LUCID: Learning-Enabled Uncertainty-Aware Certification of Stochastic Dynamical Systems

Ernesto Casablanca[1], **Oliver Schön**[1], **Paolo Zuliani**[2], **Sadegh Soudjani**[3]
[1]Newcastle University, Newcastle upon Tyne, United Kingdom
[2]Università di Roma "La Sapienza", Rome, Italy
[3]Max Planck Institute for Software Systems, Kaiserslautern, Germany
e.casablanca2@ncl.ac.uk, o.schoen2@ncl.ac.uk, zuliani@di.uniroma1.it, sadegh@mpi-sws.org

Abstract

Ensuring the safety of AI-enabled systems, particularly in high-stakes domains such as autonomous driving and healthcare, has become increasingly critical. Traditional formal verification tools fall short when faced with systems that embed both opaque, black-box AI components and complex stochastic dynamics. To address these challenges, we introduce LUCID (Learning-enabled Uncertainty-aware Certification of stochastIc Dynamical systems), a verification engine for certifying safety of black-box stochastic dynamical systems from a finite dataset of random state transitions. As such, LUCID is the first known tool capable of establishing quantified safety guarantees for such systems. Thanks to its modular architecture and extensive documentation, LUCID is designed for easy extensibility.

LUCID employs a data-driven methodology rooted in control barrier certificates, which are learned directly from system transition data, to ensure formal safety guarantees. We use conditional mean embeddings to embed data into a Reproducing Kernel Hilbert Space (RKHS), where an RKHS ambiguity set is constructed that can be inflated to robustify the result to out-of-distribution behavior.

A key innovation within LUCID is its use of a finite Fourier kernel expansion to reformulate a semi-infinite non-convex optimization problem into a tractable linear program. The resulting spectral barrier allows us to leverage the fast Fourier transform to generate the relaxed problem efficiently, offering a scalable yet distributionally robust framework for verifying safety. LUCID thus offers a robust and efficient verification framework, able to handle the complexities of modern black-box systems while providing formal guarantees of safety. These unique capabilities are demonstrated on challenging benchmarks.

Source Code — https://github.com/TendTo/lucid
Documentation — https://tendto.github.io/lucid/

1 Introduction

Embodied forms of AI are on the rise, including applications such as autonomous vehicles, robotics, personalized healthcare, and smart infrastructure. Their core functionality is built around the advances of deep learning, enabling systems to understand and reason in complex and human-like ways to produce a desired behavior. Whilst the black-box nature of AI components has been a crucial contributor to the widespread success of deep learning, in response to a rapidly evolving legal scrutinization (Veale and Zuiderveen Borgesius 2021), the resulting lack of traceability and certifiability has put a premature halt to its deployment in safety-critical applications.

As deep learning agents are left to choose their actions autonomously in closed-loop interaction with the physical world, they are confronted with a world riddled with randomness and uncertainty. Efforts to establishing trust in their safe operation have been largely focused on developing tools to verify the input–output behavior of Neural Networks (NNs) embedded in the systems (Liu et al. 2021). Unfortunately, there do not exist any tools that could take these results and certify the safety of the entire system, as uncertainty-aware models of closed-loop systems are rarely available and simulators are often too complicated and opaque to perform verification directly (Wongpiromsarn et al. 2023). This motivates a holistic black-box treatment (Corso et al. 2021), where safety guarantees are to be established from behavioral data and in account of the unknown laws of randomness themselves.

There exist no tools capable of quantifying safety guarantees for complex stochastic closed-loop systems from data. For the setting where a model of the closed-loop system is available, NPINTERVAL by Harapanahalli, Jafarpour, and Coogan (2023) verifies safety of nonlinear systems with non-deterministic disturbance based on existing NN verification tools. See the references therein and the annual friendly competition by Abate et al. (2024) for adjacent work. Few data-driven approaches for stochastic systems exist. OMNISAFE is a comprehensive platform for the development of safe Reinforcement Learning (RL) algorithms (Ji et al. 2024). However, safe RL does generally only *encourage* safer behavior, without providing any rigorous or quantified guarantees on the probability of the absence of unsafe behavior. As a popular working principle for certifying safety, *Control Barrier Certificates* (CBCs) (Prajna 2006) are at the basis of tools such as TRUST (Gardner et al. 2025), which supports only polynomial dynamics, and FOSSIL (Edwards, Peruffo, and Abate 2024), which is model-based, and both being restricted to deterministic systems.

To close this gap (see Table 1), we introduce LUCID,

Tool	Supported Features					
	Guarantees	Data Driven	Stochastic Dyn.	Non-Poly. Dyn.	Stat. Correct.	Closed Loop
LUCID (this work)	✓	✓	✓	✓	✓	✓
TRUST (2025)	✓	✓	✗	✗	✓[1]	✓
FOSSIL (2024)	✓	✗	✗	✓	NA	✓
OMNISAFE (2024)	✗	✓	✓	✓	✗	✓
NPINTERVAL (2023)	✓	✗	✓/✗[2]	✓	NA	✓

Table 1: Qualitative comparison with existing tools based on their supported features: quantified safety guarantees, data driven, stochastic dynamics, non-polynomial dynamics, statistical correctness guarantees of the learned model (only applicable if data driven), and support for closed-loop systems.

the first verification engine for black-box stochastic dynamical systems with quantified guarantees. At its core, LUCID learns CBCs for unknown systems based solely on data, by constructing an uncertainty-aware estimator of the expected system behavior based on *Conditional Mean Embeddings* (CMEs) (Muandet et al. 2017). Crucially, the underlying learning framework establishes quantified safety probabilities with distributionally robust guarantees for arbitrary smooth dynamics, thus relaxing the need for restrictive structural assumptions common to related approaches. For instance, Schön, Zhong, and Soudjani (2024) learn CBCs for systems with polynomial dynamics and Chen et al. (2025) assume systems with known deterministic component.

Alternative approaches often leave the statistical correctness with respect to the underlying data-generating process unaddressed: Kazemi and Soudjani (2020) use model-free reinforcement learning, relying on known Lipschitz constants; Lew and Pavone (2021) compute reachable sets for stochastic systems using a sampling-based scheme, which yields only asymptotic guarantees; and Salamati et al. (2024) use a scenario-based method, based on Lipschitz constants and exponentially large datasets. For data-driven safety verification of stochastic systems via conformal prediction (Lindemann et al. 2024), no tools are available.

We summarize the main contributions of this work:

- We introduce LUCID, a novel verification engine that learns control barrier certificates from data using kernel-based CMEs. LUCID uses a tractable reformulation of the problem via a finite Fourier expansion, enabling efficient barrier synthesis based on a linear program.
- A robust and extensible software implementation, with both C++ and Python interfaces, supporting configuration via YAML, JSON, or Python scripts, and offering both a *Command Line Interface* (CLI) and a *Graphical User Interface* (GUI).
- Numerical evaluation on a suite of benchmarks, demonstrating LUCID's unique capabilities, robustness, and practical utility.

[1]Assuming the data satisfies persistence of excitation.

[2]Accepts non-deterministic bounded disturbances.

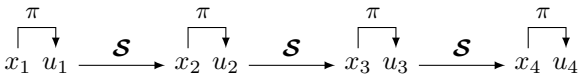

Figure 1: Evolution of the closed-loop system.

Organization The paper is organized as follows. Section 2 provides the theoretical foundations of LUCID, including the safety of black-box dynamical systems, control barrier certificates, conditional mean embeddings, and derivation of the relaxed linear program. Section 3 describes the architecture and functionalities of LUCID alongside a running example, presenting LUCID's components and how they interact. Section 4 evaluates LUCID on a suite of benchmarks. Finally, Section 5 provides a conclusion and future extensions.

2 Theoretical Working Principles

2.1 Safety of Black-Box Dynamical Systems

System Description Many AI-driven systems exhibit complex, nonlinear, and stochastic behavior that can be modeled as discrete-time stochastic processes with Markovian dynamics:

$$\mathcal{S}\colon \quad x_{t+1} = f(x_t, a_t, w_t), \quad w_t \sim p_w, \tag{1}$$

where $f\colon \mathbb{X} \times \mathbb{A} \times \mathbb{W} \to \mathbb{X}$ is a continuous vector field describing the evolution of the system state $x_t \in \mathbb{X} \subset \mathbb{R}^n$ over time $t \in \mathbb{N}_{\geq 0}$, driven by control actions $a_t \in \mathbb{A} \subset \mathbb{R}^m$ and process noise $w_t \in \mathbb{W} \subset \mathbb{R}^l$. The noise is assumed to be drawn from a stochastic distribution p_w in an independent and identically distributed (i.i.d.) manner. This general formulation subsumes a wide range of systems, including discrete-time Markov Decision Processes (MDPs).

Black-Box Policies Here, the focus is on systems $\mathcal{S}$ driven by black-box control policies of the form $\pi\colon \mathbb{R}^n \to \mathbb{R}^m$, i.e., at every time step $t = 0, 1, 2, \ldots$ a continuous action $a_t \in \mathbb{A}$ is selected based on the current state x_t. Such policies could be, for example, neural networks trained via RL, or any other black-box function generating actions in $\mathbb{R}^m$ (see Figure 1). The resulting closed-loop system is denoted as $\mathcal{S}^\pi$.

Dataset Due to their complexity and opacity, such systems must often be treated as black boxes in their entirety, assuming only access to a finite amount, N, of system observations

$$\mathcal{D}_N\colon \quad \{(x^i, a^i, x^i_+)\}_{i=1}^N, \tag{2}$$

where every sampled transition from $x^i \in \mathbb{X}$ to a successor $x^i_+ \in \mathbb{X}$ is generated as a realization

$$(x^i, a^i, x^i_+) \sim \int \delta_{f(x^i,a^i,w)}(dx^i_+)\, p_w(dw)\, \mathcal{U}_{\mathbb{X}}(dx^i)\, \mathcal{U}_{\mathbb{A}}(da^i),$$

with $\mathcal{U}_{\mathbb{X}}$ and $\mathcal{U}_{\mathbb{A}}$ uniform distributions on $\mathbb{X}$ and $\mathbb{A}$, respectively, and δ indicating the Dirac delta distribution capturing the system dynamics.

Assuming i.i.d. data of the form (2) and full observability offers statistical guarantees on the consistency of the data-driven CME constructed in Section 2.3. We point to literature addressing dependent data, assuming either ergodicity, burn-in time, or reduced sample effectiveness w.r.t. the mixing time (Zhang et al. 2024; Ziemann et al. 2023).

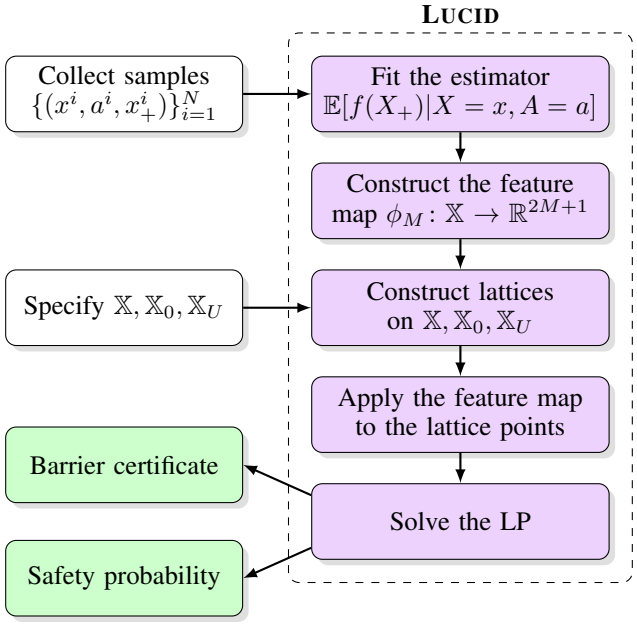

Figure 2: Sequence of steps LUCID goes through to generate a barrier certificate.

Safety Problem A question that often arises when designing a policy π for a system $\boldsymbol{\mathcal{S}}$, especially in engineering contexts, is if the closed-loop system $\boldsymbol{\mathcal{S}}^\pi$ elicits safe behavior. That is, when starting from a domain $\mathbb{X}_0 \subset \mathbb{X}$ under a policy π, the system $\boldsymbol{\mathcal{S}}^\pi$ shall avoid unsafe regions $\mathbb{X}_U \subset \mathbb{X}$ (such as obstacles) for at least some predefined time horizon $T \in \mathbb{N} \cup \{\infty\}$.

Certifying safety of a discrete-time stochastic system over an uncountable state space does not generally admit an analytical solution and is extremely challenging, especially for complex dynamics (Abate et al. 2008). Furthermore, for unbounded stochasticity $w_t \sim p_w$, there is generally no yes/no answer to safety, but a safety probability $P_{\text{safe}}(\boldsymbol{\mathcal{S}}^\pi) \in [0, 1)$. Note that given a policy π, for every initial state $x_0 \in \mathbb{X}_0$ there is an associated safety probability. Our goal is to estimate a *lower bound* on the infimum of such probabilities across all initial states in $\mathbb{X}_0$.

Problem Statement Assuming access to a finite dataset $\boldsymbol{\mathcal{D}}_N$ from the black-box system $\boldsymbol{\mathcal{S}}$ in (1), quantify a certifiable lower bound on the probability of the black-box system $\boldsymbol{\mathcal{S}}^\pi$ being safe with respect to a safety specification given by $(\mathbb{X}_0, \mathbb{X}_U, T)$.

Available sampling-based methods such as Monte Carlo simulation are unfit to solve this problem, as they compute guarantees for fixed initial conditions $x_0 \in \mathbb{X}_0$. LUCID provides a solution for continuous sets $\mathbb{X}_0$ by automating the steps in Figure 2, outlined in the next sections.

2.2 Control Barrier Certificates

CBCs[3] leverage the concept of set invariance to arrive at an abstraction-free numerical solution to finding a lower bound on $P_{\text{safe}}(\boldsymbol{\mathcal{S}}^\pi)$. This has made them popular tools for safety verification and controller synthesis (Prajna 2006).

A non-negative function $\boldsymbol{B} \colon \mathbb{X} \to \mathbb{R}_{\geq 0}$ is a CBC of a system $\boldsymbol{\mathcal{S}}$ with reference to an unsafe set $\mathbb{X}_U$ if it satisfies

(a) $\forall x_0 \in \mathbb{X}_0 \colon \boldsymbol{B}(x_0) \leq \eta$;

(b) $\forall x_U \in \mathbb{X}_U \colon \boldsymbol{B}(x_U) \geq 1$; and

(c) $\forall x \in \mathbb{X},\ \exists a \in \mathbb{A} \colon \mathbb{E}[\boldsymbol{B}(X_+)|X = x,\ A = a] - \boldsymbol{B}(x) \leq c$;

for some constants $1 > \eta \geq 0$ and $c \geq 0$. Here, upper case X_+, X, and A denote the random variables underlying concrete realizations (x_t, a_t, x_{t+1}) elicited by $\boldsymbol{\mathcal{S}}$. Intuitively, if one can find a CBC for a system $\boldsymbol{\mathcal{S}}$, then, a lower bound on the probability of $\boldsymbol{\mathcal{S}}$ being safe can be quantified based on the distance between the two level sets 1 and η (Kushner 1967):

$$P_{\text{safe}}(\boldsymbol{\mathcal{S}}^\pi) \geq 1 - (\eta + cT), \tag{3}$$

where T is the desired time horizon.

Whilst (3) can provide a robust assessment of a system's safety, in practice, the bound can be overly conservative and thus several improved variants of the original barrier constraints exist (e.g., Anand et al. 2022). As these conditions in essence all rely on the computation of a stochastic constraint with respect to the expected behavior of the system, they are basically interchangeable.

Although there exist model-*based* efficient solutions to this problem for linear and control affine systems, this is generally a semi-infinite problem, demanding a data-driven solution. Furthermore, for the data-driven case, establishing constraint (c) rigorously without relying on impractical assumptions is extremely challenging.

2.3 Data-Driven Dynamics Estimation via Conditional Mean Embeddings

To reason about the expected value of a random variable, embedding the variable into a (higher dimensional) space and forming a data-driven estimate is a well-established concept in machine learning (Schölkopf and Smola 2002; Steinwart and Christmann 2008). Following the same reasoning, the *kernel mean embedding* represents the embedding of a probability measure into an RKHS via a feature map ϕ associated with a positive definite kernel k (Smola et al. 2007; Muandet et al. 2017). For conditional probability distributions a similar concept exists: CMEs can be used to model the expected value of any RKHS function $g \colon \mathbb{X} \to \mathbb{R}$ under a stochastic process such as $\boldsymbol{\mathcal{S}}$ (Park and Muandet 2020; Muandet et al. 2017). For the Gaussian kernel,

$$k(x, x') := \sigma_f^2 \exp\left(-\tfrac{1}{2}(x - x')^\top \Sigma\,(x - x')\right), \tag{4}$$

where $\Sigma := \operatorname{diag}(\sigma_l)^{-2}$, with hyperparameters $\sigma_f, \sigma_l \in \mathbb{R}$, the associated RKHS encompasses all smooth functions g.

[3]CBCs and discrete-time *Control Barrier Functions* (CBFs) (Cosner et al. 2024) are equivalent.

A data-driven estimate can then be obtained in closed form from a finite amount of data $\mathcal{D}_N$:

$$\mathbb{E}[f(X_+) \,|\, X = x,\, A = a] \approx k_{XA}^N(x,a)^\top \left[K_{XA}^N + N\lambda I_N\right]^{-1} f(X_+^N), \tag{5}$$

with column vector $k_{XA}^N(x,a) := [k((x^i,a^i),(x,a))]_{i=1}^N$, Gram matrix $K_{XA}^N := [k((x^i,a^i),(x^j,a^j))]_{i,j=1}^N$, regularization factor $\lambda \geq 0$, identity matrix I_N, and $f(X_+^N) := [f(x_+^i)]_{i=1}^N$. The empirical estimator in (5) converges in expectation to the true CME for $N \to \infty$ and $\lambda \to 0$ (Park and Muandet 2020).

It is common practice to robustify empirical estimates such as (5) to out-of-sample behavior by constructing an RKHS ambiguity set centered at the empirical CME. The result is a distributionally robust estimator with an adjustable robustness radius. The details are omitted here for brevity, but the interested reader is referred to the works by Kuhn, Shafiee, and Wiesemann (2025) and Li et al. (2022).

2.4 Data-Driven Spectral Barriers

Based on the data-driven estimator in (5), the problem of computing CBCs using data can be formulated as a nonconvex semi-infinite program, which for general classes of systems and barriers is extremely difficult to solve. To arrive at a tractable solution, LUCID conducts two additional steps:

1. Spectral Abstraction Inspired by the popular random Fourier features approach by Rahimi and Recht (2007), the Gaussian kernel (4) admits a Fourier expansion

$$k(x,x') \equiv \sigma_f^2 \int_{\mathbb{R}^n} \mathcal{N}(d\omega \,|\, 0, \Sigma)\, e^{\mathbf{i}\omega^\top (P(x) - P(x'))}, \tag{6}$$

with the zero-mean Gaussian distribution $\mathcal{N}(d\omega \,|\, 0, \Sigma)$ with covariance Σ, the imaginary unit $\mathbf{i} := \sqrt{-1}$, and where the affine transform $x \mapsto P(x)$ maps the domain $\mathbb{X}$ into the unit hypercube $[0,1]^n$. Partitioning the space of spatial frequencies $\omega \in \mathbb{R}^n$ into a set of discrete frequency bands (see Figure 3) yields a spectral abstraction of the associated RKHS, i.e., characterizing learnable functions $\boldsymbol{B}$ in the form of Fourier series. By truncating the series to a finite number, M, of fixed frequency bands $\omega_j \in \mathbb{R}^n$, $j \in \{0,\dots,M\}$, the resulting barriers are of the form

$$\boldsymbol{B}(x) = \alpha_0 + \sum_{i=1}^{M} \alpha_i \cos\left(\omega_i^\top P(x)\right) + \beta_i \sin\left(\omega_i^\top P(x)\right).$$

Notably, $\boldsymbol{B}$ admits the linear form $\boldsymbol{B}(x) = \phi_M(x)^\top b$, with a truncated Fourier feature map $\phi_M \colon \mathbb{X} \to \mathbb{R}^{2M+1}$, parametrized by learnable spectral amplitudes

$$b = \left[\frac{\alpha_0}{\sigma_f^2 \mathbf{w}_0^2} \;\; \frac{\alpha_1}{2\sigma_f^2 \mathbf{w}_1^2} \;\; \frac{\beta_1}{2\sigma_f^2 \mathbf{w}_1^2} \;\; \cdots \;\; \frac{\alpha_M}{2\sigma_f^2 \mathbf{w}_M^2} \;\; \frac{\beta_M}{2\sigma_f^2 \mathbf{w}_M^2}\right]^\top \in \mathbb{R}^{2M+1},$$

where the weights $\mathbf{w}_0,\dots,\mathbf{w}_M \in \mathbb{R}_{\geq 0}$ associated with each frequency band are determined efficiently from the kernel's Gaussian spectral measure via the multivariate CDF (see Figure 4). The CME-based estimator (5) is approximated in the same finite basis via $H \in \mathbb{R}^{(2M+1)\times(2M+1)}$ such that

$$k_{XA}^N(x,a)^\top \left[K_{XA}^N + N\lambda I_N\right]^{-1} \Phi_{M,+}^N \approx \varphi_M(x,a)^\top H,$$

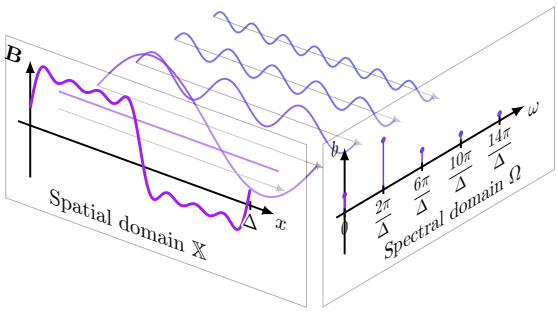

Figure 3: Spectral barrier certificate $\boldsymbol{B}(x) = \phi_M(x)^\top b$.

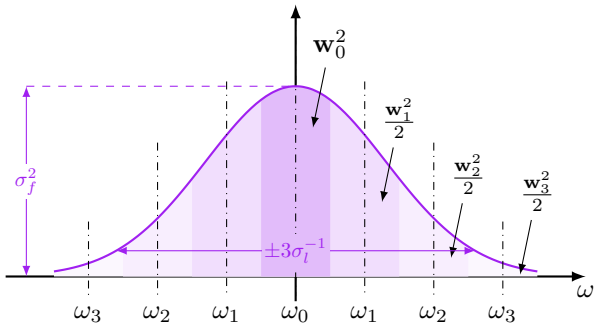

Figure 4: Abstraction of a 1-dim. Gaussian spectral measure of the Gaussian kernel, shown in (6).

where $\Phi_{M,+}^N := [\phi_M(x_+^i)^\top]_{i=1}^N$, $\varphi_M(x,a) \colon \mathbb{X} \times \mathbb{A} \to \mathbb{R}^{2M+1}$ a feature map augmenting $\phi_M(x)$, and with an approximation error decreasing exponentially with M (Rahimi and Recht 2007). This reduces the barrier synthesis problem to a semi-infinite linear program, with barrier conditions linear in b:

(a) $\forall x_0 \in \mathbb{X}_0 \colon\ \phi_M(x_0)^\top b \leq \eta$;
(b) $\forall x_U \in \mathbb{X}_U \colon\ \phi_M(x_U)^\top b \geq 1$; and
(c) $\forall x \in \mathbb{X},\ \exists a \in \mathbb{A} \colon\ \varphi_M(x,a)^\top (Hb - b) \leq c$.

For the case where a is provided by a given fixed policy π, $\varphi_M(x,a)$ reduces to $\phi_M(x)$.

2. Finite-Constraint Relaxation To obtain a program with *finitely* many constraints, trigonometric bounding results (Pfister and Bresler 2018; Schön, Zhong, and Soudjani 2025) are used to obtain a relaxed linear program by sampling spatial lattices on $\mathbb{X}$, $\mathbb{X}_0$, and $\mathbb{X}_U$. For example, to enforce barrier constraint (a) in Section 2.2, it suffices to construct a lattice $\{x_0^i\}_{i=1}^{N_0} \subset \mathbb{X}_0$ and impose the constraints $b^\top \phi_M(x_0^i) \leq \eta - \epsilon$ for $i = 1,\dots,N_0$, where $\epsilon > 0$ is a computed coefficient. Selecting lattices with a sampling density meeting the Nyquist-Shannon sampling theorem with respect to the highest frequency ω_M appearing in the barrier and truncated estimator, the relaxation retains all guarantees. As a result, the semi-infinite problem is relaxed to a finitely-constrained Linear Program (LP), provided in full in the extended version (Casablanca et al. 2025). Recall that given an appropriate robustness radius, the barriers produced by LU-

CID based on data $\mathcal{D}_N$ are statistically correct with respect to the unknown true system $\mathcal{S}$, and a lower bound for the safety probability is computed according to (3).

3 Tool Structure and Functionalities

LUCID implements the previously outlined functionality, written in C++ and designed to be used as a library or standalone executable. The choice of a low level language gives us plenty of freedom and fine-grained control over the execution of the software. We expose a set of interfaces allowing users to highly customize the verification process. A high-level visualization of LUCID's architecture is illustrated in Figure 5, while a more technical description can be found in the online documentation at

https://tendto.github.io/lucid/.

We also provide a Python wrapper, called PYLUCID, to facilitate the integration of the tool into existing workflows and effortlessly leverage well-established libraries such as NumPy (Harris et al. 2020) and SciPy (Virtanen et al. 2020). Moreover, PYLUCID can be extensively configured in a variety of ways (e.g., Python scripts, YAML files, GUI), making it the recommended way to operate LUCID. For the rest of the paper, we will thus focus on PYLUCID when describing the user interfaces. Further information about PYLUCID are deferred to the extended version (Casablanca et al. 2025).

Configuration To certify safety of a system, LUCID accepts a configuration comprising data from the system and the safety specification (see Figure 5). We suggest defining it as a *.yaml* or equivalent *.json* file. If more flexibility is needed, a Python script generating a configuration can be used instead. Regardless of their format, configuration files can be loaded with the command `pylucid <config file >` to start the tool's verification pipeline. The same configuration can also be passed directly as command line arguments. PYLUCID also provides a browser-based GUI to aid in the configuration and execution of the tool.

In its current form, LUCID implements all the functionality needed to certify safety of a closed-loop system with a given control policy π. Thus the action a in the previous section is replaced with the policy $\pi(x)$. Future releases will expand this core functionality to safe controller synthesis as well. Due to its modular architecture, LUCID can be easily extended in multiple directions, as outlined in Section 5.

3.1 Configuring and Running the Tool

LUCID is built with a modular architecture, where each component is responsible for a specific task in the certification process. Since the components extend from a common interface, they can be easily replaced or extended. An overview of the core components and how they interact is shown in Figure 5. Classes and interfaces available in PYLUCID are written in `monospace font`.

To make the explanation more intuitive, we will use a one-dimensional linear system as a running example:

$$\mathcal{S}: \quad x_{t+1} = 0.5x_t + w_t, \quad w_t \sim \mathcal{N}(\cdot\,|0, 0.01), \qquad (7)$$

with state space $\mathbb{X} = [-1, 1]$, initial set $\mathbb{X}_0 = [-0.5, 0.5]$, and unsafe regions $\mathbb{X}_U = [-1, -0.9] \cup [0.9, 1]$. While going

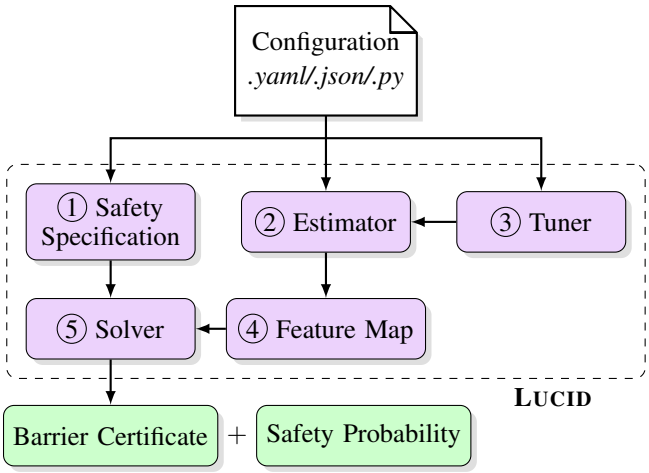

Figure 5: General architecture of LUCID, highlighting its core components and their connections.

over each component of LUCID, we will show how they can be configured within a *.yaml* configuration file to capture the system (7) and its safety specification.

Dataset Due to its data-driven nature, all that LUCID needs to operate is a set of sampled transitions $\mathcal{D}_N$ from the system, as shown in (2), and the safety specification itself. The samples can be specified directly in the configuration file, divided into $x^1, \ldots, x^N$ and $x_+^1, \ldots, x_+^N$:

```
x_samples: [[5.930e-01], ..., [-1.937e-01]]
xp_samples: [[3.015e-01], ..., [-1.018e-01]]
```

Alternatively, the samples can be either generated by the Python configuration script or stored in a separate file.

① Safety Specification LUCID understands sets $\mathbb{X}, \mathbb{X}_0, \mathbb{X}_U$ specified as `RectSet`, `SphereSet`, or `Multiset` objects (collections of sets from the first two categories). They can be included in the configuration as shown below:

```
X_bounds: "RectSet([-1], [1])"
X_init: "RectSet([-0.5], [0.5])"
X_unsafe: ["RectSet([-1], [-0.9])",
           "RectSet([0.9], [1])"]
```

② Estimator The core of LUCID is its `Estimator`, namely the `KernelRidgeRegressor`, which uses the `GaussianKernel` to learn the underlying system dynamics by estimating the CME from the samples. Its predictions of the expected next state x_+ are used to determine the constraints for the CBC LP. The setup is configured as follows:

```
kernel: "GaussianKernel"
estimator: "KernelRidgeRegressor"
```

③ Tuner Being parameter-free approaches, kernel methods do not require the expensive learning processes of other machine learning methods, such as neural networks. However, they still depend on a number of hyperparameters, such as the kernel bandwidth σ_f, the lengthscale σ_l, and the regularization constant λ (see (4)–(5)). Changes in their val-

ues can have significant impact on the `Estimator`'s efficiency and accuracy. The process of finding good values for these hyperparameters is known as *hyperparameter tuning*, with the optimal parameters being problem dependent. LUCID provides a set of utilities, which specialize the `Tuner` interface, to aid in this task:

- `MedianHeuristicTuner` uses the *median heuristic* (Garreau, Jitkrittum, and Kanagawa 2018) to produce rule-of-thumb-type estimates for the hyperparameters σ_f and σ_l of, e.g., the Gaussian kernel (4), via closed-form expressions. It can be useful as a starting point for subsequent improvements.
- `LbfgsTuner` finds the hyperparameters that maximize the *log marginal likelihood* (Rasmussen and Williams 2006), defined as

$$\log p(X_N^+|X_N,\theta) = -\frac{1}{2}X_N^{+\top}(K_X^N + N\lambda I_N)^{-1}X_N^+ - \frac{1}{2}\log|K_X^N + N\lambda I_N| - \frac{N}{2}\log(2\pi),$$

  where $\theta := (\sigma_f, \sigma_l, \lambda)$ is the hyperparameterization being optimized. We use the L-BFGS or L-BFGS-B quasi-Newton optimization algorithms (Fletcher 2000), implemented in the `LBFGS++` library.
- `GridSearchTuner` implements the grid search method, exploring the space of possible hyperparameter values to maximize the `Estimator`'s R^2 score,

$$R^2 = 1 - \left(\sum_{i=1}^M (y_i - \hat{y}_i)^2 / \sum_{i=1}^M (y_i - \bar{y})^2\right),$$

  with $\hat{y}_i$ and $\bar{y} := (\sum_{i=1}^N y_i)/N$ being the i^{th} predicted and mean observed outputs, respectively.

Tuners open a wide range of possibilities to the user. We recommend using them within a Python script generating a configuration to automate the tuning process before finalizing the `Estimator`'s hyperparameter values:

```
def scenario_config(c: Configuration):
    t = LbfgsTuner(lb=[1e-5], ub=[1e5])
    c.estimator = KernelRidgeRegressor()
    c.estimator.fit(c.x_samples, c.
    xp_samples, tuner=t)
    return c
```

In this example, we set the hyperparameters explicitly:

```
sigma_l: 0.0446
lambda: 1.0e-5
set_scaling: 0.04
```

The parameter `set_scaling` can be used to lower the constraint-tightening ϵ (see Section 2.4.2.) by increasing the size of the sets $\mathbb{X}, \mathbb{X}_0, \mathbb{X}_U$; here, for example, by 4%.

④ Feature Map We exploit the spectral kernel expansion in (6) to construct an explicit approximated feature map, composed of trigonometric functions with increasing frequencies (see Figure 3). The underlying kernel expansion, visualized in Figure 4, can be controlled to trade-off efficiency and accuracy/conservativeness. After selecting its hyperparameters (σ_f, σ_l), the feature map can be used to map any point from $\mathbb{X}$ to the RKHS associated with the kernel. To this end, LUCID partitions the kernel's spectral measure: `LinearTruncatedFourierFeatureMap` defines equally spaced frequency bands on the interval $[-3\sigma_l, 3\sigma_l]$, capturing 99.73% of the spectral measure. This partitioning is shown in Figure 4. Users may define custom feature maps to produce other partitions.

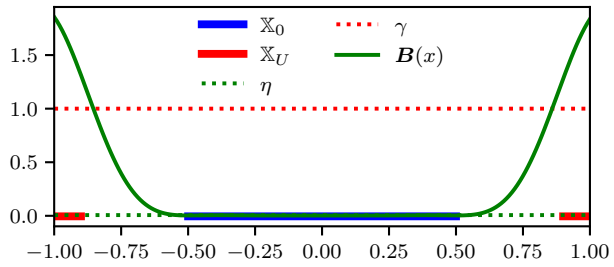

Figure 6: Barrier certificate for the running example. The value of the barrier $\boldsymbol{B}$ is plotted against the state space $\mathbb{X}$. The green line represents the barrier value. The initial and unsafe sets are highlighted in blue and red, respectively.

In our example, we truncate the Fourier expansion to 6 frequency bands, including the constant term, and use 300 lattice points to build the constraints for the optimization.

```
num_frequencies: 6
lattice_resolution: 300
feature_sigma_l: 0.0925
```

⑤ Solver From the components discussed above, LUCID generates and solves a finite-constraint LP as outlined in Section 2.4. If successful, LUCID returns a CBC, a quantified lower bound on the true safety probability $P_{\text{safe}}(\boldsymbol{\mathcal{S}}^\pi)$, as well as the corresponding constants η and c.

Generating the LP, specifically choosing an appropriate lattice density, involves trading off efficiency versus conservativeness, with any sampling density beyond the Nyquist frequency yielding a valid relaxation. In the benchmarks in Section 4 we showcase the relation numerically. For solving the LP, LUCID can interface with different linear optimizers: GUROBI, ALGLIB, or HIGHS. Here, we use GUROBI:

```
optimiser: "GurobiOptimiser"
```

Running the Tool We have configured all the components needed to run LUCID. Putting it all together in a single configuration file, we can run the tool with the command `pylucid config.yaml --plot`. LUCID will parse the configuration, synthesize a barrier certificate, and plot the result. For the running example, we obtain the barrier certificate shown in Figure 6 for a time horizon of $T = 15$, certifying safety of the system in (7) with a probability of at least 93.07% ($\eta = 0.006, c = 0.004$).

Validation (optional) Albeit barriers generated by LUCID are by design formally sound with respect to the data-driven estimator, if the latent system dynamics are known, LUCID provides the option to formally verify the correctness of the resulting barrier using the DREAL SMT solver (Gao, Kong, and Clarke 2013). Since we know the expected behavior of

the system $\mathcal{S}$ from (7), we confirm that $\boldsymbol{B}$ is indeed a valid CBC by adding the following lines:

```
system_dynamics: ["x1 / 2"]
verify: true
```

4 Experimental Evaluation

To ascertain the performance of LUCID, we conduct a series of experiments on a Windows 10 machine with an AMD Ryzen 9 5950X 16-Core Processor @ 3.40 GHz, NVIDIA GeForce RTX 3090 GPU, and 64 GB of RAM. All runs had the random seed set to 42 to ensure reproducibility.

We adapt the `Barr2` and `Barr3` benchmarks from Abate et al. (2021). Both are two-dimensional highly nonlinear systems to which we add stochastic noise $w_t \sim \mathcal{N}(\cdot|0, 0.01I_2)$. Given $N = 1000$ samples, initial set $\mathbb{X}_0$, and unsafe set $\mathbb{X}_U$, we synthesize a barrier $\boldsymbol{B}$ that guarantees trajectories starting in $\mathbb{X}_0$ do not enter $\mathbb{X}_U$ within $T = 5$ time steps. Note that here we only certify safety w.r.t. the empirical distribution, i.e., the CME constructed from the observed data. The synthesized barriers are shown in Figures 7–8. We also consider a new benchmark `Over`, where an autonomous vehicle controlled by a NN is overtaking another vehicle. The dynamics of the ego vehicle are given by Dubin's car model with an added noise vector w where each component is drawn from a zero-mean Gaussian with standard deviation 0.01, 0.01, and 0.001 respectively. The steering wheel angle is supplied by the NN controller and we travel at a fixed velocity.

Table 2 summarizes the results of all the experiments presented. As a point-wise baseline, we estimate the safety probability at selected initial states via Monte Carlo simulation, yielding approximately 95–100% safety in the reported benchmarks. Note that these estimates do not extend to set-wise guarantees. Further details on the experiments can be found in the extended version (Casablanca et al. 2025).

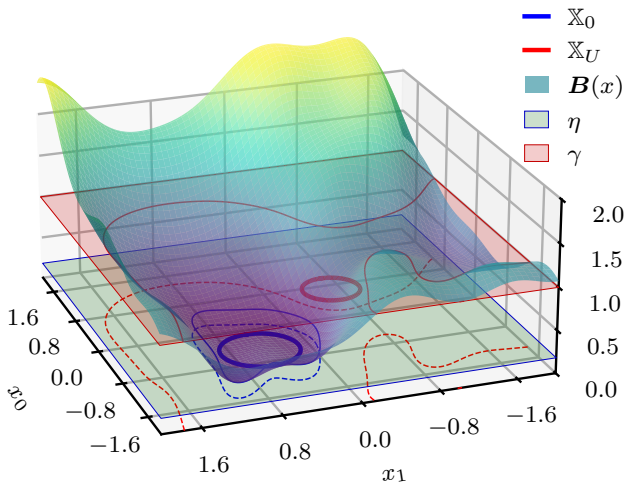

Figure 7: CBC synthesized for the `Barr2` benchmark.

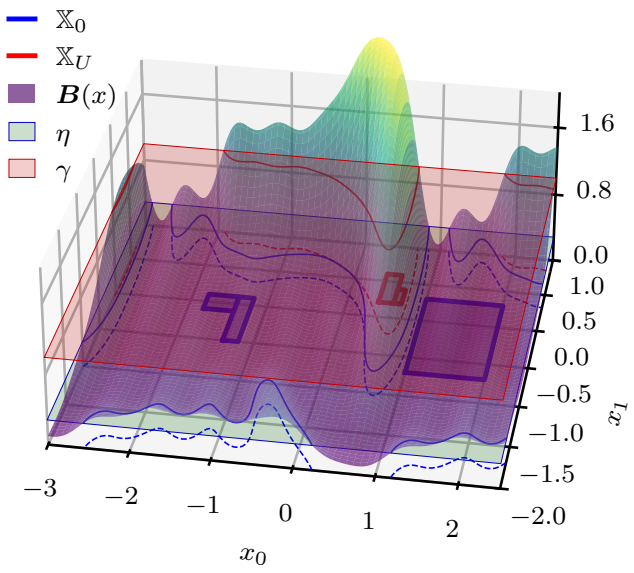

Figure 8: CBC synthesized for the `Barr3` benchmark.

	T	#Freq.	Lattice Size	Runtime [mm:ss]	Safety Prob.
`Linear`	15	6	300^1	00:01	93.07%
`Barr2`	5	7	330^2	04:38	63.17%
`Barr3`	5	7	330^2	03:20	51.63%
`Over`	5	5	70^3	93:56	53.66%

Table 2: Computational benchmarks: The lattice size indicates the number of points of the lattice in each dimension. T is the time horizon associated with the lower bound on the safety probability, displayed in the last column.

5 Conclusion and Future Extensions

This paper introduces LUCID, a novel tool capable of quantifying safety guarantees for black-box systems with complex stochastic dynamics and deep learning components in the loop. As such, LUCID fills a gap currently unaddressed by preexisting tools. To achieve this, LUCID leverages Conditional Mean Embeddings (CME) to learn control barrier certificates from data and recasts the problem into a tractable linear form via a spectral abstraction.

LUCID is built for extensibility. In its current form, it assumes access to full state measurements (see (2)), as is the case when working with simulated systems, and focuses on verification, i.e., when a policy is given. Extending LUCID to partially observed settings and safe controller synthesis is on our agenda and builds on top of the results presented in this paper. LUCID derives barriers based on the empirical CME, which can be robustified against out-of-sample system behavior by tightening the barrier constraint (c) (see Sections 2.3–2.4). The scalability of LUCID can be improved by performing sparse CME computations or specializing the estimator's `Kernel` to embed prior knowledge about the system dynamics.

Acknowledgments

Ernesto Casablanca is supported by the Engineering and Physical Sciences Research Council (EPSRC), grant number EP/W524700/1. Paolo Zuliani is supported by the project SERICS (PE00000014) under the Italian MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. The work of Sadegh Soudjani is supported by the EIC SymAware project 101070802 and the ERC Auto-CyPheR project 101089047.

References

Abate, A.; Ahmed, D.; Edwards, A.; Giacobbe, M.; and Peruffo, A. 2021. FOSSIL: A Software Tool for the Formal Synthesis of Lyapunov Functions and Barrier Certificates using Neural Networks. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 1–11.

Abate, A.; Althoff, M.; Bu, L.; Ernst, G.; Frehse, G.; Geretti, L.; Johnson, T. T.; Menghi, C.; Mitsch, S.; Schupp, S.; et al. 2024. The ARCH-COMP Friendly Verification Competition for Continuous and Hybrid Systems. In *International TOOLympics Challenge*, 1–37. Springer.

Abate, A.; Prandini, M.; Lygeros, J.; and Sastry, S. 2008. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11): 2724–2734.

Anand, M.; Murali, V.; Trivedi, A.; and Zamani, M. 2022. k-Inductive Barrier Certificates for Stochastic Systems. In *Proceedings of the 25th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '22. New York, NY, USA: Association for Computing Machinery. ISBN 9781450391962.

Casablanca, E.; Schön, O.; Zuliani, P.; and Soudjani, S. 2025. LUCID: Learning-Enabled Uncertainty-Aware Certification of Stochastic Dynamical Systems (Extended Version). *arXiv*.

Chen, Y.; Li, Y.; Li, S.; and Yin, X. 2025. Distributionally Robust Control Synthesis for Stochastic Systems with Safety and Reach-Avoid Specifications. *arXiv preprint arXiv:2501.03137*.

Corso, A.; Moss, R.; Koren, M.; Lee, R.; and Kochenderfer, M. 2021. A survey of algorithms for black-box safety validation of cyber-physical systems. *Journal of Artificial Intelligence Research*, 72: 377–428.

Cosner, R. K.; Sadalski, I.; Woo, J. K.; Culbertson, P.; and Ames, A. D. 2024. Generative modeling of residuals for real-time risk-sensitive safety with discrete-time control barrier functions. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, i–viii. IEEE.

Edwards, A.; Peruffo, A.; and Abate, A. 2024. Fossil 2.0: Formal Certificate Synthesis for the Verification and Control of Dynamical Models. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, 1–10.

Fletcher, R. 2000. *Nonlinear Programming*. John Wiley & Sons, Ltd. ISBN 9781118723203.

Gao, S.; Kong, S.; and Clarke, E. M. 2013. dReal: An SMT Solver for Nonlinear Theories over the Reals. In Bonacina, M. P., ed., *Automated Deduction – CADE-24*, 208–214. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-38574-2.

Gardner, J.; Wooding, B.; Nejati, A.; and Lavaei, A. 2025. TRUST: StabiliTy and Safety ContRoller Synthesis for Unknown Dynamical Models Using a Single Trajectory. In *Proceedings of the 28th ACM International Conference on Hybrid Systems: Computation and Control*, 1–16.

Garreau, D.; Jitkrittum, W.; and Kanagawa, M. 2018. Large sample analysis of the median heuristic. arXiv:1707.07269.

Harapanahalli, A.; Jafarpour, S.; and Coogan, S. 2023. Forward invariance in neural network controlled systems. *IEEE Control Systems Letters*, 7: 3962–3967.

Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N. J.; Kern, R.; Picus, M.; Hoyer, S.; van Kerkwijk, M. H.; Brett, M.; Haldane, A.; del Río, J. F.; Wiebe, M.; Peterson, P.; Gérard-Marchant, P.; Sheppard, K.; Reddy, T.; Weckesser, W.; Abbasi, H.; Gohlke, C.; and Oliphant, T. E. 2020. Array programming with NumPy. *Nature*, 585(7825): 357–362.

Ji, J.; Zhou, J.; Zhang, B.; Dai, J.; Pan, X.; Sun, R.; Huang, W.; Geng, Y.; Liu, M.; and Yang, Y. 2024. Omnisafe: An infrastructure for accelerating safe reinforcement learning research. *Journal of Machine Learning Research*, 25(285): 1–6.

Kazemi, M.; and Soudjani, S. 2020. Formal policy synthesis for continuous-state systems via reinforcement learning. In *Integrated Formal Methods: 16th International Conference (IFM '20)*, 3–21. Springer.

Kuhn, D.; Shafiee, S.; and Wiesemann, W. 2025. Distributionally robust optimization. *Acta Numerica*, 34: 579–804.

Kushner, H. J. 1967. *Stochastic stability and control*, volume 33. Academic Press New York.

Lew, T.; and Pavone, M. 2021. Sampling-based reachability analysis: A random set theory approach with adversarial sampling. In *Conference on Robot Learning*, 2055–2070. PMLR.

Li, Z.; Meunier, D.; Mollenhauer, M.; and Gretton, A. 2022. Optimal rates for regularized conditional mean embedding learning. *Advances in Neural Information Processing Systems*, 35: 4433–4445.

Lindemann, L.; Zhao, Y.; Yu, X.; Pappas, G. J.; and Deshmukh, J. V. 2024. Formal verification and control with conformal prediction. *arXiv preprint arXiv:2409.00536*.

Liu, C.; Arnon, T.; Lazarus, C.; Strong, C.; Barrett, C.; Kochenderfer, M. J.; et al. 2021. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4(3-4): 244–404.

Muandet, K.; Fukumizu, K.; Sriperumbudur, B.; and Schölkopf, B. 2017. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1–2): 1–141.

Park, J.; and Muandet, K. 2020. A measure-theoretic approach to kernel conditional mean embeddings. *Advances in Neural Information Processing Systems*, 33: 21247–21259.

Pfister, L.; and Bresler, Y. 2018. Bounding multivariate trigonometric polynomials. *IEEE Transactions on Signal Processing*, 67(3): 700–707.

Prajna, S. 2006. Barrier certificates for nonlinear model validation. *Automatica*, 42(1): 117–126.

Rahimi, A.; and Recht, B. 2007. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20.

Rasmussen, C. E.; and Williams, C. K. 2006. *Gaussian Processes for Machine Learning*. Springer.

Salamati, A.; Lavaei, A.; Soudjani, S.; and Zamani, M. 2024. Data-driven verification and synthesis of stochastic systems via barrier certificates. *Automatica*, 159(C): 111323.

Schölkopf, B.; and Smola, A. J. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Schön, O.; Zhong, Z.; and Soudjani, S. 2024. Data-Driven Distributionally Robust Safety Verification Using Barrier Certificates and Conditional Mean Embeddings. In *2024 American Control Conference (ACC)*, 3417–3423.

Schön, O.; Zhong, Z.; and Soudjani, S. 2025. Kernel-Based Learning of Safety Barriers. *arXiv preprint*.

Smola, A.; Gretton, A.; Song, L.; and Schölkopf, B. 2007. A Hilbert space embedding for distributions. In Hutter, M.; Servedio, R. A.; and Takimoto, E., eds., *Algorithmic Learning Theory*, 13–31. Springer.

Steinwart, I.; and Christmann, A. 2008. *Support Vector Machines*. Springer. ISBN 0387772413.

Veale, M.; and Zuiderveen Borgesius, F. 2021. Demystifying the Draft EU Artificial Intelligence Act – Analysing the good, the bad, and the unclear elements of the proposed approach. *Computer Law Review International*, 22(4): 97–112.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

Wongpiromsarn, T.; Ghasemi, M.; Cubuktepe, M.; Bakirtzis, G.; Carr, S.; Karabag, M. O.; Neary, C.; Gohari, P.; and Topcu, U. 2023. Formal Methods for Autonomous Systems. *arXiv preprint arXiv:2311.01258*.

Zhang, T. T.; Lee, B. D.; Ziemann, I.; Pappas, G. J.; and Matni, N. 2024. Guarantees for nonlinear representation learning: Non-identical covariates, dependent data, fewer samples. In *Proceedings of the 41st International Conference on Machine Learning*, 59126–59147.

Ziemann, I.; Tsiamis, A.; Lee, B.; Jedra, Y.; Matni, N.; and Pappas, G. J. 2023. A Tutorial on the Non-Asymptotic Theory of System Identification. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, 8921–8939.