb{DD}_{1,3}$. To make the simulation work uniformly on any input in $[0,1]^4$, we construct a width-4 preprocessing gadget that detects deviation from the encoding set. This appendix develops the auxiliary gadgets and combines them.

C.1 Why we need this gadget

If width and dimension were unrestricted, NP-completeness of verification for ReLU networks would follow from a SAT-style reduction [Wurm, 2023a], or from the bounded-time reachability hardness for two-dimensional CPAMs [Bazille et al., 2014]. Translating the CPAM to a ReLU network via [Arora et al., 2016] yields verification hardness for ReLU networks of some fixed width, but the width is not estimated and is at best loose. Moreover, this approach is unrelated to Turing-machine complexity, so it gives only NP-hardness rather than the full time-bound spectrum (Theorem 7).

We instead exploit Theorem 26: ReLU networks of width 3 already simulate Turing machines, but on the fractal encoding $\mathbb{DD}_{1,3}$. Since the verification problem asks for the existence of an input in a *convex* region Init, we must close the gap between "valid encoding inputs" and "arbitrary convex inputs". The encoding-validity gadget below performs this closure with one extra unit of width.

C.2 Auxiliary gadgets

We construct three width-2 subnetworks that constitute the building blocks of the validity check.

Lemma 29 (Error indicator)**.** *There exists a* ReLU *network* $N_{err}\colon \mathbb{R} \to \mathbb{R}^2$ *in* $\mathcal{NN}^{\mathrm{ReLU}}_{1,2,2}$ *such that, for every* $x \in [0,4]$*,* $N_{err}(x) = (x, err(x))$*, where* $err\colon \mathbb{R} \to \mathbb{R}_{\geq 0}$ *satisfies* $0 \leq err(x) \leq 1$ *for all* $x \in \mathbb{R}_{\geq 0}$*, and* $err(x) = 0 \iff x \in [1,2] \cup [3,4]$*.*

Proof. Let $f_{(0,1)}(x) = \mathrm{ReLU}(1-x)$ and $f_{(2,3)}(x) = \frac{1}{2} - \mathrm{ReLU}(x - \frac{5}{2}) - \mathrm{ReLU}(\frac{5}{2} - x)$. Then $f_{(0,1)}$ is strictly positive on $[0,1)$ and non-positive elsewhere; $f_{(2,3)}$ is strictly positive on $(2,3)$ and non-positive elsewhere. Define $\mathrm{err}(x) = \mathrm{ReLU}(f_{(0,1)}(x)) + \mathrm{ReLU}(f_{(2,3)}(x))$. By construction, $\mathrm{err}(x) = 0$ iff $x \in [1,2] \cup [3,4]$. Rewriting as a nested ReLU composition gives

$$\mathrm{err}(x) = \mathrm{ReLU}\Big(3 - x - \mathrm{ReLU}\big(5 - 2x - \mathrm{ReLU}\big(2 - x + \mathrm{ReLU}(1-x)\big)\big)\Big),$$

which is a finite composition of ReLU and affine operations on a single real input, hence directly implementable by a ReLU network of width 2: at each layer one neuron computes the inner ReLU expression while the second neuron carries the running input value x via $\mathrm{ReLU}(x) = x$ (valid since $x \geq 0$ on the domain $[0,4]$). See Figure 3.

□

Lemma 30 (Leading-digit extraction)**.** *There exists a* ReLU *network* $N_{lead}\colon \mathbb{R} \to \mathbb{R}^2$ *in* $\mathcal{NN}^{\mathrm{ReLU}}_{1,2,2}$ *such that, for every* $x = \sum_{i=1}^{\infty} a_i 4^{-i} \in (0,1)$*,* $N_{lead}(x) = (x, lead(x))$*, where* $lead(x) = a_1$ *whenever* $a_1 \in \{1,3\}$*.*

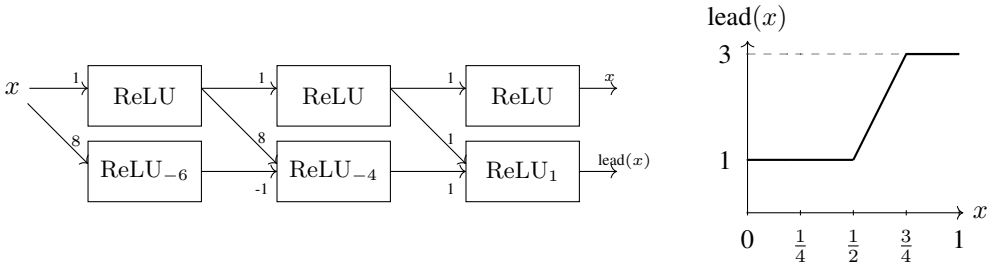

Figure 4: Network N_{lead}. Boxes represent ReLU neurons; subscripts indicate biases, edge labels indicate weights, and missing labels correspond to bias 0 and weight 1.

Proof. The digits $a_1 \in \{1, 3\}$ correspond exactly to $x \in [\frac{1}{4}, \frac{1}{2}] \cup [\frac{3}{4}, 1]$ (with gaps in between). Define

$$\text{lead}(x) = 1 + 8\,\mathrm{ReLU}(x - \tfrac{1}{2}) - 8\,\mathrm{ReLU}(x - \tfrac{3}{4}).$$

A direct case analysis verifies: $x \in [\frac{1}{4}, \frac{1}{2}] \Rightarrow \text{lead}(x) = 1$; $x \in [\frac{3}{4}, 1] \Rightarrow \text{lead}(x) = 3$. See Figure 4. □

Lemma 31 (Base-4 digit shift)**.** *There exists a* ReLU *network* $N_{\textit{shift}} \colon \mathbb{R} \to \mathbb{R}^2$ *in* $\mathcal{NN}^{\mathrm{ReLU}}_{1,2,2}$ *such that, for every* $x \in (0, 1)$, $N_{\textit{shift}}(x) = (x, \textit{shift}(x))$, *where* $\textit{shift}(x) \in [0, 1]$ *for* $x \in (0, 1)$ *and* $\textit{shift}(x) = \sum_{i=2}^{\infty} a_i\, 4^{-(i-1)}$ *when* $x = \sum_{i=1}^{\infty} a_i 4^{-i}$ *with* $a_1 \in \{1, 3\}$.

Proof. It suffices to verify $\text{shift}(x) = 4x - \text{lead}(x)$. □

C.3 Width-4 validity check via the $\{1, 3\}$-base-4 digit gadget

Proposition 32 (Width-4 encoding-validity check)**.** *Fix* $n \in \mathbb{N}$. *There exists a* ReLU *network* $N_{T(n)} \in \mathcal{NN}^{\mathrm{ReLU}}_{2,2,4}$ *such that, for every* $x = \sum_{i=1}^{\infty} a_i 4^{-i} \in [0, 1]$, $N_{T(n)}(x, 0) = (T_n(x), \varepsilon_n(x))$ *satisfies:*

1. *if* $x = \sum_{i=1}^{n} a_i 4^{-i}$ *with* $a_i \in \{1, 3\}$, *then* $T_n(x) = x$ *and* $\varepsilon_n(x) = 0$*;*
2. *conversely, if* $\varepsilon_n(x) = 0$, *then* x *has the form* $\sum_{i=1}^{n} a_i 4^{-i}$ *with* $a_i \in \{1, 3\}$.

Proof. The network $N_{T(n)}$ iteratively decodes the first n base-4 digits of x, accumulating any deviation from $\{1, 3\}$ into an error variable ϵ.

Construction. Given the input pair (x, p), the network performs the following loop:

1. Initialise $\epsilon \leftarrow 0$.
2. For $i = 1, \ldots, n$:
   (a) Update the error: $\epsilon \leftarrow \mathrm{ReLU}(\epsilon + \text{err}(4x))$.
   (b) Extract the leading base-4 digit: $a \leftarrow \text{lead}(x)$.
   (c) Update the accumulator: $p \leftarrow \mathrm{ReLU}(p + a \cdot 4^{-i})$.
   (d) Shift the input: $x \leftarrow \text{shift}(x)$.
3. Return $(p, x + \epsilon)$.

By Lemmas 29, 30, and 31, each instruction can be implemented by a width-2 ReLU subnetwork. The registers p and ϵ must be propagated throughout the computation alongside the working register x, yielding total width 4; see Figure 5.

Correctness. Let (x_i, p_i, ϵ_i) denote the values after i iterations, with initial values $(x_0, p_0, \epsilon_0) = (x, 0, 0)$.

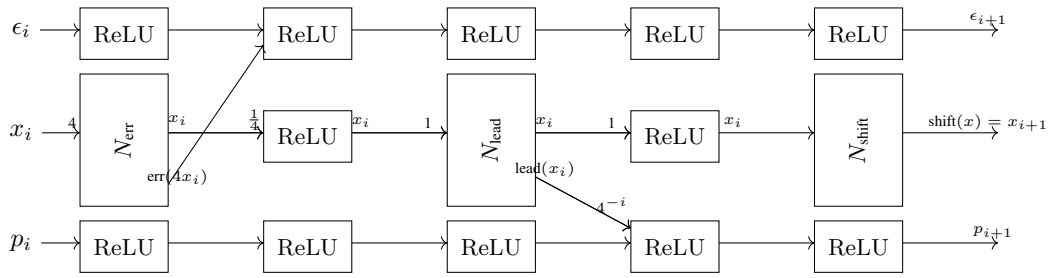

Figure 5: Sub-network of $N_{T(n)}$ implementing the i-th loop iteration, which updates (x_i, p_i, ϵ_i) into $(x_{i+1}, p_{i+1}, \epsilon_{i+1})$ using ReLU neurons and the three components N_{err}, N_{lead}, and N_{shift}. Outputs are shown at the sources of the arrows, and weights at their targets. The sub-network has width 4. The components N_{lead} and N_{shift} have constant depth. For readability, only one ReLU layer is drawn in parallel with these components; formally, one should include as many ReLU layers as the depths of the corresponding sub-networks.

Property 1. Suppose $x = \sum_{j=1}^{n} a_j 4^{-j}$ with $a_j \in \{1, 3\}$. By induction, $x_i = \sum_{j=i+1}^{\infty} a_j 4^{-(j-i)}$ and $p_i = \sum_{j=1}^{i} a_j 4^{-j}$. After n iterations, $x_n = 0$ and $p_n = x$. Moreover, $\epsilon_n = \sum_{i=1}^{n} \text{err}(4x_{i-1}) = 0$ since each $\text{err}(4x_{i-1})$ vanishes when $a_i \in \{1, 3\}$ by Lemma 29. Hence $T_n(x) = p_n = x$ and $\varepsilon_n(x) = x_n + \epsilon_n = 0$.

Property 2. Suppose $N_{T(n)}(x, 0) = (x', 0)$, i.e. $x_n + \epsilon_n = 0$. Both $x_n \geq 0$ (since shift is non-negative) and $\epsilon_n \geq 0$ (sum of ReLUs), so $x_n = 0$ and $\epsilon_n = 0$. We argue by contradiction: suppose x is not of the form $\sum_{j=1}^{n} a_j 4^{-j}$ with $a_j \in \{1, 3\}$. Two cases.

- The first n digits are in $\{1, 3\}$, but $x \neq \sum_{j=1}^{n} a_j 4^{-j}$. Then $x_n = \sum_{j>n} a_j 4^{-(j-n)} > 0$, contradiction (on $\mathbb{DD}_{1,3}$, $x_n \geq 1/3$).
- There is some $\ell \leq n$ with $a_\ell \notin \{1, 3\}$, taken minimal. Then $\text{err}(4x_{\ell-1}) > 0$, hence $\epsilon_n > 0$, contradiction.

This completes the proof. □

Remark 33 (The per-step certificate ε of Theorem 3). The per-step certificate of Theorem 3 is

$$\varepsilon(x_1, x_2) \;=\; \max\big\{\, \text{err}(4x_1),\; \text{err}\big(4\,\text{suf}(x_2)\big),\; d_Q\big(\text{pre}(x_2)\big) \,\big\} \;\in\; [0, 1],$$

where $\text{pre}(x_2)$ and $\text{suf}(x_2)$ are the state-prefix and tape-suffix portions of x_2 (Section 2.3), obtained by a constant-depth slicing (a CPAM, expressible as a finite ReLU expression by Lemma 25), and $d_Q \colon [0, 1] \to [0, 1]$ is a state-distance subgadget that vanishes exactly on $\text{enc}_M(Q)$, built as a $|Q|$-fold minimum of tent functions. The maxima are realized in width 2 via $\max\{a, b\} = a + \text{ReLU}(b - a)$. Thus $\varepsilon(x_1, x_2) = 0$ iff the leading base-4 digit of x_1 and of the tape-suffix of x_2 lies in $\{1, 3\}$ (that is, $4x_1$ and $4\,\text{suf}(x_2)$ lie in the closed cylinders $[1, 2] \cup [3, 4]$, Lemma 29) and the state prefix lies in $\text{enc}_M(Q)$. The zero set of ε is a finite union of boxes, which a finite ReLU network can realize. Membership of the *full* expansions in $\mathbb{DD}_{1,3}$ cannot be checked in one step (Proposition 11): it is checked across iterations, one digit per step, because the simulator's own tape shifts expose successive digits as leading digits. The gadget of Proposition 32 belongs to the finite-word setting of Bournez et al. [2026]: it vanishes exactly when x is a base-4 word of length n over $\{1, 3\}$, and is positive on $\mathbb{DD}_{1,3}$. Only its one-digit building block err (Lemma 29) enters ε.

D Full proof of Theorem 3

This appendix formalizes the construction sketched in Section 3 of the main body, using the building blocks of Appendices B and C. We construct N_M explicitly, establish a structural invariant on the certificate, and derive parts (i) and (ii) of Theorem 3.

Remark 34 (Certifying a set). Part (ii) of Theorem 3 says, informally, that N_M *certifies* a set $V \subseteq [0, 1]^4$: a coordinate is non-decreasing along the iterates and vanishes along the whole trajectory

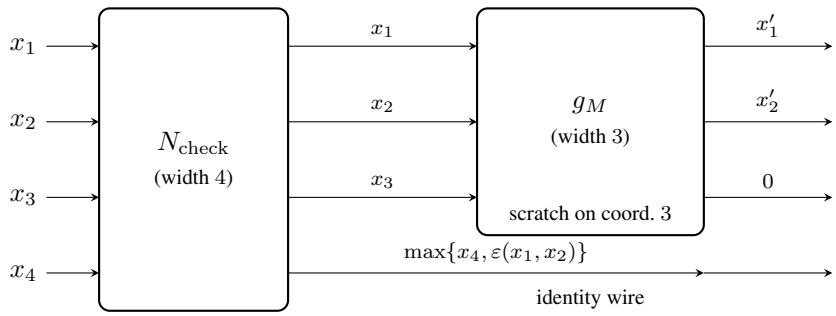

Figure 6: The width-4 network $N_M = N_{\text{sim}} \circ N_{\text{check}}$. The block N_{check} updates the certificate coordinate x_4 to $\max\{x_4, \varepsilon(x_1, x_2)\}$ while leaving (x_1, x_2, x_3) untouched; the block N_{sim} then applies the width-3 simulator g_M to (x_1, x_2) with x_3 as scratch, and propagates the updated x_4 through an identity wire.

of x exactly when $x \in V$. Here V is the set of inputs whose visited digits (Section 2.3) lie in $\{1, 3\}$ and whose state prefix stays in $\text{enc}_M(Q)$. This set contains the encoded configurations, and it is exactly what soundness requires. The quantification over all k is essential, since no finite ReLU network has zero set $\mathbb{DD}_{1,3}$ (Remark 4). The certificate appears only inside the witness network built for hardness (Remark 6).

D.1 Construction of the simulating network N_M for a Turing machine M

Recall the encoding map from Appendix B: a configuration $c = (u, q, v) \in \mathcal{C}_M$ is encoded as $\nu(c) = (\text{enc}_{\{1,3\}}(u), \text{enc}_M(q, v)) \in [0, 1]^2$, where the second coordinate co-encodes state and right tape. The full input to N_M is

$$\text{enc}_M(c) := \big(\nu(c)_1,\ \nu(c)_2,\ 0,\ 0\big) \ \in\ [0, 1]^4,$$

with the third coordinate used as scratch by the width-3 simulator g_M and the fourth as a validity accumulator.

The network $N_M \colon [0, 1]^4 \to [0, 1]^4$ is defined as the composition $N_M = N_{\text{sim}} \circ N_{\text{check}}$ of two register-network blocks; see Figures 1 (schematic) and 6 (formal).

(C) N_{check} *(width-4 certificate update)*:

$$(x_1, x_2, x_3, x_4) \ \longmapsto\ \big(x_1,\ x_2,\ x_3,\ \max\{x_4,\ \varepsilon(x_1, x_2)\}\big),$$

where $\varepsilon \colon [0, 1]^2 \to [0, 1]$ is the *per-step* certificate of Remark 33: $\varepsilon(x_1, x_2) = 0$ iff the leading base-4 digit of x_1 and of the tape-suffix of x_2 lies in $\{1, 3\}$ and the state prefix of x_2 lies in $\text{enc}_M(Q)$. Membership of the full expansions in $\mathbb{DD}_{1,3}$ is checked across iterations, one digit per step (Remark 33). The $\max$ is realized in width 2 via $\max\{a, b\} = a + \text{ReLU}(b - a)$.

(S) N_{sim} *(width-3 simulation)*: applies the width-3 network $g_M \in \mathcal{NN}_{2,2,3}^{\text{ReLU}}$ of Theorem 26 to (x_1, x_2) using x_3 as scratch, while propagating x_4 unchanged through an identity wire (this is consistent with the register-form structure of N_M, see Remark 42).

The order (check *before* simulation) is essential: any encoding violation is detected before g_M has the chance to overwrite (x_1, x_2). The total per-iteration depth is $D_M = O(\text{depth}(g_M)) = \text{poly}(|M|)$, and the width is 4.

D.2 Structural invariant

The key technical content of the proof is the following invariant, which formalizes the claim made informally in Stage 2 of Section 3.

Lemma 35 (Encoding-validity invariant)**.** *Let $x = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, x_4^{(0)}) \in [0, 1]^4$ be any input, and let $(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}) := N_M^{[k]}(x)$ for $k \geq 0$. Call step j* locally valid *if the leading base-4 digit*

of $x_1^{(j)}$ and of the tape-suffix of $x_2^{(j)}$ lies in $\{1,3\}$ and the state-prefix of $x_2^{(j)}$ lies in $\mathrm{enc}_M(Q)$. Then for every $k \geq 0$:

(a) (Monotonicity.) $x_4^{(k+1)} \geq x_4^{(k)}$ and $x_4^{(k)} \in [0,1]$.

(b) (Local validity preserves the certificate.) If every step $j \leq k$ is locally valid, then $x_4^{(k)} = x_4^{(0)}$. In particular, if $x_1^{(0)} \in \mathbb{DD}_{1,3}$, the tape-suffix of $x_2^{(0)}$ is in $\mathbb{DD}_{1,3}$ and the state-prefix of $x_2^{(0)}$ is in $\mathrm{enc}_M(Q)$, then every step is locally valid, since g_M preserves these properties (Theorem 26), and $x_4^{(k)} = x_4^{(0)}$ for all k.

(c) (Violation detected.) Conversely, if some step $j \leq k$ is not locally valid, then $x_4^{(k)} \geq \varepsilon(x_1^{(j)}, x_2^{(j)}) > 0$ for all $k \geq j+1$. In particular, if $x_4^{(0)} = 0$, then $x_4^{(k)} > x_4^{(0)}$, and this persists forever.

Summed over the steps $j < k$, the local-validity conditions are exactly the condition of Theorem 3(ii): every digit visited during the first k iterations (Section 2.3) lies in $\{1,3\}$, and the state prefix stays in $\mathrm{enc}_M(Q)$ along the trajectory.

Proof. (a) The fourth coordinate is updated by $x_4^{(k+1)} = \max\{x_4^{(k)}, \varepsilon(x_1^{(k)}, x_2^{(k)})\}$, which is non-decreasing. Since $\varepsilon \leq 1$ (Remark 33) and $x_4^{(0)} \in [0,1]$, induction gives $x_4^{(k)} \in [0,1]$.

(b) If step j is locally valid, then $\varepsilon(x_1^{(j)}, x_2^{(j)}) = 0$ by Remark 33, hence $x_4^{(j+1)} = \max\{x_4^{(j)}, 0\} = x_4^{(j)}$. Iterating from $j = 0$ to $k-1$ yields $x_4^{(k)} = x_4^{(0)}$. The second claim follows since g_M maps valid encodings to valid encodings (Theorem 26), so every leading digit exposed along the trajectory stays in $\{1,3\}$ and the state prefix stays in $\mathrm{enc}_M(Q)$.

(c) If step j is not locally valid, then $\varepsilon(x_1^{(j)}, x_2^{(j)}) > 0$ (Remark 33), so $x_4^{(j+1)} \geq \varepsilon(x_1^{(j)}, x_2^{(j)}) > 0$. By monotonicity, $x_4^{(k)} \geq x_4^{(j+1)}$ for all $k \geq j+1$. □

D.3 Proof of Theorem 3

Proof of Theorem 3. Part (i): faithful simulation on encoded inputs. Let $\tau \in \{0,1\}^\omega$ be an infinite binary tape and $c_0 := c_0(\tau)$ the initial configuration of M on τ. The input is $\mathrm{enc}_M(c_0) = (\nu(c_0)_1, \nu(c_0)_2, 0, 0)$ with both $\nu(c_0)_1$ and $\nu(c_0)_2$ encoded as required.

We claim by induction on $t \geq 0$ that $N_M^{[C_M\, t]}(\mathrm{enc}_M(c_0)) = \mathrm{enc}_M(c_t(\tau))$, where C_M is the per-step iteration count of g_M in Theorem 26. The base case $t = 0$ is immediate. For the inductive step, suppose $N_M^{[C_M\, t]}(\mathrm{enc}_M(c_0)) = \mathrm{enc}_M(c_t)$ for some t. By Theorem 26, g_M maps $\nu(c_t) \mapsto \nu(c_{t+1})$ in C_M iterations. Since the trajectory remains within encoded configurations (the simulator preserves $\mathbb{DD}_{1,3}$-encoded tapes by Theorem 26), the validity hypothesis of Lemma 35(b) holds at every step, so the fourth coordinate stays at 0. Hence $N_M^{[C_M(t+1)]}(\mathrm{enc}_M(c_0)) = \mathrm{enc}_M(c_{t+1})$.

Part (ii): online certificate properties. The boundedness and monotonicity of $\pi_4(N_M^{[k]})$ are exactly Lemma 35(a). For $x_4^{(0)} = 0$, the condition for $\pi_4(N_M^{[k]}(x)) = 0$ to hold is exactly local validity of every step $j < k$, which by the closing claim of Lemma 35 is the visited-digit condition of the statement. The contrapositive (a violation forces $x_4 > 0$ thereafter) is Lemma 35(c).

Effective construction. The reduction $M \mapsto N_M$ is computable in polynomial time: g_M is constructed in $\mathrm{poly}(|M|)$ time by Theorem 26, the certificate gadget in $\mathrm{poly}(|M|)$ time by Remark 33, and the register-form composition is immediate by construction (Remark 42). Instantiating $M = U$, where U is a fixed universal TM (Lemma 39), yields a single fixed network whose iterates simulate every (M', w). □

D.4 A worked example

There is no encoder anywhere in the construction: in the reachability problem, the input to the network is not produced by encoding anything. It is chosen by the existential quantifier directly inside

the convex region Init, and N_M is applied to whichever point the quantifier picks. Almost every point of Init is not a valid encoding, and being finitely representable is unrelated to being a valid encoding: the rationals in $[0,1]$ are dense, and only a measure-zero subset of them encodes anything. The following two witnesses make this concrete. Fix any M and the regions Init, Acc of Section 4.1: the state prefix of x_2 is pinned to $\mathrm{enc}_M(q_0)$, $x_3 = x_4 = 0$, and x_1 is free in $[0,1]$.

Witness A (a valid encoding). Take $x_1 = 0.\overline{1}_4 = \frac{1}{3}$ (all digits equal to 1, hence $x_1 \in \mathbb{DD}_{1,3}$), and let the tape-suffix digits of x_2 all be 1 as well. Every step is locally valid: $4x_1 = \frac{4}{3} \in [1,2]$, so $\mathrm{err}(4x_1) = 0$ (Lemma 29), and likewise on the suffix of x_2; the same happens at every later step, since g_M maps encodings to encodings (Theorem 26). So $x_4 = 0$ throughout and the iterates are exactly $\mathrm{enc}_M(c_t)$. If M accepts the all-ones tape, the trajectory enters Acc.

Witness B (not an encoding). Take $x_1 = \frac{1}{8} = 0.02_4$, with x_2 as in Witness A. This point is perfectly representable (three bits), lies in Init, and is not in $\mathbb{DD}_{1,3}$: its leading digit is 0. The gadget evaluates $\mathrm{err}(4x_1) = \mathrm{err}(\frac{1}{2})$, and with err as in Lemma 29 the first tent contributes $\mathrm{ReLU}(1 - \frac{1}{2}) = \frac{1}{2}$ while the second contributes 0, so $\varepsilon(x_1, x_2) = \frac{1}{2}$. Hence x_4 jumps to $\frac{1}{2}$ at the first iteration and, being non-decreasing, stays at least $\frac{1}{2}$ forever. Since Acc requires $x_4 \le 0$, this witness can never reach Acc, whatever the continuous extension of g_M does with it downstream.

Witness B is exactly the case the certificate exists for: without it, g_M acting on $\frac{1}{8}$ is an unconstrained continuous piecewise-affine map, and nothing would prevent it from landing in Acc, making the instance answer YES for reasons unrelated to M.

E Verification spectrum at width 4

E.1 Polynomial-bit-complexity witness lemma

The NP-membership claim in the proof of Theorem 7 (unary T, width-4 ReLU) and in the proof of Theorem 10 (unary T, width-6 smooth ρ, after $\rho \to \mathrm{ReLU}$ emulation) rests on the following classical observation.

Lemma 36 (Polynomial-bit-complexity for ReLU network reachability)**.** *Let $N\colon [0,1]^d \to [0,1]^d$ be a* ReLU *network with rational weights of bit complexity b, depth L, and width w. Let* Init, Acc $\subseteq [0,1]^d$ *be convex regions specified by rational linear inequalities of bit complexity $\le b'$, and let $T \in \mathbb{N}$. If there exists a real $x \in$ Init with $N^{[t]}(x) \in$ Acc for some $t \le T$, then there exists a rational $x \in$ Init with the same property and bit complexity* $\mathrm{poly}(d, w, L, T, b, b')$.

Proof. Suppose $x^* \in$ Init is a real point such that $N^{[t^*]}(x^*) \in$ Acc for some $t^* \le T$. We construct a rational witness of polynomial bit complexity.

Step 1: activation pattern. A single forward pass through a ReLU network N of width w and depth L involves at most wL neurons, hence wL ReLU activations. For each $t \in \{1, \dots, T\}$, the composition $N^{[t]}$ involves $wtL \le wTL$ such neurons. Hence, at most wTL ReLU activations in total along the trajectory $x^*, N(x^*), \dots, N^{[T]}(x^*)$. Each neuron is in one of two states: *active* ($\mathrm{ReLU}(z) = z$, i.e. $z \ge 0$) or *inactive* ($\mathrm{ReLU}(z) = 0$, i.e. $z \le 0$). Let $\sigma^* \in \{+,-\}^{wTL}$ denote the activation pattern realized at x^*.

Step 2: polyhedral characterization. Fix any activation pattern $\sigma \in \{+,-\}^{wTL}$. On the set of points $x \in [0,1]^d$ for which iterating N produces the pattern σ, each ReLU acts as either the identity or zero, and so $N^{[t]}$ becomes an *affine* map $A_\sigma^{(t)}\colon \mathbb{R}^d \to \mathbb{R}^d$ for every $t \le T$. The set of x realizing σ is the polyhedron

$$P_\sigma := \big\{ x \in [0,1]^d : \mathrm{sign}(z_{i,t}(x)) = \sigma_{i,t} \text{ for every neuron } i \text{ and iteration } t \le T \big\},$$

where each $z_{i,t}(x)$ is an affine function of x with rational coefficients of bit complexity at most $\mathrm{poly}(d, w, L, T, b)$ (the bit complexity grows by at most $O(b \log w)$ at each layer).

Step 3: linear system. For $\sigma = \sigma^*$ and $t = t^*$, the conjunction "$x \in$ Init, $x \in P_\sigma$, $A_\sigma^{(t)}(x) \in$ Acc" is a system $\mathcal{S}$ of rational linear inequalities. Each inequality has rational coefficients of bit complexity $\mathrm{poly}(d, w, L, T, b, b')$, and the system has at most $\mathrm{poly}(d, w, L, T, b')$ inequalities. The system $\mathcal{S}$ is feasible over $\mathbb{R}$ since x^* satisfies it.

Step 4: rational witness. By a standard result in polyhedral combinatorics [Schrijver, 1999, Ch. 10], any feasible system of rational linear inequalities admits a rational solution of bit complexity polynomial in the bit complexity of the system. Applied to $\mathcal{S}$, this yields a rational $x \in [0,1]^d$ of bit complexity $\mathrm{poly}(d, w, L, T, b, b')$ with $x \in \mathrm{Init}$, $x \in P_{\sigma^*}$, and $N^{[t^*]}(x) = A_{\sigma^*}^{(t^*)}(x) \in \mathrm{Acc}$. □

E.2 Underlying complexity statements about Turing machines

We use the following classical facts about Turing-machine complexity.

Lemma 37 (Basic statements from complexity theory)**.** *Let $\Sigma = \{0,1\}$. The following decision problems are complete for the indicated complexity classes.*

*(i) (*Unary time bound, NP-complete.*) Given a deterministic Turing machine M and $T \in \mathbb{N}$ in unary, does there exist $w \in \Sigma^*$ and $t \le T$ such that M accepts w in t steps?*

*(ii) (*Binary time bound, NEXP-complete.*) Given a deterministic Turing machine M and $T \in \mathbb{N}$ in binary, does there exist $w \in \Sigma^*$ and $t \le T$ such that M accepts w in t steps?*

*(iii) (*No time bound, RE-complete.*) Given a deterministic Turing machine M, does there exist $w \in \Sigma^*$ such that M accepts w (in finite time)?*

Proof. We treat the three items in turn.

(i) NP*-completeness (unary time bound).*

NP**-membership.** Let (M, T) be an instance, with T given in unary. A nondeterministic polynomial-time verifier guesses $w \in \Sigma^*$ of length at most T and a time $t \le T$, then simulates M on input w for exactly t steps. Since T is in unary, the witness (w, t) has polynomial bit length and the simulation runs in polynomial time. The verifier accepts iff M is in an accepting state at step t.

NP**-hardness.** We reduce from SAT. Let φ be a Boolean formula in CNF with n variables. Construct a deterministic Turing machine M_φ that, on input $w \in \{0,1\}^*$, checks whether $|w| = n$ (rejecting otherwise), interprets w as a Boolean assignment, and accepts iff φ is satisfied under this assignment. The machine M_φ runs in time $\mathrm{poly}(|\varphi|)$; let T be a unary encoding of an upper bound on this runtime. Then $\varphi \in \mathrm{SAT}$ iff $\exists w \in \{0,1\}^n$, $\exists t \le T$, M_φ accepts w in t steps. The reduction is polynomial-time.

(ii) NEXP*-completeness (binary time bound).*

NEXP**-membership.** Let (M, T) be an instance with T in binary, of total input size $n = |M| + |T|$. Since M visits at most T tape positions in T steps, we may restrict to $w \in \Sigma^{\le T}$. A nondeterministic verifier guesses such a w in $\le T = 2^{O(n)}$ nondeterministic bits and a time $t \le T$, then simulates M on w for t steps in deterministic time $\mathrm{poly}(|M|, T) = 2^{O(n)}$. Total nondeterministic running time is $2^{O(n)}$, hence in NEXP.

NEXP**-hardness.** We reduce from SUCCINCT-3SAT, the canonical NEXP-complete problem Papadimitriou and Yannakakis [1986], Arora and Barak [2009]. An instance is a Boolean circuit C of size s that, on input a $\lceil 2 \log N \rceil$-bit clause index, outputs three pairs (variable index, sign) encoding a clause of a 3-CNF formula φ over $N = 2^{|s|}$ variables; the question is whether φ is satisfiable. From C, construct a deterministic Turing machine M_C that, on input $w \in \{0,1\}^N$ interpreted as an assignment, iterates over all $\mathrm{poly}(N)$ clauses of φ (each obtained by simulating C, which runs in time $\mathrm{poly}(s) = \mathrm{poly}(\log N)$) and accepts iff every clause is satisfied. Then M_C has size $\mathrm{poly}(s)$ and runs in time $T_0 = \mathrm{poly}(N) \cdot \mathrm{poly}(s)$; let $T = T_0$ in binary, of size $O(\log N) = \mathrm{poly}(s)$ bits. The equivalence $\varphi \in$ SUCCINCT-3SAT $\iff$ $\exists w \in \{0,1\}^N$, $\exists t \le T$, M_C accepts w in t steps yields the reduction.

(iii) RE*-completeness (no time bound).*

RE**-membership.** A semi-decision procedure enumerates pairs $(w, t) \in \Sigma^* \times \mathbb{N}$ (e.g. by $|w| + t$ increasing) and simulates M on w for t steps; it accepts iff an accepting configuration is reached.

RE**-hardness.** We reduce from the halting problem $\mathrm{HALT}_\epsilon = \{M : M \text{ halts on } \epsilon\}$, which is RE-complete Arora and Barak [2009]. Given M, let M' be the machine that ignores its input

and simulates M on ϵ. Then $M \in \mathrm{HALT}_\epsilon$ iff $\exists w \in \Sigma^*$, M' accepts w. The map $M \mapsto M'$ is polynomial-time. □

Remark 38 (Two binary-T variants: EXPTIME vs. NEXP). With T in binary, two natural variants of the decision problem above must be distinguished:

- *Fixed input.* "Given M, a fixed input w, and T in binary, does M accept w in at most T steps?" This is EXPTIME-complete: the deterministic simulation runs in time $T \leq 2^{|T|}$, and hardness follows from the time-hierarchy theorem.
- *Existential input.* As in Lemma 37(ii) above: w is quantified existentially. This is NEXP-complete.

The two problems differ in whether w is part of the input or (fixed) quantified existentially. Our reduction in the proof of Theorem 7 encodes w as the real-valued point $x \in \mathrm{Init}$ chosen existentially inside the network's convex initial region, hence corresponds to the existential variant. This is why we obtain NEXP, not EXPTIME, for binary T.

By passing to a fixed universal Turing machine, the same statements hold with M replaced by an instance of a single fixed U.

Lemma 39 (Fixing the machine)**.** *There exists a fixed deterministic universal Turing machine U such that the three problems of Lemma 37 retain their respective complexity classifications when "M accepts w in t steps" is replaced by "U accepts $\langle M, w\rangle$ in at most $p_U(t, |M|)$ steps" for some fixed polynomial p_U.*

Proof. Fix once and for all a deterministic universal Turing machine U with the standard property (see e.g. Hopcroft and Ullman [2000], Arora and Barak [2009]) that there exists a polynomial p_U such that, for every deterministic Turing machine M, every $w \in \Sigma^*$, and every $t \in \mathbb{N}$,

$$M \text{ accepts } w \text{ in } t \text{ steps} \iff U \text{ accepts } \langle M, w\rangle \text{ in at most } t' \leq p_U(|M|, t) \text{ steps}, \tag{1}$$

where $\langle M, w\rangle \in \Sigma^*$ is a standard binary encoding of the pair, of size $O(|M| + |w|)$.

The map $\Phi\colon (M, T) \mapsto (M, p_U(|M|, T))$ (with $\Phi(M) = M$ in case (iii)) is, by (1), a two-sided reduction between each item of Lemma 37 and its universal-machine variant. It remains to verify that Φ is polynomial-time and that the resulting universal-machine variant falls into the claimed class.

For Lemma 37(i), T is in unary, so $T' = p_U(|M|, T)$ is computable and written in unary in polynomial time. For Lemma 37(ii), T is in binary, so $|T'| = O(\log p_U(|M|, T)) = \mathrm{poly}(|M|, |T|)$ and Φ is polynomial-time. For Lemma 37(iii), Φ is the identity on M.

In each case, membership in the appropriate class (NP/NEXP/RE) follows by viewing the universal-machine variant as a special case of Lemma 37 applied to the single fixed machine U on inputs of the form $\langle M, w\rangle$: the new time bound is $T' = p_U(T, |M|)$, which has the same encoding type as T (unary/binary/none, depending on the item); the size of $\langle M, w\rangle$ is polynomial in $|M|$ and $|w|$. Hardness follows from Equation (1) together with the hardness part of the corresponding item of Lemma 37.

□

E.3 Proof of Theorem 7

We now assemble the ingredients to give a formal proof of Theorem 7, restating the result for convenience.

Theorem (Theorem 7, restated)**.** The bounded-time recurrent reachability problem (Definition 19) for width-4 ReLU networks is:

(i) NP-complete when T is encoded in unary;

(ii) NEXP-complete when T is encoded in binary;

(iii) RE-complete (undecidable) when T is unbounded.

Reduction recipe (uniform across (i)–(iii)). For each item, we reduce a Turing-machine acceptance problem to width-4 recurrent reachability. The reduction is identical across items; only the source problem and the encoding of T vary.

Let M be a deterministic Turing machine and $T \in \mathbb{N}$ a time bound (unary, binary, or absent). By Theorem 3, applied to M, we obtain a width-4 ReLU network $N_M \colon [0,1]^4 \to [0,1]^4$ of depth $D_M = \mathrm{poly}(|M|)$, constructible in polynomial time, with the simulation property (part (i)) and the online certificate property (part (ii)). Recall from Appendix B that the second coordinate x_2 co-encodes the state and right tape: a state-prefix $\mathrm{enc}_M(q) \in \{0, 1, \ldots, 4^{|q|_4} - 1\} \cdot 4^{-|q|_4}$ of length $|q|_4 = \lceil \log_4 |Q| \rceil$ is followed by a $\mathbb{DD}_{1,3}$-encoded right tape.

Define the convex regions

$$\mathrm{Init} := \big\{ x \in [0,1]^4 \ : \ \mathrm{enc}_M(q_0) \le 4^{|q|_4}\, x_2 \le \mathrm{enc}_M(q_0) + 1,\ x_3 = 0,\ x_4 = 0 \big\},$$
$$\mathrm{Acc} := \big\{ x \in [0,1]^4 \ : \ \mathrm{enc}_M(q_F) \le 4^{|q|_4}\, x_2 \le \mathrm{enc}_M(q_F) + 1,\ x_4 \le 0 \big\}.$$

Both regions are specified by axis-aligned linear inequalities with rational coefficients of bit complexity $O(|q|_4) = O(\log |M|)$, hence convex and polynomial-time constructible from M.

The reduction maps the Turing-machine instance (M, T) (or just M for case (iii)) to the reachability instance $(N_M, \mathrm{Init}, \mathrm{Acc}, T \cdot C_M)$, where C_M is the per-step iteration count of Theorem 3.

Correctness of the recipe. We claim:

$$\begin{aligned} &\text{There exist } x \in \mathrm{Init} \text{ and } t \le T \cdot C_M \text{ with } N_M^{[t]}(x) \in \mathrm{Acc} \\ &\quad \Longleftrightarrow \quad \exists\, \tau \in \{0,1\}^\omega,\ \exists\, t' \le T,\ M \text{ accepts } \tau \text{ in } t' \text{ steps.} \end{aligned} \tag{2}$$

(⇒) Suppose $x \in \mathrm{Init}$ and $N_M^{[t]}(x) \in \mathrm{Acc}$ for some $t \le T \cdot C_M$. We have $x_4(0) = 0$ and $\pi_4(N_M^{[t]}(x)) \le 0$ (since $N_M^{[t]}(x) \in \mathrm{Acc}$). By monotonicity of x_4 along the trajectory (Lemma 35(a)), this forces $\pi_4(N_M^{[k]}(x)) = 0$ for every $k \le t$. By items (b)–(c) of Lemma 35, every step $k \le t$ is locally valid: the leading digit of $x_1^{(k)}$ and of the tape-suffix of $x_2^{(k)}$ lies in $\{1, 3\}$, and the state-prefix of $x_2^{(k)}$ lies in $\mathrm{enc}_M(Q)$. Since N_M acts by affine maps that are uniform on leading-digit cylinders, the state prefixes of its first t iterates depend only on the digits visited during the first t iterations. Let x' be obtained from x by setting every unvisited digit of x_1 and of the tape-suffix of x_2 to 1. Then x' encodes a configuration $c_0(\tau)$ with $\tau \in \{0,1\}^\omega$, and $N_M^{[k]}(x)$ and $N_M^{[k]}(x')$ have the same state prefix for every $k \le t$. By Theorem 3(i), N_M faithfully simulates M on the infinite binary tape τ encoded by x', and its trajectory reaches a state-encoded q_F at iteration $\le t$. Hence M accepts τ in some $t' \le t/C_M \le T$ steps.

(⇐) Conversely, suppose M accepts some $\tau \in \{0,1\}^\omega$ in $t' \le T$ steps. Set $x = \mathrm{enc}_M(c_0(\tau)) \in \mathrm{Init}$. By Theorem 3(i), $N_M^{[C_M\, t']}(x) = \mathrm{enc}_M(c_{t'}(\tau))$, whose state-prefix is $\mathrm{enc}_M(q_F)$ and whose fourth coordinate is 0 (by Lemma 35(b), since the trajectory remains within encoded configurations). Hence $N_M^{[C_M\, t']}(x) \in \mathrm{Acc}$ with $C_M\, t' \le T \cdot C_M$.

Polynomial-time computability. The reduction $(M, T) \mapsto (N_M, \mathrm{Init}, \mathrm{Acc}, T \cdot C_M)$ is polynomial-time: N_M is constructed in $\mathrm{poly}(|M|)$ by Theorem 3; Init and Acc are described by $O(|q|_4) = O(\log |M|)$ rational coefficients; and the new time bound $T \cdot C_M$ has size $|T| + \mathrm{poly}(|M|)$, which preserves the encoding (unary stays unary, binary stays binary).

Hardness of (i)–(iii). Apply the recipe with the source problems of Lemma 37:

- (i) Source: Lemma 37(i), NP-complete with unary T. The recipe yields a polynomial-time reduction to width-4 reachability with unary time bound $T \cdot C_M$.
- (ii) Source: Lemma 37(ii), NEXP-complete with binary T. The recipe yields a polynomial-time reduction preserving the binary encoding of the time bound.
- (iii) Source: Lemma 37(iii), RE-complete with no time bound. The recipe yields a polynomial-time reduction to unbounded reachability.

Membership in (i): NP**.** Given (M, T) with T in unary, build $(N_M, \mathrm{Init}, \mathrm{Acc}, T \cdot C_M)$ in polynomial time. By Lemma 36 applied with parameters $d = 4$, $w = 4$, $L = D_M = \mathrm{poly}(|M|)$, time bound $T \cdot C_M = \mathrm{poly}(|M|, T)$, weight bit complexity $b = \mathrm{poly}(|M|)$, and region bit complexity $b' = O(\log |M|)$, any reachability witness can be taken rational of bit complexity $\mathrm{poly}(|M|, T)$. A nondeterministic verifier guesses such a rational $x \in \mathrm{Init}$, simulates N_M exactly for $T \cdot C_M$ iterations, and checks at each step whether the trajectory enters Acc. Total time is $\mathrm{poly}(|M|, T)$, hence NP-membership.

Membership in (ii): NEXP**.** Given (M, T) with T in binary, build $(N_M, \mathrm{Init}, \mathrm{Acc}, T \cdot C_M)$ as before. Apply Lemma 36 with the same parameters except $T \cdot C_M$ now binary, of bit length $|T| + O(\log |M|)$. The lemma yields a rational witness of bit complexity $\mathrm{poly}(|M|, T)$, which is exponential in the input size $|M| + |T|$. A nondeterministic verifier guesses such a witness in $\mathrm{poly}(|M|, T) = 2^{\mathrm{poly}(|M|, |T|)}$ nondeterministic time, then simulates N_M for $T \cdot C_M = 2^{O(|T|)} \cdot \mathrm{poly}(|M|)$ steps in deterministic exponential time. Total nondeterministic running time is $2^{\mathrm{poly}(|M|, |T|)}$, hence NEXP-membership.

Membership in (iii): RE**.** A semi-decision procedure simulates N_M from each rational point of Init enumerated in some computable order, for increasing numbers of iterations, accepting if the trajectory ever enters Acc. By Lemma 36, if the instance is positive then some rational $x \in \mathrm{Init}$ of finite bit complexity reaches Acc in finite time, so the procedure halts and accepts on positive instances. Hence the problem is in RE.

Combining hardness and membership in each case proves Theorem 7.

F Robust hardness transfer from ReLU to smooth activations

This appendix proves Theorem 10 by transferring NP-hardness from ReLU networks at width 4 to ρ-networks at width 6. We work with the *register* form of ReLU networks, in which each hidden layer updates a single coordinate.

The key technical ingredient, used to construct identity-preserving gadgets and approximate ReLU from ρ, is the local-identity approximation of Kidger and Lyons [2020, Lemma 4.1].

Proposition 40 (Local approximation of the identity, Kidger and Lyons, 2020, Lemma 4.1)**.** *Let $\rho \colon \mathbb{R} \to \mathbb{R}$ be continuous and continuously differentiable at at least one point with nonzero derivative there. Let $L \subseteq \mathbb{R}$ be compact. Then a single enhanced neuron with activation ρ uniformly approximates the identity function $\mathrm{Id} \colon \mathbb{R} \to \mathbb{R}$ on L, with arbitrarily small error.*

Proof sketch (after Kidger and Lyons, 2020). By hypothesis there exist $\alpha \in \mathbb{R}$ where ρ is continuously differentiable with $\rho'(\alpha) \neq 0$. For $h \in \mathbb{R} \setminus \{0\}$ set $\phi_h(x) = hx + \alpha$ and $\psi_h(x) = (x - \rho(\alpha))/(h\rho'(\alpha))$. Then $\iota_h = \psi_h \circ \rho \circ \phi_h$ is realized by a single enhanced neuron, and $\iota_h(x) \to x$ uniformly on any compact L as $h \to 0$, by the mean-value theorem applied to ρ on $[\alpha, \alpha + hx]$. $\square$

F.1 Register networks

Definition 41 (ReLU-register network, restated from Definition 15)**.** A ReLU-register network $F \colon \mathbb{R}^d \to \mathbb{R}^d$ of register width k and depth L is a composition

$$F = A_{\mathrm{out}} \circ R_L \circ \cdots \circ R_1 \circ A_{\mathrm{in}} \colon \mathbb{R}^d \to \mathbb{R}^d,$$

where $A_{\mathrm{in}} \colon \mathbb{R}^d \to \mathbb{R}^k$ and $A_{\mathrm{out}} \colon \mathbb{R}^k \to \mathbb{R}^d$ are affine, and each register-layer $R_\ell \colon \mathbb{R}^k \to \mathbb{R}^k$ is of the form: there exist an index $j_\ell \in \{1, \ldots, k\}$ and parameters $a_\ell \in \mathbb{R}^k$, $b_\ell \in \mathbb{R}$ such that $(R_\ell(x))_i = x_i$ for $i \neq j_\ell$ and $(R_\ell(x))_{j_\ell} = \mathrm{ReLU}(a_\ell \cdot x + b_\ell)$.

Remark 42 (Native register form of N_M). The network N_M of Theorem 3 is, by its construction, in register form of register width 4:

- the width-3 simulator g_M of Theorem 26 is in register form by the construction of Bournez et al. [2025];
- the certificate update $x_4 \leftarrow \max\{x_4, \varepsilon(x_1, x_2)\}$ modifies a single coordinate, computed in width 2 via $\max\{a, b\} = a + \mathrm{ReLU}(b - a)$;

- the unmodified coordinates are propagated unchanged.

In particular, N_M can be passed directly to the ρ-emulation of Lemma 17 below without any preliminary register-form conversion.

F.2 Approximation of ReLU register networks by smooth networks

We approximate ReLU-register networks by ρ-networks in two steps: first one register-layer at a time (Lemma 43), then the entire network by telescoping (Lemma 17).

This is precisely Lemma 16 of the main body, which we restate and prove here.

Lemma 43 (Approximating one ReLU register-layer at width $k+2$)**.** *Let $\rho\colon \mathbb{R} \to \mathbb{R}$ be continuous, non-polynomial, and C^1 at some point with non-zero derivative. Fix $k \geq 1$, a compact set $K \subseteq \mathbb{R}^k$, an index $j \in \{1, \dots, k\}$, and parameters $a \in \mathbb{R}^k$, $b \in \mathbb{R}$. Consider the ReLU register-layer $F\colon \mathbb{R}^k \to \mathbb{R}^k$ with $F_i(x) = x_i$ for $i \neq j$ and $F_j(x) = \mathrm{ReLU}(a \cdot x + b)$. Then for every $\varepsilon > 0$ there exists a ρ-network $\widetilde{F} \in \mathcal{NN}^{\rho}_{k,k,k+2}$, constructible in time $\mathrm{poly}(k, \log(1/\varepsilon), \mathrm{size}(a, b))$, such that*

$$\sup_{x \in K} \|\widetilde{F}(x) - F(x)\|_\infty \leq \varepsilon.$$

Proof. The strategy is to dedicate k of the $k+2$ neurons to approximating the identity on the k register coordinates, and the remaining 2 neurons to approximating the ReLU operation $z \mapsto \mathrm{ReLU}(a \cdot x + b)$. Coordinates are then assembled via the output affine map.

Step 1: ingredients. By hypothesis on ρ and Proposition 40, for every compact $L \subseteq \mathbb{R}$ and every $\eta > 0$ there exists a single enhanced ρ-neuron computing a function $\iota_\eta\colon \mathbb{R} \to \mathbb{R}$ with $\sup_{z \in L} |\iota_\eta(z) - z| \leq \eta$.

By the effective ReLU emulation result of Zhang et al. [2024] (see also Kidger and Lyons, 2020, Theorem 4.3 for existence), for activations satisfying (H1)–(H3) and every compact $L' \subseteq \mathbb{R}$ and every $\eta > 0$, there exists a ρ-network $\mathcal{R}_\eta\colon \mathbb{R} \to \mathbb{R}$ of width 2 and constant depth, constructible in time $\mathrm{poly}(\log(1/\eta), \log B)$ where $B = \sup_{z \in L'} |z|$, such that $\sup_{z \in L'} |\mathcal{R}_\eta(z) - \mathrm{ReLU}(z)| \leq \eta$.

Step 2: compact sets used. Let $C_a := \sup_{x \in K} |a \cdot x + b|$, finite by compactness of K, and set $L' := [-C_a, C_a]$. Let $L \subseteq \mathbb{R}$ be a compact set containing every coordinate value x_i for $x \in K$, $i \in \{1, \dots, k\}$.

Step 3: construction of $\widetilde{F}$. Set $\eta := \varepsilon$ and define $\widetilde{F} \in \mathcal{NN}^{\rho}_{k,k,k+2}$ as follows. The hidden layer has $k+2$ neurons:

- Neurons $1, \dots, k$: each neuron i takes input x_i via the input affine map and applies the identity-emulating gadget ι_η on L. By Step 1, neuron i outputs $\iota_\eta(x_i)$ with $|\iota_\eta(x_i) - x_i| \leq \eta$ for $x \in K$.
- Neurons $k+1$ and $k+2$: jointly compute the ReLU gadget $\mathcal{R}_\eta$ applied to the pre-activation $a \cdot x + b$, formed via the input affine map. By Step 1, the combined output approximates $\mathrm{ReLU}(a \cdot x + b)$ within η on L'.

The output affine map assembles the k output coordinates: for $i \neq j$, set $\widetilde{F}_i(x) := \iota_\eta(x_i)$; for $i = j$, set $\widetilde{F}_j(x) := \mathcal{R}_\eta(a \cdot x + b)$.

Step 4: error bound. For every $x \in K$ and every coordinate i:

- if $i \neq j$: $|\widetilde{F}_i(x) - F_i(x)| = |\iota_\eta(x_i) - x_i| \leq \eta = \varepsilon$;
- if $i = j$: $|\widetilde{F}_j(x) - F_j(x)| = |\mathcal{R}_\eta(a \cdot x + b) - \mathrm{ReLU}(a \cdot x + b)| \leq \eta = \varepsilon$.

Hence $\|\widetilde{F}(x) - F(x)\|_\infty \leq \varepsilon$ uniformly on K.

Step 5: constructibility. Both ι_η and $\mathcal{R}_\eta$ are constructed by the proofs of Proposition 40 and Theorem 4.3 of Kidger and Lyons [2020] in time $\mathrm{poly}(\log(1/\eta), \mathrm{size}(a, b))$. □

We now lift the per-layer approximation to a network-level statement. This is precisely Lemma 17 of the main body, which we restate and prove here.

Lemma 44 (Lemma 17, restated)**.** *Let $\rho\colon \mathbb{R} \to \mathbb{R}$ be continuous, non-polynomial, and C^1 at some point with non-zero derivative. Let $F\colon \mathbb{R}^d \to \mathbb{R}^d$ be a* ReLU*-register network of register width k and depth L, let $K \subseteq \mathbb{R}^d$ be compact, and let $\varepsilon > 0$. Then there exists a ρ-network $\widetilde{F} \in \mathcal{NN}^{\rho}_{d,d,k+2}$ of depth $O(L)$, constructible in time $\mathrm{poly}(|F|, \log(1/\varepsilon))$, such that*

$$\sup_{x \in K} \|\widetilde{F}(x) - F(x)\|_\infty \le \varepsilon.$$

Proof. Write $F = A_{\mathrm{out}} \circ R_L \circ \cdots \circ R_1 \circ A_{\mathrm{in}}$. Let $L_\ell := \max\{1, \|a_\ell\|_1\}$ be the Lipschitz constant of R_ℓ in $\|\cdot\|_\infty$ (since ReLU is 1-Lipschitz and $|a_\ell \cdot u| \le \|a_\ell\|_1 \|u\|_\infty$); let L_{out} be the Lipschitz constant of A_{out}.

Step 1: compact propagation with buffer. Set $K_0 := A_{\mathrm{in}}(K)$ and inductively $K_\ell := R_\ell(K_{\ell-1})$, all compact in $\mathbb{R}^k$. Let $\eta := \varepsilon / L_{\mathrm{out}}$ and define the enlarged compacts $K_\ell^+ := K_\ell + \overline{B_\eta}$ (closed η-neighborhood in $\|\cdot\|_\infty$).

Step 2: per-layer approximation. Apply Lemma 43 on $K_{\ell-1}^+$ with target precision

$$\delta_\ell := \frac{\varepsilon}{L_{\mathrm{out}} \cdot (L+1) \cdot \prod_{t=\ell+1}^{L} L_t},$$

producing a ρ-block $\widetilde{R}_\ell \in \mathcal{NN}^{\rho}_{k,k,k+2}$ such that

$$\sup_{u \in K_{\ell-1}^+} \|\widetilde{R}_\ell(u) - R_\ell(u)\|_\infty \;\le\; \delta_\ell.$$

The block is constructible in time $\mathrm{poly}(k, \log(1/\delta_\ell), \mathrm{size}(a_\ell, b_\ell)) = \mathrm{poly}(|F|, \log(1/\varepsilon))$.

Step 3: composed network. Define $\widetilde{F} := A_{\mathrm{out}} \circ \widetilde{R}_L \circ \cdots \circ \widetilde{R}_1 \circ A_{\mathrm{in}}$. Each $\widetilde{R}_\ell$ has width $k+2$ and constant depth, so $\widetilde{F}$ has width $k+2$ and depth $O(L)$.

Step 4: error telescoping. For $x \in K$, let $\xi_\ell(x) := R_\ell \circ \cdots \circ R_1 \circ A_{\mathrm{in}}(x) \in K_\ell$ and $\widetilde{\xi}_\ell(x) := \widetilde{R}_\ell \circ \cdots \circ \widetilde{R}_1 \circ A_{\mathrm{in}}(x)$, with $E_\ell(x) := \|\widetilde{\xi}_\ell(x) - \xi_\ell(x)\|_\infty$ and $E_0 \equiv 0$.

We claim by induction that $\widetilde{\xi}_\ell(x) \in K_\ell^+$ and $E_\ell(x) \le \sum_{j=1}^{\ell} \delta_j \prod_{t=j+1}^{\ell} L_t$ for every $x \in K$ and $\ell \le L$. The base case $\ell = 0$ is immediate. Assume both claims for $\ell - 1$. Then $\widetilde{\xi}_{\ell-1}(x) \in K_{\ell-1}^+$, so the approximation guarantee for $\widetilde{R}_\ell$ applies at this point, giving

$$\begin{aligned} E_\ell(x) &\le \|\widetilde{R}_\ell(\widetilde{\xi}_{\ell-1}) - R_\ell(\widetilde{\xi}_{\ell-1})\|_\infty + \|R_\ell(\widetilde{\xi}_{\ell-1}) - R_\ell(\xi_{\ell-1})\|_\infty \\ &\le \delta_\ell + L_\ell \cdot E_{\ell-1}(x). \end{aligned}$$

Solving the recurrence yields $E_\ell(x) \le \sum_{j=1}^{\ell} \delta_j \prod_{t=j+1}^{\ell} L_t$. By the choice of δ_j, each summand satisfies

$$\delta_j \prod_{t=j+1}^{\ell} L_t = \frac{\varepsilon}{L_{\mathrm{out}}\,(L+1)} \cdot \frac{\prod_{t=j+1}^{\ell} L_t}{\prod_{t=j+1}^{L} L_t} \le \frac{\varepsilon}{L_{\mathrm{out}}\,(L+1)},$$

so $E_\ell(x) \le L\varepsilon / (L_{\mathrm{out}}(L+1)) \le \varepsilon / L_{\mathrm{out}} = \eta$ uniformly. This shows in particular $\widetilde{\xi}_\ell(x) \in K_\ell^+$, validating the induction.

Step 5: final bound. Apply A_{out}:

$$\|\widetilde{F}(x) - F(x)\|_\infty = \|A_{\mathrm{out}}(\widetilde{\xi}_L) - A_{\mathrm{out}}(\xi_L)\|_\infty \le L_{\mathrm{out}} \cdot E_L(x) < \varepsilon$$

uniformly on K.

Constructibility. The constants L_ℓ and L_{out} are computable from the affine parameters of F in linear time. Each δ_ℓ has $\log(1/\delta_\ell) = \mathrm{poly}(|F|, \log(1/\varepsilon))$ bits. By Step 2, each $\widetilde{R}_\ell$ is constructible in time $\mathrm{poly}(|F|, \log(1/\varepsilon))$, and the composition is linear in L. □

F.3 Robust hardness transfer

Theorem 45 (Robust hardness transfer $\mathrm{ReLU} \to \rho$)**.** *Fix $d \geq 1$ and $k \geq d$, and let ρ satisfy hypotheses (H1)–(H3) of Theorem 10. If δ-robust verification (resp. δ-robust bounded-time reachability) is* NP*-hard for width-k* ReLU*-register networks in dimension d, then the corresponding problem is* NP*-hard for width-$(k+2)$ ρ-networks.*

Proof. We give a polynomial-time many-one promise-preserving reduction. Given an instance $(N, \mathrm{Init}, R, \delta)$ of δ-verification for a ReLU-register network N of width k, apply Lemma 44 on the compact $K := \mathrm{Init}$ with target precision $\varepsilon := \delta/4$ to obtain a width-$(k+2)$ ρ-network $\widetilde{N}$ satisfying $\sup_{x \in K} \|\widetilde{N}(x) - N(x)\|_\infty \leq \delta/4$. Output the instance $(\widetilde{N}, \mathrm{Init}, R^{+\delta}, \delta/4)$, where $R^{+\delta}$ denotes the closed δ-neighborhood of R.

YES preservation. If there exists $x \in \mathrm{Init}$ with $N(x) \in R^{+\delta}$, then $dist(\widetilde{N}(x), R) \leq dist(N(x), R) + \delta/4 \leq \delta + \delta/4$, so $\widetilde{N}(x) \in R^{+(\delta+\delta/4)} \subseteq (R^{+\delta})^{+\delta/4}$: the output is YES at the new promise gap $\delta/4$.

NO preservation. If $dist(N(x), R) > 2\delta$ for all $x \in \mathrm{Init}$, then $dist(N(x), R^{+\delta}) > \delta$, hence $dist(\widetilde{N}(x), R^{+\delta}) \geq dist(N(x), R^{+\delta}) - \delta/4 > 3\delta/4 = 2 \cdot (\delta/4) + \delta/4 > 2 \cdot (\delta/4)$, so the output is NO.

For bounded-time reachability with unary T, apply the same reduction with $K = [0,1]^d$ and $\varepsilon := \delta/(4 \cdot S_T)$ where $S_T := \sum_{i=0}^{T-1} L^i$ and L is a Lipschitz constant of N on K; the standard error-recurrence telescoping (cf. Step 4 of Lemma 44) controls the iterated error within $\delta/4$ across T iterations.

The reduction is polynomial-time by the constructibility bound of Lemma 44. □

Theorem 46 (NP-membership for robust verification of smooth networks)**.** *Fix $d \geq 1$, and let ρ satisfy hypotheses (H1)–(H3) of Theorem 10. Then δ-robust verification (resp. robust bounded-time reachability with unary T) for ρ-networks of dimension d belongs to* NP*.*

Proof. A nondeterministic verifier guesses a rational $x \in \mathrm{Init}$ with polynomially many bits (justified below) and evaluates $\widetilde{y} := N(x)$ to precision $\delta/4$ using polynomial-time evaluation of ρ (by (H3)) and exact rational arithmetic for the affine layers.

Witness bit-complexity. By hypothesis (H3), ρ admits a polynomial-time-computable Lipschitz constant L_ρ on every bounded interval. Combined with the Lipschitz constants of the affine layers, N is Lipschitz with constant $L_N = \mathrm{poly}(\text{input})$ on $K = \mathrm{Init}$. By Lemma 43 applied at precision $\delta/4$, there is a ReLU-equivalent N^{ReLU} with $\sup_{x \in K} \|N(x) - N^{\mathrm{ReLU}}(x)\|_\infty \leq \delta/4$. Apply Lemma 36 to N^{ReLU}: any feasible real witness $x \in \mathrm{Init}$ for the ReLU instance can be replaced by a rational witness of bit complexity $\mathrm{poly}(\text{input})$. The promise gap of δ vs. 2δ tolerates the cumulative error $2 \cdot \delta/4 = \delta/2$, ensuring that the rational witness preserves the YES/NO classification.

Verification. The verifier evaluates $\widetilde{y}$ to precision $\delta/4$. Under the δ vs. 2δ promise, $dist(\widetilde{y}, R) \leq \delta + \delta/4$ in the YES case and $dist(\widetilde{y}, R) \geq 2\delta - \delta/4 > \delta + \delta/4$ in the NO case, so the approximate output unambiguously certifies the correct case. □

Conclusion. Combining Theorem 45 with NP-hardness of robust ReLU verification at width 4 (the unary-T case of Theorem 7, equivalently Corollary 14), and Theorem 46 for membership, yields Theorem 10.

On binary T for ρ. The argument above relies on T unary so that the cumulative approximation error $T \cdot L_\rho^T \cdot \varepsilon_{\mathrm{step}}$ remains controllable in polynomial time. With T binary, the same calculation requires $\varepsilon_{\mathrm{step}} \leq \delta/(4 \cdot 2^{|T|} \cdot L_\rho^{2^{|T|}})$, which has $\Omega(L_\rho^{2^{|T|}})$ bits of precision, doubly exponential in the input. Whether δ-robust bounded-time reachability of ρ-networks at width 6 admits a NEXP-style hardness statement at binary T remains open.

Approximation hardness for ρ-networks. The approximation-hardness of Corollary 14 extends to ρ-networks at width 6 via the same reduction: under (H1)-(H3) there exists $\eta > 0$ such that approximating the worst-case output of a width-6 ρ-network within additive error η is NP-hard (unary T). The argument is identical to the proof above, using that the $\Omega(1)$ promise gap of Corollary 14 is preserved up to a constant by the ρ-emulation. Combined with Corollary 14, this yields a unified intractability picture for ReLU and all common smooth activations at constant width.

F.4 Proof of Corollary 8

Corollary (Corollary 8, restated)**.** Verification of feedforward width-4 ReLU networks on convex $\mathrm{Init}, \mathrm{Acc} \subseteq [0,1]^4$ is NP-complete.

Proof. NP*-membership.* Apply Lemma 36 with $T = 1$ (no iteration): a nondeterministic verifier guesses a rational $x \in \mathrm{Init}$ of polynomial bit complexity, evaluates $N(x)$ exactly, and checks $N(x) \in \mathrm{Acc}$.

NP*-hardness.* We reduce from Theorem 7(i). Given a T-bounded reachability instance $(N_M, \mathrm{Init}, \mathrm{Acc}, T)$ with T in unary, where N_M and $\mathrm{Init}, \mathrm{Acc}$ are as constructed in Appendix E.3, output the unrolled feedforward network $N := N_M^{[T \cdot C_M]}$ together with Init and Acc unchanged. The unrolled N has width 4, depth $T \cdot C_M \cdot D_M = \mathrm{poly}(|M|, T)$, and is polynomial-time constructible since T is unary.

It remains to verify the equivalence $\exists x \in \mathrm{Init} : N(x) \in \mathrm{Acc} \iff \exists x \in \mathrm{Init}, \exists t \leq T \cdot C_M : N_M^{[t]}(x) \in \mathrm{Acc}$. The reverse direction is immediate by absorption: in the construction of Appendix E.3, Acc corresponds to "state q_F reached and $x_4 = 0$". Since q_F is the unique accepting (and thus terminal) state of M, once the trajectory enters Acc, N_M acts as the identity on Acc at every subsequent iteration, so $N_M^{[T \cdot C_M]}(x) \in \mathrm{Acc}$ as well. The forward direction is trivial (take $t = T \cdot C_M$). □

G Statements for various activation functions

The hypotheses of Theorem 10 (ρ continuous, non-polynomial, polynomial-time computable, C^1 at some point with non-zero derivative) are mild and cover every smooth activation in modern use. The following table summarizes which standard activations satisfy the assumptions and inherit the consequent complexity statements at width 6.

Table 2: Coverage of Theorem 10 on standard activations.

Activation ρ	Conditions of Theorem 10	Notes
$\tanh$	yes	smooth on $\mathbb{R}$
sigmoid $\sigma(x) = (1 + e^{-x})^{-1}$	yes	smooth on $\mathbb{R}$
GELU $x \cdot \Phi(x)$	yes	smooth on $\mathbb{R}$
Swish $x \cdot \sigma(\beta x)$	yes	smooth on $\mathbb{R}$, any $\beta > 0$
ELU	yes	C^1 except at 0; assumption holds at any other point
softplus $\log(1 + e^x)$	yes	smooth on $\mathbb{R}$
$\arctan$	yes	smooth on $\mathbb{R}$
Gaussian e^{-x^2}	yes	smooth on $\mathbb{R}$

For *exact* verification (without robustness), much stronger lower bounds hold for several of these activations. The following result of Wurm [2024, 2023b] shows that exact verification is $\exists\mathbb{R}$-hard for a broad class of activations, including those permitting transcendental constraints.

Theorem 47 (Wurm 2024, 2023b)**.** *With identity nodes allowed:*

1. *Verification is $\exists\mathbb{R}$-complete for any non-linear polynomial ρ, and $\exists\mathbb{R}$-hard whenever ρ coincides with a non-linear polynomial on an interval.*

2. *Verification is $\exists\mathbb{R}$-hard for any activation ρ that allows pp-defining a function coinciding with the exponential on an interval (in particular, the exponential, ELU, $e^{-|x|}$, the Gaussian e^{-x^2}, $\arctan$, and the Gallant–White cosine-smasher).*

These results explain why we restrict our analysis to the *robust* variant: exact verification falls outside the standard polynomial hierarchy, while robust verification recovers NP-completeness (Theorem 10).

H Neural ODEs at bounded dimension

Our results are stated for discrete iterated networks. In this section we transfer them to Neural ODEs [Chen et al., 2018] via the natural correspondence established by Duan et al. [2024], obtaining a *Lipschitz piecewise-affine* vector field of dimension 5 whose splitting-method-discretized flow simulates an arbitrary Turing machine. Verification hardness transfers automatically. The pure continuous-time question remains open with a precise technical obstruction.

H.1 The Duan et al. correspondence

Duan et al. [2024] establish that vanilla feedforward networks, not only residual ones, are exact numerical discretizations of (non-autonomous) dynamical systems. Their Theorem 1 shows that for every flow map ϕ^τ of a Lipschitz ODE on a compact set, there exists a vanilla leaky-ReLU FNN of width d approximating it; the construction proceeds via splitting of an associated neural ODE. The mechanism implicit in their construction also yields the converse direction we need: every leaky-ReLU FNN of width exactly d with non-singular layers *is* the splitting-method iteration of an explicit non-autonomous piecewise-affine vector field. We state and use this converse direction below.

Lemma 48 (Splitting interpretation, after Duan et al. [2024])**.** *Let σ_α be the leaky-ReLU activation, $\alpha \in (0,1)$, and let $N\colon \mathbb{R}^d \to \mathbb{R}^d$ be a vanilla feedforward leaky-ReLU network of depth L, with all hidden layers of width exactly d and non-singular weight matrices. Then there exists a Lipschitz piecewise-affine vector field $v\colon \mathbb{R}^d \times [0,L] \to \mathbb{R}^d$, piecewise constant in t on each unit interval $[k, k+1)$ and computable in time $\mathrm{poly}(\mathrm{size}(N))$ from the parameters of N, such that N is exactly the time-L output of a Lie-Trotter splitting discretization of the non-autonomous ODE $\dot{x} = v(x,t)$ at step size 1. In particular, $N(x_0)$ equals the L-th iterate of the splitting scheme started at x_0.*

Proof. For each layer $\ell \in \{1, \dots, L\}$, the layer $x \mapsto \sigma_\alpha(W_\ell x + b_\ell)$ is the Lie-Trotter composition (Trotter [1959], see e.g. Holden et al. [2010]) of the time-1 flow of an affine ODE $\dot{x} = (W_\ell - I)x + b_\ell$ (whose flow at $t = 1$ realizes $x \mapsto W_\ell x + b_\ell$ when W_ℓ is non-singular, after re-parameterization; see Duan et al., 2024 Section 3.5–3.6) and the time-1 flow of the activation ODE $\dot{x} = (\sigma_\alpha - I)(x)$ rescaled appropriately. Setting $v(x,t) = v_\ell(x)$ for $t \in [\ell - 1, \ell)$, where v_ℓ encodes both ingredients of layer ℓ, gives the required non-autonomous vector field. Composing L unit splitting steps reproduces exactly N. □

From ReLU to leaky-ReLU at width $+1$. Lemma 48 requires leaky-ReLU rather than ReLU. The two activations are related by the elementary identity $\mathrm{ReLU}(x) = \dfrac{\sigma_\alpha(x) - \alpha\, x}{1 - \alpha}$, which expresses ReLU as a linear combination of one leaky-ReLU neuron and one identity register. By keeping x alongside $\sigma_\alpha(x)$ in a single extra register, every ReLU neuron at width w can be emulated by a leaky-ReLU subnetwork at width $w + 1$, with the same depth modulo a constant. We record this as a folklore lemma.

Lemma 49 ($\mathrm{ReLU} \to$ leaky-ReLU emulation at width $+1$)**.** *For every fixed $\alpha \in (0,1)$ and every ReLU network $N \in \mathcal{NN}^{\mathrm{ReLU}}_{n,n,w}$, there exists a leaky-ReLU network $\widetilde{N} \in \mathcal{NN}^{\sigma_\alpha}_{n,n,w+1}$ with the same depth up to a constant factor, such that $\widetilde{N}(x) = N(x)$ for all $x \in \mathbb{R}^n$.*

Proof. For each ReLU neuron $y = \mathrm{ReLU}(z)$ in N, the identity $\mathrm{ReLU}(z) = (\sigma_\alpha(z) - \alpha z)/(1 - \alpha)$ allows us to compute y as a linear combination of $\sigma_\alpha(z)$ and z. Implementing this in a vanilla leaky-ReLU FNN requires keeping z alongside $\sigma_\alpha(z)$ between layers; one extra register suffices for this passthrough, where the identity wire is itself realized by a single leaky-ReLU neuron via Proposition 40. The depth increases by a constant per layer. □

H.2 Neural ODE universality at dimension 5

Combining Theorem 3 (width-4 ReLU universality) with Lemma 49 (ReLU $\to$ leaky-ReLU at width 5) and Lemma 48 (leaky-ReLU width $d \leftrightarrow$ non-autonomous ODE in dimension d) yields the following.

Theorem 50 (Discretized Neural ODE consequence at dimension 5)**.** *For every length bound $n \geq 1$, there exists a polynomial-time constructible non-autonomous Lipschitz piecewise-affine vector field $f_n \colon \mathbb{R}^5 \times [0, +\infty) \to \mathbb{R}^5$, piecewise constant in t on each unit interval and realizable layer-by-layer as a leaky-ReLU vanilla neural network of width 5, and a fixed convex acceptance region $R_{\mathrm{acc}}^{\mathrm{ODE}} \subseteq [0,1]^5$ specified by rational linear inequalities, such that the following holds. For every deterministic Turing machine M over $\{0,1\}$ and every input $w \in \{0,1\}^*$ with $|w| \leq n$, there is a polynomial-time-computable initial condition $\xi_0(M,w) \in [0,1]^5$ and an integer step count $L_M = \mathrm{poly}(|M|)$ such that the Lie-Trotter splitting discretization sequence $\{\xi_k\}_{k \geq 0}$ of $\dot{\xi} = f_n(\xi, t)$ at step size 1, started at $\xi_0(M,w)$, satisfies: M accepts $w \iff \exists\, k \geq 0, \quad \xi_{k \cdot L_M} \in R_{\mathrm{acc}}^{\mathrm{ODE}}$. Taking M to be a fixed universal Turing machine U, f_n becomes a single non-autonomous vector field parameterized by length only, so the discretized flow simulates any M on any input of length $\leq n$.*

Caveat on continuous flow. The discretization sequence $\{\xi_k\}$ is what reproduces the network iteration; it is not in general equal to the continuous flow $\xi \colon [0, +\infty) \to \mathbb{R}^5$ of $\dot{\xi} = f_n(\xi, t)$ sampled at integer times. Splitting methods are only first-order accurate in the step size, and for step size 1 the gap is real. The continuous flow exists and is unique by Lipschitz of f_n, but its values at integer times need not encode TM configurations. Furthermore, the vector field is non-autonomous (piecewise constant in t, with one piece per unit step); whether the construction can be carried out with an autonomous vector field is left open. See Appendix H.4 for related open questions.

Proof. By Theorem 3, the width-4 ReLU network N simulates a universal Turing machine on convex inputs $[0,1]^4$. By Lemma 49, an associated width-5 leaky-ReLU network $\widetilde{N} \colon [0,1]^5 \to [0,1]^5$ realizes the same iteration on the embedded set $\{(x, 0) : x \in [0,1]^4\} \subseteq [0,1]^5$. By Lemma 48 applied to $\widetilde{N}$, there is a non-autonomous Lipschitz piecewise-affine vector field $f_n \colon \mathbb{R}^5 \times [0, +\infty) \to \mathbb{R}^5$, piecewise constant in t on unit intervals, whose Lie-Trotter splitting discretization at step 1 over $L_M = \mathrm{poly}(|M|)$ time intervals reproduces $\widetilde{N}$. Setting $R_{\mathrm{acc}}^{\mathrm{ODE}} = \mathrm{Acc} \times \{0\}$ (with Acc from the verification recipe of Section 4.1) closes the equivalence. □

Reading. Width 4 in the discrete network and dimension 5 in the Neural ODE are the same quantity up to the leaky-ReLU emulation overhead. The "+1" is the cost of moving from ReLU to leaky-ReLU, the splittable activation required by the Duan correspondence. The question of whether Theorem 3 can be carried out directly with leaky-ReLU at width 4 (which would yield a dimension-4 Neural ODE), or whether the non-autonomous vector field can be made autonomous, is left open.

H.3 Verification hardness transfers

The verification spectrum of Theorem 7 transfers to Neural ODEs by Theorem 50.

Corollary 51 (Verification spectrum for Neural ODEs at dimension 5)**.** *For Lipschitz piecewise-affine Neural ODEs $\dot{\xi} = f(\xi, t)$ in dimension 5, with convex initial and acceptance regions $R_{\mathrm{init}}^{\mathrm{ODE}}, R_{\mathrm{acc}}^{\mathrm{ODE}} \subseteq [0,1]^5$, the discretized reachability problem "does the splitting-discretized trajectory enter $R_{\mathrm{acc}}^{\mathrm{ODE}}$ within T discretization steps?" is NP-complete with T unary, NEXP-complete with T binary, and RE-complete (undecidable) without a time bound.*

An analogous transfer of Theorem 10 (smooth ρ at width 6) to Neural ODEs requires extending the correspondence [Duan et al., 2024] from leaky-ReLU to other activations. The Duan et al. argument is stated for leaky-ReLU and relies specifically on its invertibility; we leave the extension to $\tanh$, GELU, sigmoid, Swish, and other smooth activations as an open direction.

Practical reading. A practitioner verifying a low-dimensional Neural ODE faces the full hardness spectrum already at dimension 5. The empirical observation that complete verifiers degrade rapidly

with integration time [Katz et al., 2017, Wang et al., 2021] is, on this account, an intrinsic complexity-theoretic phenomenon and not a solver artifact: integration time plays the role of Turing-machine time, and the hardness ladder is unconditional on solver design.

H.4 Open: pure continuous-time exact simulation

Theorem 50 establishes universality of Neural ODEs *under splitting-method discretization*. The discretized flow at integer times reproduces TM steps exactly. The continuous flow at non-integer times is well-defined (by Lipschitz of f) but need not encode any meaningful TM configuration.

A natural strengthening would be: a Lipschitz piecewise-affine $f\colon \mathbb{R}^d \to \mathbb{R}^d$ whose *continuous flow* satisfies $\xi(n) = N^{[n]}(\xi_0)$ for every integer n, for N a non-trivial non-linear PA map. Whether this is achievable is open, and we record two related questions.

Three known regimes. For the broader question of continuous-time TM simulation, three regimes are established in the literature:

- Polynomial (rather than piecewise-affine) vector fields [Bournez et al., 2007] (analytic flow, but no longer in the Neural-ODE class as commonly used).
- Piecewise-constant discontinuous fields, e.g. PCD systems [Asarin et al., 1995, Koiran et al., 1994] (closed-form jumps at cell boundaries, but not Lipschitz).
- Splitting-method discretization (Theorem 50): exact simulation at discretization times, with continuous interpolation between them.

The Lipschitz piecewise-affine class, the class most aligned with practical Neural ODEs, is not covered by any of these.

Two open questions. We highlight two natural open questions for future work, in increasing order of strength:

- (Robust continuous-time) Does there exist an autonomous Lipschitz piecewise-affine vector field $f\colon \mathbb{R}^d \to \mathbb{R}^d$ (for some constant d) such that the continuous flow simulates a universal Turing machine in the following robust sense: for every machine M and input w, the trajectory enters an ε-neighborhood of a designated accepting region in finite time iff M accepts w?
- (Exact continuous-time) Does there exist such an f with $\xi(n) = N^{[n]}(\xi_0)$ exactly at integer times, for some non-trivial PA map N derived from M?

We leave both questions for future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state the main result (Turing completeness of iterated width-4 ReLU networks on convex input spaces, Theorem 3) and its three consequences (verification spectrum, Theorem 7; robust hardness at width 6 for smooth activations, Theorem 10; Neural ODEs at dimension 5, Theorem 50); these match the formal statements of Sections 2–5 and Appendix H.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 says what the results do and do not establish (the worst-case analysis does not by itself explain solver behavior on trained networks) and lists four open problems (width 2, continuous-time simulation, trained and constrained networks, softmax and attention). Appendix H states the limitation of the Neural ODE consequence: exact simulation holds under splitting-method discretization only (Appendix H.2), and continuous-time simulation is left open (Appendix H.4).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All theorems, lemmas, and corollaries are numbered, cross-referenced, and stated with their full assumptions; proof sketches appear in the main body and complete proofs in Appendices B–H (width-3 step simulation, encoding-validity gadget, full proof of Theorem 3, verification spectrum, robust hardness transfer, activation functions, Neural ODEs).

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [N/A]

   Justification: The paper is theoretical and contains no experiments; all results are formal theorems with proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [N/A]

   Justification: The contribution is theoretical; no experiments, datasets, or code are part of the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [N/A]

   Justification: The paper does not include experiments; there are no training or test procedures, hyperparameters, or optimizers involved.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

Justification: The paper contains no experiments; results are deterministic mathematical statements (completeness for NP, NEXP, RE).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [N/A]

   Justification: No experiments are conducted; the paper required no computational experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics; the work is purely theoretical (complexity-theoretic lower bounds on neural network verification) and conforms with the Code in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper establishes foundational complexity-theoretic results about neural network verification with no direct deployment path or application to specific systems; no concrete positive or negative societal impacts are foreseen beyond informing the design of verification tools.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper releases no models, datasets, or other artifacts that could pose a misuse risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [N/A]

Justification: The paper does not use any external code, datasets, or model assets; prior theoretical results are credited via standard bibliographic citations.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release any new assets (no datasets, code, or models).

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper involves no crowdsourcing and no human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper involves no human-subjects research, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not part of the methodology of this paper; any use was limited to writing, editing, and formatting assistance and did not affect the scientific content.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.