VeriFlow: Modeling Distributions for Neural Network Verification

Faried Abu Zaid[1], **Daniel Neider**[2, 3], **Mustafa Yalçıner**[2, 3]

[1]Independent Researcher, Munich, Germany
[2]TU Dortmund University, Dortmund, Germany
[3]Center for Trustworthy Data Science and Security University Alliance Ruhr, Dortmund, Germany
fariedaz@gmail.com, daniel.neider@tu-dortmund.de, mustafa.yalciner@tu-dortmund.de

Abstract

Formal verification has emerged as a promising method to ensure the safety and reliability of neural networks. However, many relevant properties, such as fairness or global robustness, pertain to the entire input space. If one applies verification techniques naively, the neural network is checked even on inputs that do not occur in the real world and have no meaning. To tackle this shortcoming, we propose the VeriFlow architecture as a flow-based density model tailored to allow any verification approach to restrict its search to some data distribution of interest. We argue that our architecture is particularly well suited for this purpose because of two major properties. First, we show that the transformation that is defined by our model is piecewise affine. Therefore, the model allows the usage of verifiers based on constraint solving with linear arithmetic. Second, upper density level sets (UDL) of the data distribution are definable via linear constraints in the latent space. As a consequence, representations of UDLs specified by a given probability are effectively computable in the latent space. This property allows for effective verification with a fine-grained, probabilistically interpretable control of how (a-)typical the inputs subject to verification are.

Introduction

The outstanding performance of neural networks in tasks such as object detection (Zhao et al. 2019), image classification, anomaly detection, and natural language processing, made them a popular solution for many real-world-applications, including safety-critical ones. With the increasing popularity of neural networks, defects and limitations of these systems have been witnessed by the general public. The AI incident database[1] keeps track of harms and near-harms caused by AI-Systems in the real world.

Ideally, the safety and fairness properties of such inherently opaque neural networks should be formally guaranteed when used in safety-critical applications. As a solution, formal verification can be used to check whether a neural network satisfies a given safety property for the entire input space or whether there exists some (synthetic) input for which the desired property is violated. This is in contrast to the statistical testing methods classically employed in machine learning, where the output of the neural network is checked against a finite set of samples, usually from a held-out test set.

However, state-of-the-art formal verification methods are designed to verify either global or local properties. Global properties ensure a specific behavior of the neural network on the whole input space. As an example, fairness properties require the neural network to predict the same output for any two inputs that only differ in some sensitive attribute. Local properties, on the other hand, ensure a specific behavior of the neural network only in some part of the input space that is usually restricted using the training set. One well-studied example for a local property is *adversarial robustness*, which requires from the neural network for any point from the data set that any minor perturbation of that point does not significantly change the prediction (Szegedy et al. 2014). However, both global and local properties have shortcomings limiting their applicability. Local properties suffer from the same problem as statistical testing, i.e., they rely on a high-quality data set that the verification property is based on. Global properties refer to the entire input space, including regions that we may not need to verify, such as noise-inputs or regions of the input space for which there are only very few training samples available (epistemic uncertainty). In order to restrict global properties to some data distribution of interest, generative adversarial networks (GANs) (Goodfellow et al. 2014b) and variational autoencoders (VAEs) (Kingma and Welling 2014) can be used to verify the global property only in the high-density region of the data distribution. However, while these models are indeed able to capture the data distribution, they do not allow for probabilistic interpretability. For example, variational autoencoders are generally not designed to ensure that the low-density (high-density) data points from the training data set are mapped to the tail (center) of the latent space distribution.

To overcome this issue, we design a flow model tailored towards the application in neural network verification. We leverage our flow model to restrict the input space of the neural network under verification to the underlying data distribution. This allows restricting the verification of global properties to the data distribution of interest. In contrast to GANs and VAEs, our VeriFlow architecture does not only allow for efficient sampling but also provides probabilistic interpretability via tractable likelihoods (Papamakarios

[1]https://incidentdatabase.ai/

et al. 2021a; Goodfellow et al. 2014a; Rezende and Mohamed 2015; Dinh, Krueger, and Bengio 2015). This feature is unique to our flow architecture and important to facilitate fine-grained probabilistic control when restricting the input space to typical inputs during verification. This approach makes the verification property less reliant on the dataset, while still keeping the input space focused on meaningful data.

Summary[2] The goal of VeriFlow is to enable restricting the verification of a neural network to a probabilistically meaningful subset of the input space. While this approach no longer provides a formal safety guarantee over the entire input space, it instead offers a proof of safety over the likely inputs identified by a secondary neural network: our flow model. To this end, we propose a **bijective, invertible, and piecewise linear** flow model that **preserves monotonic relationship between latent and learned densities** by mapping high-density (and low-density) regions in the data distribution to high-density (and low-density) regions in the base distribution. Crucially, **high-density regions are defined in terms of ℓ_k-norms in the base distribution**, allowing us to express the high-density data region as a set of linear constraints. These constraints, in turn, can be incorporated into the specification of downstream verification tasks. In the following two sections, we formally introduce all bolded properties and sketch the VeriFlow architecture. **In our experiments**, we present a novel notion of global robustness, and evaluate VeriFlow's learning ability.

Preliminaries

Verifying neural networks involves checking if a network f satisfies a semantic property P, often expressed as $\phi(x) \implies \psi(f(x))$, where ϕ and ψ are pre- and postconditions. Two conceptually different verification approaches are relevant for this paper: *constraint-based verification* and *abstract interpretation*. We briefly explain the core idea of each verification approach. In **constraint-based verification**, the network f and property P are translated into a logical formula $\Psi_{f,P}$, whose validity implies that f satisfies P. Verifying $\Psi_{f,P}$ involves checking the unsatisfiability of $\neg\Psi_{f,P}$. If $\neg\Psi_{f,P}$ is satisfiable, a counterexample exists witnessing a violation of p by f. Efficient SMT solvers like Marabou 2 (Wu et al. 2024a) handle these formulas for networks with piece wise linear components such as ReLU activations. Constraint-based verification methods are *complete* and always provide counterexamples when the property is violated, though their runtime can be high for positive proofs. **Abstract Interpretation** symbolically executes a neural network using geometric abstract domains, such as zonotopes or star sets, which over-approximate input sets. By propagating these through the network, the procedure over-approximates possible outputs. Verification succeeds if the output lies entirely within the "safe space" defined by the semantic property ψ. If outputs are outside, the property fails; partial overlap leaves the result inconclusive due to the over-approximation in the process.

[2] A complete version of this work is available on arXiv and contains additional experiments and all proofs (arXiv:2406.14265).

Flow-based models are a class of neural networks designed to learn complex data distributions, supporting both sample generation and density estimation. Specifically, these models learn a bijective, continuously differentiable transformation — known as a *diffeomorphism* — that maps a simple base distribution B with probability density function p_B to a complex data distribution D with probability density function p_D. In our setting, we define the flow in the direction from the base distribution to the data distribution, that is, from latent to observed variables. Let F denote the learned transformation, implemented as an invertible neural network. The invertibility of F allows us to leverage the model for two complementary tasks: 1. **Sampling:** Draw a sample $z \sim B$, then apply the forward map to obtain $x = F(z)$. The result, x, is a new synthetic sample from the learned data distribution. 2. **Density estimation:** Given a data point x, compute its likelihood under the model using the change-of–variables formula: $p_D(x) = \left|\det \frac{\partial F^{-1}}{\partial x^T}\right| \cdot p_B(F^{-1}(x))$.

If the determinant of the $\det \frac{\partial F^{-1}}{\partial x^T}$ is constant, the we say that the flow is **uniformly scaling**.

Base Distributions

The most commonly used base distribution is the *isotropic Gaussian*, a multivariate normal distribution with zero mean and identity covariance matrix. When a flow model uses this distribution as its base, it is referred to as a *normalizing flow*. While the Gaussian distribution is a natural choice, it is by no means the only option. In fact, the Knothe-Rosenblatt Rearrangement Theorem (Knothe 1957; Rosenblatt 1952) guarantees that for any two absolutely continuous probability distributions, there exists a smooth, invertible mapping (a *diffeomorphism*) that transforms one distribution into the other. This foundational result justifies using alternative base distributions: the expressive power of flow models is theoretically unaffected by the choice, as long as the flow is flexible enough.

Alternative base distributions relevant to us are those that are *k-radial monotonic*, where the probability density depends solely on the ℓ_k-norm of the input:

Definition 1 (Radial Distributions and Radial Profile) *Let $k \in \mathbb{N}_{>0} \cup \{\infty\}$, and let X be a random variable in $\mathbb{R}^d$ with density function $p(x)$. We say that X is* k-radially distributed *if there exists a function $g : \mathbb{R}_+ \to \mathbb{R}_+$ such that $p(x) = g(|x|_k)$, where $|x|_k$ denotes the ℓ_k-norm of x. The function g is called the **radial profile** of X. If g is strictly decreasing, then X is called* k-radial monotonic.

Intuitively, a k-radial distribution is one where all points at the same ℓ_k-distance from the origin have the same density. If the density decreases as the norm increases, the distribution is *radially monotonic*. This includes well-known distributions like the Gaussian (for $k = 2$) and the Laplace distribution (for $k = 1$). Importantly, k-radial distributions are completely determined by the distribution of the ℓ_k-norm. Here, the norm distribution w.r.t. a random variable X refers to the distribution of $|X|_k$. The following definition makes this explicit by showing how to construct a k-radial distribution from a one-dimensional norm distribution:

Definition 2 (k-Radial Distribution Construction) *Let $\rho : \mathbb{R}_+ \to \mathbb{R}_+$ be a one-dimensional probability density function, and let $k \in \mathbb{N}_{>0} \cup \{\infty\}$. The profile of the* k-radial distribution *in d-dimensional space with norm distribution ρ is given by $g(r) = \rho(r) \left(\frac{\partial V_k^d(r)}{\partial r}\right)^{-1}$, where $V_k^d(r)$ denotes the volume of the ℓ_k-ball of radius r in $\mathbb{R}^d$.*

Here, the derivative $\frac{\partial V_k^d(r)}{\partial r}$ captures the rate at which the volume increases as we move outward in norm, i.e. $g(r)$ describes the density of the corresponding radial distribution at points x with $|x|_k = r$. The formula ensures that when we integrate the density over spherical shells, i.e., sets of points with the same norm, we recover the desired norm distribution ρ.

Although the above construction enables precise density estimation for every point via the norm distribution, density estimators are often used more coarsely — such as in our work — to determine whether a point lies in a high- or low-density region. To formalize this, we define upper and lower density level sets, which represent regions containing high-density samples (center of the distribution) and low-density samples (tail of the distribution), respectively.

Definition 3 *Given the density of the input distribution $p_D : \mathbb{R}^d \to \mathbb{R}_+$, the set of points whose density exceeds a given threshold t is called the upper density level set (UDL) and is defined as $L_D^{\uparrow}(t) \coloneqq \{x \in \mathbb{R}^d \mid p_D(x) > t\}$. Conversely, the lower density level set (LDL) contains the set of points subceeding the density threshold: $L_D^{\downarrow}(t) \coloneqq \{x \in \mathbb{R}^d \mid p_D(x) \leq t\} = \mathbb{R}^d \setminus L_D^{\uparrow}$. If for $q \in [0,1]$ there is a UDL of D with probability q, then we write $\mathrm{UDL}_D(q) \coloneqq \inf_t\{L_D^{\uparrow}(t) \mid P_D(L_D^{\uparrow}(t)) = q\}$ to denote the minimal UDL of probability q.*

Note that the existence $\mathrm{UDL}_D(q)$ for all $q \in [0,1]$ is guaranteed if $P_D(\{x \mid p(x)_D = t\}) = 0$ for all $t > 0$, where P_D denotes the probability induced by p_D, defined as $P(x \in S) \coloneqq \int_S p_D(x)dx$.

Crucial for our application is the fact that density level sets of k-radial distributions can be described in terms of quantiles of the radial profile. Clearly, each density-level set is the union of k-spherical shells, i.e. $\mathrm{UDL}_X(q) = \{x \mid |x|_k \in R\}$ for some $R \subseteq \mathbb{R}^+$. Therefore, in order to describe $\mathrm{UDL}_X(q)$, it is sufficient to describe its norm projection $R = \{|x|_k \mid x \in \mathrm{UDL}_X(q)\}$, which has a very simple box shape for radial monotonic distributions:

Observation 1 *Let X be a k-radially distributed random variable with radial profile g. Then $\{|x|_k \mid x \in \mathit{UDL}_X(q)\} = \{r \mid g(r) > t\}$, where $t = \mathit{quantile}_{g(|X|_k)}(1-q)$. Moreover, if X is radial monotonic, then $\{|x|_k \mid x \in \mathit{UDL}_X(q)\} = [0, \mathit{quantile}_{|X|_k}(q))$.*

Piecewise Affine Flows

A function $f : \mathcal{X} \to \mathcal{Y}$ is piecewise affine, if there exists a partition of the domain $\mathcal{X} = \mathcal{X}_1 \cup \cdots \cup \mathcal{X}_n$ such that f restricted to $\mathcal{X}_i$ is affine for every i. We call $\mathcal{X}_1, \ldots, \mathcal{X}_n$ affine regions of f. As we argued earlier, piecewise affinity is crucial for efficient SMT-based verification. Therefore, our flow transformation should be piecewise affine, which we can achieve e.g., by using ReLU networks. If we can ensure that the defined function is bijective, then we obtain a continuous piecewise affine bijection where the affine regions can be represented as intersections of open and closed half-spaces (Moser et al. 2022). Hence, the regions are contained in the Borel algebra $\mathcal{B}(\mathbb{R}^d)$.

Proposition 1 asserts that the change of variables formula remains valid for piecewise affine bijections (instead of diffeomorphisms), provided that the affine regions are Borel sets.

Proposition 1 *Let $F : \mathbb{R}^d \to \mathbb{R}^d$ be a piecewise affine bijection, with affine regions $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathcal{B}(\mathbb{R})^d$ and let B be an absolutely continuous random variable. Then, $p_{F(B)}(x) = p_B(F^{-1}(x)) \left|\det \frac{\partial F^{-1}}{\partial x}\right|$, where the Jacobian of F^{-1} is evaluated piecewise, according to the affine regions.*

VeriFlow: A Density-Preserving Architecture

In this section, we build upon the preliminary results and present the novel property of VeriFlow: the preservation of density level sets between the data distribution and learned distribution. This relationship distinguishes uniformly scaling flows from most other flow classes and justifies using the density level sets of the base distribution as a presentation for those of the data distribution. Specifically, Proposition 2 shows that uniformly scaling flows preserve a monotonic relationship between latent and learned densities:

Proposition 2 (Density Level Set Preservation) *Let F be a uniformly scaling flow with base distribution B. Then F maps upper density level sets of the base distribution to upper density level sets of the data distribution: $\mathrm{UDL}_{F(B)}(q) = F(\mathrm{UDL}_B(q))$, where $F(B)$ is the random variable defined by F with base distribution B and $F(\mathrm{UDL}_B(q))$ is the image of $\mathrm{UDL}_B(q)$ under F.*

Combining Proposition 2 and Observation 1 yields that uniformly scaling flows with radial base distributions (w.r.t ℓ_1 or ℓ_∞ norm) allow for simple density level set descriptions via linear constraints in latent space:

Corollary 1 *Let F be a uniformly scaling flow and B a k-radial base distribution with radial profile g. Then $\{|F^{-1}(x)|_k \mid x \in \mathit{UDL}_{F(B)}(q)\} = \{r \mid g(r) \geq t\}$ where $t = \mathit{quantile}_{g(|B|_k)}(1-q)$. Moreover, if B is radial monotonic, then $\{|F^{-1}(x)|_k \mid x \in \mathit{UDL}_{F(B)}(q)\} = [0, \mathit{quantile}_{|B|_k}(q))$.*

In particular, for a radial base distribution w.r.t. the 1- or the ∞-norm, density level sets of the learned distribution can be defined in latent space using linear constraints.

The Components of VeriFlow To identify learnable building blocks that satisfy our aforementioned requirements, we survey established flow architectures. A central finding is that additive conditioning layers with ReLU activations, such as additive coupling, yield precisely the kind of networks we seek. These include additive coupling layers (Dinh, Krueger, and Bengio 2015), additive

auto-regressive layers (Kingma et al. 2017; Papamakarios, Pavlakou, and Murray 2017), and masked additive convolutional flows such as MACow (Ma et al. 2019b). Furthermore, bijective affine transformations parameterized by LU-decomposed matrices, named LUNets (Chan, Penquitt, and Gottschalk 2023), are also suitable. LU layers constitute a powerful replacement for the permutations or masks that were originally proposed to be combined with coupling-like layers. Although this addition turned out to be very beneficial, we note that the number of parameters scales quadratically with the dimension, which is a bottleneck in high dimensions. The poor scalability of vanilla LU-layers led us to propose one-star-convolutions — convolutions with kernel size 1 along all spatial dimensions, adapted to input topology (e.g., 1D for sequences, 2D for images). We apply one-star-convolutions together with LU-decomposed bijective channel transforms, providing a natural way to define a LU-decomposed bijective affine transformations on the entire set of input components via parameter sharing (Kingma and Dhariwal 2018). An in-depth analysis of the above mentioned layers can be found in the supplementary material.

Finally, we propose the VeriFlow architecture. We build VeriFlow from blocks of shape $B_i = A_i^{-1} \circ C_i \circ A_i$, where A_i is an affine transform given by a one-star-convolution with LU decomposed bijective channel transform and C is an additive coupling layer. This shape is motivated by the above analysis and results from the normalizing flow literature showing that applying an adjoint group action with respect to the subgroup of affine transforms of the group of diffeomorphisms over $\mathbb{R}^d$ to a coupling layer is more beneficial than simply alternating coupling and affine transforms (Hoogeboom, van den Berg, and Welling 2019; Ma et al. 2019a). The complete flow is of shape $F = A_{n+1} \circ B_n \circ B_{n-1} \circ \cdots \circ B_1$, where the final affine transform ensures that the flow can have an arbitrary (constant) Jacobian determinant. Note that each block is volume preserving since determinants of A_i and A_i^{-1} cancel each other, and additive coupling layers are volume preserving by design (Dinh, Krueger, and Bengio 2015). In conclusion, we obtain the following results for our architecture.

Observation 2 (VeriFlow is Uniformly Scaling)

Observation 3 (VeriFlow is Piecewise Affine)

The Neuro-Symbolic Verification Pipeline

The key idea of the *neuro-symbolic verification* framework as proposed by Xie, Kersting, and Neider (2022) is to use neural networks as part of the specification to enable the verification of a variety of complex properties. Our approach is aligned with the Neuro-Symbolic verification framework as we enable the verification of neural network properties over the UDLs or LDLs of proxy distributions learned by VeriFlow. The UDL in latent space captures likely inputs under the base distribution, while the UDL in the target (data) space captures typical inputs resembling those in the dataset. Verifying properties within a UDL thus proves/falsifies that the property holds for all *likely* inputs.

The verification pipeline proceeds as follows. First, we define a density level set in the latent space using linear constraints (Corollary 1). These constraints are transformed through the flow model into output constraints that precisely characterize the corresponding upper density level set in the data space, when a constraint-based verification tool is employed (Proposition 2). This is enabled by the piecewise linearity of the flow model (Proposition 1). In contrast, abstract interpretation uses the input bounds to compute an over-approximation of the output region, yielding a sound but conservative approximation of the same density level set. We illustrate this pipeline using an abstract interpretation-based verifier in Figure 1.

To ensure their reliability in formal verification settings, we carefully evaluate and validate the trained flow models.

Validation and Calibration Capturing the true input distribution by the flow as closely as possible is crucial for meaningful verification results. Therefore, assessing the deviation between true and learned distribution empirically is a natural part of the verification pipeline. For normalizing flows, typical assessment methods exploit that the maximum likelihood objective of the flow training is equivalent to minimizing the KL divergence between the empirical and the defined base distribution. One can therefore measure discrepancies in the latent space (Papamakarios et al. 2021b; Linhart, Gramfort, and Rodrigues 2023). Here, we use the term empirical base distribution to denote the pre-image of the true data distribution under the flow.

Since we work with radial base distributions and are focused on density level sets, we test for discrepancies in norm distribution and radiality separately. More precisely, we propose the following measures to assess the quality of the empirical latent norm distribution and the defined norm distribution in the latent space: (i) KS Statistic which quantifies the maximum difference between two cumulative distribution functions: $\sup_{r \in \mathbb{R}^+} |\mathrm{CDF}_{|F^{-1}(X)|}(r) - \mathrm{CDF}_{|B|}(r)|$, (ii) and kernel density estimates (KDEs) and probability–probability (PP) plots using a consistent non-parametric estimator. KDEs visualize the density of the transformed and base distributions, while PP-plots compare their empirical quantiles by plotting their cumulative distributions against each other. These measures give us quantitative information about the maximal and the local deviation between the learned and the true distribution, measured in probability. Since we project into the one-dimensional norm-space, we can obtain reliable estimates in a sample-efficient way. The tests regarding the norm-distribution are complemented by two radiality tests: (i) We assess sign symmetry using a binomial test — that is, whether positive and negative signs in the latent dimensions appear with equal frequency, as would be expected in a truly symmetric radial distribution, (ii) We apply an energy distance test to the normalized absolute values of the latent vectors to assess whether these directions are uniformly distributed over the simplex (projection to the positive part of ℓ_1 sphere), another key property of a radial distribution (Dodge 2008; Rizzo and Székely 2016). All tests provide p-values, which we can combine into a single rejection criterion by applying Bonferroni correction (Dunn 1961). The separate assessment of norm distribution and radiality allows us to distinguish different types of

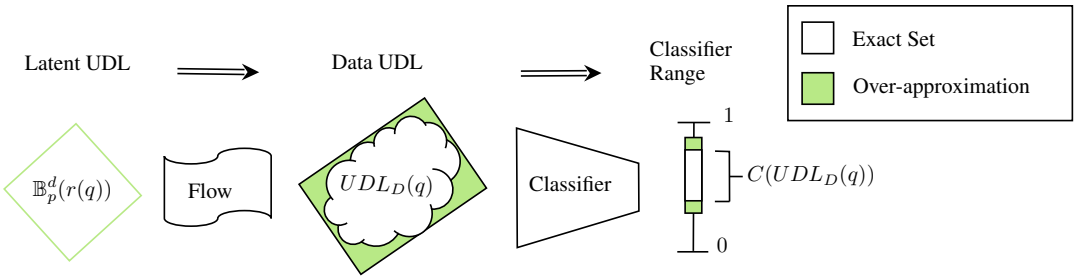

Figure 1: Visualization of a verification pipeline for an in-distribution verification of a classifier using VeriFlow. The procedure starts by defining the UDL exactly in the latent space. The true classification range w.r.t. the UDL equals the result of pushing the set consecutively through the flow and the classifier. An over-approximation can be obtained via abstract interpretation.

discrepancies. For instance, a well-aligned empirical norm-distribution but with discrepancies in radiality indicates that the flow over-approximated the true distribution: A matching norm distribution indicates that the learned UDLs have approximately the correct probability under the true distribution. However, a failed radiality test suggests that some areas of the latent UDL encode OOD data (since we do not find corresponding data in the data set). Hence, the described scenario provides evidence that the learned UDLs are supersets of the true UDLs.

In case that moderate discrepancies in the norm distribution have been identified, e.g. $0.05 > p > 0.0001$, we may use the norm distribution of the empirical base distribution for recalibration: We adjust the thresholds for the base distribution such that the returned density level set in the target distribution contains the desired fraction of the data.

Interpretability Recalibration of density level sets allows to maintain interpretability, even in the presence of discrepancies between the true and the learned distribution: Calibration against a test set D_{Cal} ensures that a density level set with target probability q will indeed contain a q-fraction of D_{Cal}. Hence, a successful verification will always establish a stronger statement than the empirical check $P_{x \sim D_{\text{Cal}}}(\phi(x)) \geq q$, since an infinite number of variants are verified along with the contained data.

Experiments

Implementation We implement the VeriFlow as an open-source Python library that can be trained on all datasets mentioned in this paper in an out-of-the-box fashion. Our library ensures that definable flows guarantee all the theoretical properties outlined in this paper and implements general radial distributions with learnable norm distribution for various norms. The resulting flow models can be exported in ONNX format, relying exclusively on ONNX node types that are supported by the VNN-LIB standard (Demarchi et al. 2023). Additionally, we provide evaluation methods for the previously outlined quality assessment of the flow.

Evaluation & Validation A crucial step before using VeriFlow in verification is to empirically check whether the data distribution was learned faithfully. To this end, we showcase our validation procedure on flows that were trained on MNIST digits, which are of particular interest for the verification experiments. As proposed earlier, we use KS Statisticand PP-Plots for validation. Results are shown in Figure 3 for all digits. Both validation methods show that the latent norm distribution of the data matches the defined latent norm distribution almost perfectly. For the KS statistic, we provide an example with the flow trained on digit 0 and confirm that the KS value of 0.0375 ($p = 0.152 \gg 0.05$) gives no statistically significant evidence for discrepancies. The other digits yield similar results, though not all directly meet the acceptance criterion. The radiality test, however, fails. Despite an average p-value of 0.0778, only about half of the dimensions pass the Bonferroni corrected sign symmetry test ($p > 0.05/(2 \cdot 784)$). Similarly, the projected uniformity test fails with p being numerically 0. We conclude that the learned UDLs contain the corresponding true data UDLs very well but are also likely to contain some OOD data (over approximation).

Robustness Verification

We demonstrate one use case of VeriFlow in the context of local robustness — a popular benchmark specification (Brix et al. 2024). Local robustness involves checking whether a neural network's prediction remains unchanged for minor perturbations. More formally, let $c : \mathbb{R}^n \to \mathbb{R}^m$ be a classifier, let $\mathcal{D} \subseteq \mathbb{R}^n$ be a dataset. The network is *locally robust* for a point $x \in \mathcal{D}$ w.r.t. a distance metric dist and radius $\epsilon > 0$, if the following specification is satisfied:

$$\phi \colon \left\{x, x' \in \mathbb{R}^n\right\}$$
$$y \leftarrow c(x) \quad y' \leftarrow c(x')$$
$$\psi \colon \left\{\text{dist}(x, x') \leq \epsilon \Rightarrow \arg\max_i y = \arg\max_i y'\right\}$$

This Hoare (1969) triple notation $\{\phi\}$ `assignments` $\{\psi\}$ corresponds to the logical form $\phi(x) \Rightarrow \psi(f(x))$ from the background section. Clearly, the runtime for verifying local robustness over the entire dataset increases linearly with its size $|\mathcal{D}|$.

To tackle this scalability problem, we demonstrate how to leverage our flow model to verify local robustness on multiple instances with only one verification run. To formalize this, let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a flow model with base distribution B and $c : \mathbb{R}^n \to \mathbb{R}^m$ be a classifier. Then, we propose

verifying the following global robustness specification:

$$\phi\colon \left\{x_\ell \in UDL_B(q), x' \in \mathbb{R}^n\right\}$$
$$x_t \leftarrow F(x_\ell) \quad y \leftarrow c(x_t) \quad y' \leftarrow c(x')$$
$$\psi\colon \left\{\text{dist}(x_t, x') \leq \epsilon \Rightarrow \arg\max_i y = \arg\max_i y'\right\}$$

where $q \in [0, 1]$ is a probability density threshold. By the design of our flow model, the data contained in the UDL, $\{F(x_\ell) \mid x_\ell \in UDL_B(q)\}$, includes $1 - q$ fraction of the dataset D. Intuitively, our specification is a new notion of *global robustness* as our formulation becomes independent of the dataset $\mathcal{D}$, but still keeps the input space restricted to meaningful inputs aligned with the data distribution $\mathcal{D}$. In other words, we verify robustness for all likely inputs. Therefore, a certificate produced for our global robustness specification also holds for the instance-wise local robustness specification for a $1 - q$ fraction of the dataset $\mathcal{D}$. Note that the inverse direction does not apply: a robustness certificate for some fraction of the dataset $\mathcal{D}$ does not imply global robustness as proposed in our specification due to the additional synthetic data that may be contained in our flow model.

In order to practically evaluate how the runtime of verifying global robustness compares to verifying local robustness, we conduct experiments on the MNIST dataset. Specifically, we aim to analyze after how many instance-wise verifications the overhead of integrating a flow model into the specification is paid off. To this end, we train our VeriFlow model $F : \mathbb{R}^{28\times28} \to \mathbb{R}^{28\times28}$ on digit-zero samples from the MNIST dataset. Our convolutional flow model has one coupling block with the same adjoint coupling architecture as presented in the main section and uses a 1-radial lognormal base distribution. Training is performed using the Adam optimizer. For classification, we employ a two-layer ReLU network with 91% test-accuracy defined as $c : \mathbb{R}^{28\times28} \to \mathbb{R}^{10}$. We note that VeriFlow places no restrictions on the classifier. We use a small classifier to ensure that our runtime measurements reflect the flow model rather than the classifier's complexity. We employ the SOTA verification tools Marabou 2 (Wu et al. 2024b), and the ABCrown framework's alpha-crown implementation (Xu et al. 2021) with branch-and-bound for complete verification. Specifically, we use (only) Marabou 2 to verify our global robustness specification since ABCrown does not support the type of specification proposed here (Alberti et al. 2025). To conduct the local robustness verification, we use both Marabou 2 and ABCrown. Specifically, we conduct two variations of this experiment. One where the perturbation is high ($\epsilon = 0.1$) for measuring the runtime for *falsification* when the network is not robust, and one where the perturbation is low ($\epsilon = 0.001$) for *verification* when the network is indeed robust for all inputs. We visualize both experiments in the same plot in Figure 2. Unfortunately, due to the erratic behavior of the verification tool when integrating multiple neural networks, the input space in our experiments deviates slightly from the intended specification: instead of being constrained by the UDL, the values of x are restricted to a small box centered within the high-density region of the distribution, containing less than 1% of the UDL. To keep the

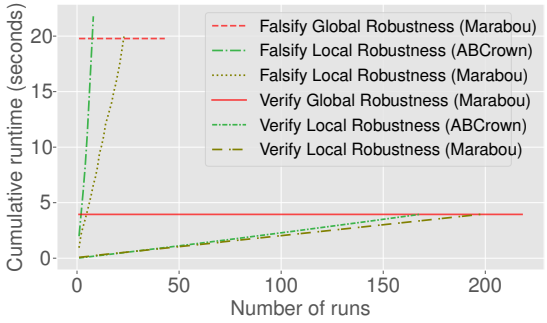

Figure 2: Robustness verification and falsification

experiment consistent, we sample from the aforementioned box of the flow model as input for the instance-wise local robustness verification (instead of picking points from the dataset $\mathcal{D}$ as presented earlier). By doing so, we ensure that the robustness certificate of the flow-based specification still implies instance-wise robustness. While the results shown in Figure 2 may represent an optimistic estimate of the runtimes for the intended specification, they still provide insight into how many instance-wise verifications are needed before the additional overhead introduced by the flow model can become worthwhile. Specifically, Figure 2 plots the cumulative runtime of instance-wise verification (falsification) runs to the runtime of our global robustness verification (falsification). There, we can observe that after around 170 samples for Marabou and 160 samples for ABCrown, the runtime for instance-wise verification exceeds the runtime of our global robustness verification of 3.7 seconds. For falsification, we can observe that after around 23 samples with Marabou and 7 samples with ABCrown, the cumulative runtime of local robustness falsification exceeds the runtime of our global robustness falsification. We conjecture that verification than falsification in our experiments due to the bound propagation method used in Marabou already simplifying or trivializing the MILP formulation. Since the MNIST test set contains around 1000 instances for each digit, a (successful) verification of our global robustness specification could already pay off after 20% of the test cases.

Ablation Study

In this section, we show the effectiveness of our architecture as a density estimator and generative model. Specifically, we compare VeriFlow against MACow (Ma et al. 2019b), a well-established flow-based model built exclusively from masked additive convolutional coupling layers. Hence, MACow is uniformly scaling and shares the properties described in Proposition 2 and Corollary 1 with VeriFlow. To the authors knowledge, MACow is currently the strongest uniformly scaling flow baseline. The key difference between these two architectures is that VeriFlow extends MACow by applying an adjoint affine group action to the coupling layers. Furthermore, VeriFlow uses a custom radial base distribution with a mixture of gammas norm-distribution, while MACow uses the classical standard normal. Otherwise, the

Dataset	MACow	Veriflow (Ours)
MNIST digit 0	−1719.908	**-2320.757**
MNIST digit 1	−1577.637	**-3123.905**
MNIST digit 2	−1026.944	**-2176.636**
MNIST digit 3	−1237.165	**-2531.264**
MNIST digit 4	−1033.265	**-2254.180**
MNIST digit 5	−1651.304	**-3013.184**
MNIST digit 6	−1331.267	**-2820.397**
MNIST digit 7	−891.180	**-2238.973**
MNIST digit 8	−1694.938	**-2971.620**
MNIST digit 9	−2006.115	**-3062.877**
Full MNIST	−1250.627	**-2389.460**
Fashion MNIST	−847.522	**-1386.659**
CIFAR-10	−4215.694	**-4500.373**

Table 1: Negative log-likelihood (NLL) comparison.

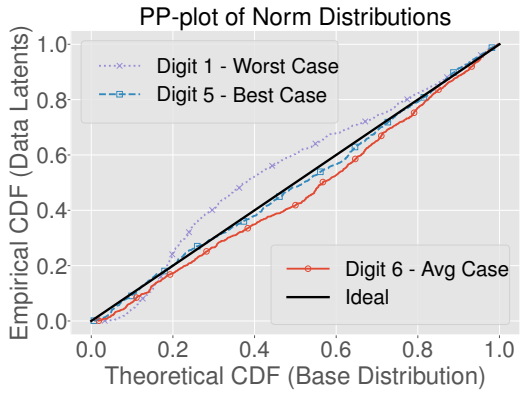

Figure 3: VeriFlow's Latent norm PP-plot for selected MNIST digits.

parametrization is kept the same, except for a learning rate of 10^{-3} and a patience of 3, i.e. early stopping after 3 consecutive epochs without validation loss improvement, for VeriFlow and a learning rate of 10^{-5} a patience of 10 for MACow — this was necessitated because the training of MACow was unstable otherwise. We use 5 coupling blocks (except for CIFAR-10, where we use 10 blocks) and 3 layers per conditioner with a uniform kernel size of 3 for both. We use the SophiaG optimizer (Liu et al. 2023).

We train our models on several widely used, moderate-dimensional image datasets, including individual MNIST digits, full MNIST, FashionMNIST, and CIFAR-10, using an IdeaPad laptop with a Ryzen 7 7735HS processor and 16 GB of RAM. As usual for density estimation on (8 bit) images, we used uniform dequantization, i.e. adding uniform noise sampled from $[0, 1/255]$, to obtain a continuous distribution, and report the NLL of dequantized images (Tsubota and Aizawa 2023). We summarize our results in Table 1. Overall, we observed more stable training and considerably higher likelihoods with our architecture.

Related Work & Conclusion

Flow models are on the forefront of modern density estimation and have received significant attention over the last decade (Papamakarios et al. 2019). A constant Jacobian determinant is usually observed at the time of the introduction of the respective layer in the context of the likelihood computation (Dinh, Krueger, and Bengio 2015; Ma et al. 2019b; Kingma et al. 2017; Papamakarios, Pavlakou, and Murray 2017), although typically without further investigation of the induced properties. The role of the Jacobian determinant in general has been investigated in the context of the exploding determinant phenomenon (Kim et al. 2020; Liao and He 2021; Lyu et al. 2022). Deeper investigations of uniformly scaling flows have been conducted in (Maziarka et al. 2022), where connections to DeepSVDDs are established by training flows with a volume minimization objective and, recently, in (Draxler et al. 2024), which shows that learnable norm-distributions are necessary for the universality of uniformly scaling flows. The latter result confirms observations that we made during our experiments and that led us to implement learnable norm-distributions via mixture models.

Using **neural networks as part of the specification** to capture semantic properties — such as meaningful perturbations for improving robustness verification — has been explored in prior work (Mirman, Gehr, and Vechev 2020; Mohapatra et al. 2020; Wu et al. 2023). The *neuro-symbolic verification* framework generalizes this idea by proposing a specification language that supports the verification of a neural network with properties specified by other neural networks. (Xie, Kersting, and Neider 2022).

Most **verification infrastructure** supports the neural architecture of our flow model and are able to precisely represent the shape of its UDL. Specifically, Polytopes, Zonotopes, or even simple boxes are sufficient for representing the UDL of our flow model in the latent space precisely. This enables the use of a variety of domains that are based on the aforementioned shapes, such as Deepzono (Singh et al. 2018). Besides Marabou and ABCrown, all other verifiers that support the VNN-Lib standard (Demarchi et al. 2023) are conceivable for verification with VeriFlow.

In the past, **global robustness** specifications have been proposed by Leino, Wang, and Fredrikson (2021); Katz et al. (2017); Chen et al. (2021); Gopinath et al. (2018); Kabaha and Drachsler-Cohen (2024). However, while some works restrict the input space in their global robustness specification, none of these restrictions are based on the notion of probabilistic interpretability.

Conclusion We have presented VeriFlow, a flow-based density model that enables effective verification of neural networks within a learned proxy distribution and fits in existing verification infrastructure.

Acknowledgements

This work has been financially supported by Deutsche Forschungsgemeinschaft, DFG Project number 459419731, and the Research Center Trustworthy Data Science and Security (https://rc-trust.ai), one of the Research Alliance centers within the UA Ruhr (https://uaruhr.de).

References

Alberti, M.; Bobot, F.; Girard-Satabin, J.; Grastien, A.; Varasse, A.; and Chihani, Z. 2025. The CAISAR Platform: Extending the Reach of Machine Learning Specification and Verification. *CoRR*, abs/2506.12084.

Brix, C.; Bak, S.; Johnson, T. T.; and Wu, H. 2024. The Fifth International Verification of Neural Networks Competition (VNN-COMP 2024): Summary and Results. *CoRR*, abs/2412.19985.

Chan, R. K.-W.; Penquitt, S.; and Gottschalk, H. 2023. LU-Net: Invertible Neural Networks Based on Matrix Factorization. In *2023 International Joint Conference on Neural Networks (IJCNN)*, 1–10. IEEE. ISBN 978-1-66548-867-9.

Chen, Y.; Wang, S.; Qin, Y.; Liao, X.; Jana, S.; and Wagner, D. A. 2021. Learning Security Classifiers with Verified Global Robustness Properties. In Kim, Y.; Kim, J.; Vigna, G.; and Shi, E., eds., *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, 477–494. ACM.

Demarchi, S.; Guidotti, D.; Pulina, L.; and Tacchella, A. 2023. Supporting Standardization of Neural Networks Verification with VNNLIB and CoCoNet. In Narodytska, N.; Amir, G.; Katz, G.; and Isac, O., eds., *Proceedings of the 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems, FoMLAS@CAV 2023, Paris, France, July 17-18, 2023*, volume 16 of *Kalpa Publications in Computing*, 47–58. EasyChair.

Dinh, L.; Krueger, D.; and Bengio, Y. 2015. NICE: Nonlinear Independent Components Estimation. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

Dodge, Y. 2008. *The Concise Encyclopedia of Statistics: Binomial Test*, 36–39. New York, NY: Springer. ISBN 978-0-387-32833-1.

Draxler, F.; Wahl, S.; Schnörr, C.; and Köthe, U. 2024. On the universality of volume-preserving and coupling-based normalizing flows. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Dunn, O. J. 1961. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293): 52–64.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014a. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014b. Generative Adversarial Nets. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Gopinath, D.; Katz, G.; Păsăreanu, C. S.; and Barrett, C. 2018. DeepSafe: A Data-Driven Approach for Assessing Robustness of Neural Networks. In Lahiri, S. K.; and Wang, C., eds., *Automated Technology for Verification and Analysis*, 3–19. Cham: Springer International Publishing. ISBN 978-3-030-01090-4.

Hoare, C. A. R. 1969. An Axiomatic Basis for Computer Programming. *Commun. ACM*, 12(10): 576–580.

Hoogeboom, E.; van den Berg, R.; and Welling, M. 2019. Emerging Convolutions for Generative Normalizing Flows. *International Conference on Machine Learning (ICML)*, 2771–2780.

Kabaha, A.; and Drachsler-Cohen, D. 2024. Verification of Neural Networks' Global Robustness. *Proc. ACM Program. Lang.*, 8(OOPSLA1): 1010–1039.

Katz, G.; Barrett, C. W.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In Majumdar, R.; and Kuncak, V., eds., *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, 97–117. Springer.

Kim, H.; Lee, H.; Kang, W. H.; Lee, J. Y.; and Kim, N. S. 2020. SoftFlow: Probabilistic Framework for Normalizing Flow on Manifolds. In *NeuIPS*.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative Flow with Invertible 1x1 Convolutions. *Advances in Neural Information Processing Systems (NeurIPS)*, 31: 10215–10224. ArXiv preprint arXiv:1807.03039.

Kingma, D. P.; Salimans, T.; Jozefowicz, R.; Chen, X.; Sutskever, I.; and Welling, M. 2017. Improved Variational Inference with Inverse Autoregressive Flow. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, 4743–4751. Curran Associates Inc. ISBN 978-1-5108-3881-9.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In Bengio, Y.; and LeCun, Y., eds., *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Knothe, H. 1957. Contributions to the Theory of Convex Bodies. *Michigan Mathematical Journal*, 4(1): 39–52.

Leino, K.; Wang, Z.; and Fredrikson, M. 2021. Globally-Robust Neural Networks. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, 6212–6222. PMLR.

Liao, H.; and He, J. 2021. Jacobian Determinant of Normalizing Flows. *CoRR*, abs/2102.06539.

Linhart, J.; Gramfort, A.; and Rodrigues, P. 2023. L-C2ST: Local Diagnostics for Posterior Approximations in Simulation-Based Inference. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Liu, H.; Li, Z.; Hall, D.; Liang, P.; and Ma, T. 2023. Sophia: A Scalable Stochastic Second-order Optimizer for Language Model Pre-training. *arXiv preprint*, arXiv:2305.14342. Version 4 revised March 2024.

Lyu, J.; Chen, Z.; Feng, C.; Cun, W.; Zhu, S.; Geng, Y.; Xu, Z.; and Yongwei, C. 2022. Para-CFlows: C^k-Universal Diffeomorphism Approximators as Superior Neural Surrogates. In *Advances in Neural Information Processing Systems*, volume 35, 28829–28841.

Ma, X.; Kong, X.; Wang, S.; and Hovy, E. 2019a. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. *International Conference on Machine Learning (ICML)*, 4314–4323.

Ma, X.; Kong, X.; Zhang, S.; and Hovy, E. 2019b. MaCow: Masked Convolutional Generative Flow. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 5893–5902. Curran Associates Inc.

Maziarka, L.; Smieja, M.; Sendera, M.; Struski, L.; Tabor, J.; and Spurek, P. 2022. OneFlow: One-Class Flow for Anomaly Detection Based on a Minimal Volume Region. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11): 8508–8519.

Mirman, M.; Gehr, T.; and Vechev, M. T. 2020. Robustness Certification of Generative Models. *CoRR*, abs/2004.14756.

Mohapatra, J.; Weng, T.; Chen, P.; Liu, S.; and Daniel, L. 2020. Towards Verifying Robustness of Neural Networks Against A Family of Semantic Perturbations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 241–249. Computer Vision Foundation / IEEE.

Moser, B. A.; Lewandowski, M.; Kargaran, S.; Zellinger, W.; Biggio, B.; and Koutschan, C. 2022. Tessellation-Filtering ReLU Neural Networks. In *Tessellation-Filtering ReLU Neural Networks*, volume 4, 3335–3341.

Papamakarios, G.; Nalisnick, E.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2019. Normalizing Flows for Probabilistic Modeling and Inference. *ArXiv191202762 Cs Stat*.

Papamakarios, G.; Nalisnick, E.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2021a. Normalizing Flows for Probabilistic Modeling and Inference. *The Journal of Machine Learning Research*, 22(1): 57:2617–57:2680.

Papamakarios, G.; Nalisnick, E. T.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2021b. Normalizing Flows for Probabilistic Modeling and Inference. *J. Mach. Learn. Res.*, 22: 57:1–57:64.

Papamakarios, G.; Pavlakou, T.; and Murray, I. 2017. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Rezende, D.; and Mohamed, S. 2015. Variational inference with normalizing flows. In *International conference on machine learning*, 1530–1538. PMLR.

Rizzo, M. L.; and Székely, G. J. 2016. Energy Distance. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8(1): 27–38.

Rosenblatt, M. 1952. Remarks on a Multivariate Transformation. *The Annals of Mathematical Statistics*, 23(3): 470–472.

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. T. 2018. Fast and Effective Robustness Certification. In Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 10825–10836.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In Bengio, Y.; and LeCun, Y., eds., *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Tsubota, K.; and Aizawa, K. 2023. Comprehensive Comparisons of Uniform Quantization in Deep Image Compression. *IEEE Access*, 11: 4455–4465.

Wu, H.; Isac, O.; Zeljić, A.; Tagomori, T.; Daggitt, M.; Kokke, W.; Refaeli, I.; Amir, G.; Julian, K.; Bassan, S.; et al. 2024a. Marabou 2.0: A Versatile Formal Analyzer of Neural Networks. *arXiv preprint arXiv:2401.14461*.

Wu, H.; Isac, O.; Zeljic, A.; Tagomori, T.; Daggitt, M. L.; Kokke, W.; Refaeli, I.; Amir, G.; Julian, K.; Bassan, S.; Huang, P.; Lahav, O.; Wu, M.; Zhang, M.; Komendantskaya, E.; Katz, G.; and Barrett, C. W. 2024b. Marabou 2.0: A Versatile Formal Analyzer of Neural Networks. In Gurfinkel, A.; and Ganesh, V., eds., *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part II*, volume 14682 of *Lecture Notes in Computer Science*, 249–264. Springer.

Wu, H.; Tagomori, T.; Robey, A.; Yang, F.; Matni, N.; Pappas, G. J.; Hassani, H.; Pasareanu, C. S.; and Barrett, C. W. 2023. Toward Certified Robustness Against Real-World Distribution Shifts. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning, SaTML 2023, Raleigh, NC, USA, February 8-10, 2023*, 537–553. IEEE.

Xie, X.; Kersting, K.; and Neider, D. 2022. Neuro-Symbolic Verification of Deep Neural Networks. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 3622–3628. ijcai.org.

Xu, K.; Zhang, H.; Wang, S.; Wang, Y.; Jana, S.; Lin, X.; and Hsieh, C. 2021. Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Zhao, Z.-Q.; Zheng, P.; Xu, S.-t.; and Wu, X. 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11): 3212–3232.