Exponent Relaxation of Polynomial Zonotopes and Its Applications in Formal Neural Network Verification

Tobias Ladner, Matthias Althoff

Technical University of Munich, Germany
{tobias.ladner,althoff}@tum.de

Abstract

Formal verification of neural networks is a challenging problem due to the complexity and nonlinearity of neural networks. It has been shown that polynomial zonotopes can tightly enclose the output set of a neural network. Unfortunately, the tight enclosure comes with additional complexity in the set representation, thus, rendering subsequent operations expensive to compute, such as computing interval bounds and intersection checking. To address this issue, we present a novel approach to restructure a polynomial zonotope to tightly enclose the original polynomial zonotope while drastically reducing its complexity. The restructuring is achieved by relaxing the exponents of the dependent factors of polynomial zonotopes and finding an appropriate approximation error. We demonstrate the applicability of our approach on output sets of neural networks, where we obtain tighter results in various subsequent operations, such as order reduction, zonotope enclosure, and range bounding.

1 Introduction

While neural networks demonstrate impressive results in various fields (Kiran et al. 2021), their applicability to safety-critical environments still needs to be improved due to their vulnerabilities to adversarial attacks (Goodfellow, Shlens, and Szegedy 2015). Adversarial attacks show that slight perturbations to the input of a neural network can result in unexpected outputs. Therefore, in recent years there has been a growing interest in the formal verification of neural networks (Bak, Liu, and Johnson 2021; Lopez et al. 2022). While it is shown that this problem is NP-hard for ReLU networks (Katz et al. 2017), many sound approaches verify neural networks by formulating the problem as an optimization problem (Xu et al. 2020; Ferrari et al. 2022; Katz et al. 2019; Botoeva et al. 2020) or use reachability analysis (Henriksen and Lomuscio 2020; Kochdumper et al. 2023; Bogomolov et al. 2019; Schilling, Forets, and Guadalupe 2022). Abstraction refinement strategies like branch-and-bound can improve the verification results (Wang et al. 2021).

In this work, we focus on reachability analysis using polynomial zonotopes (Kochdumper and Althoff 2020) and consider an abstraction refinement strategy, which outer-approximates nonlinear layers using higher-order polynomials (Ladner and Althoff 2023a). The main advantage of this refinement strategy over branch-and-bound approaches is that we maintain a single set. A single set is especially attractive for closed-loop verification (Lopez et al. 2022), where splitting is particularly expensive due to repeated neural network evaluations. While we can enclose the nonlinear layers tightly using higher-order polynomials, subsequent operations on the output set might be more expensive due to the complexity of the resulting polynomial zonotope, including intersection checking with an unsafe set. An alternative is to outer-approximate activation functions using linear functions to keep the set representation simple at the cost of losing accuracy. Better accuracy can then only be obtained by splitting sets at the input or the neuron level, causing the aforementioned problems.

While challenges arising from polynomial zonotopes have been addressed (Huang et al. 2023), common strategies to simplify them remain splitting and the enclosure by a zonotope (see Sec. 2). However, the zonotope enclosure can be very outer-approximative as dependencies between the monomials of a polynomial zonotope are lost. Unfortunately, these dependencies naturally occur when using higher-order polynomials to outer-approximate nonlinear functions. We address this dependency issue in this work.

To summarize our contributions, we present a novel approach to restructure polynomial zonotopes such that we have fewer dependencies between the monomials without inducing large outer-approximations by preserving the underlying structure. The restructuring is done by a) finding monomials with similar exponents, b) relaxing the exponents from one monomial to match the other, and c) finding an appropriate approximation error. Further, we present an approach to efficiently apply our novel exponent relaxation on the entire polynomial zonotope using graph theory. To the best of our knowledge, our approach is the first to consider the dependencies of monomials during restructuring.

2 Preliminaries

Notation

We denote scalars and vectors by lower-case letters, matrices by upper-case letters, and sets by calligraphic letters.

The i-th element of a vector $v \in \mathbb{R}^n$ is written as $v_{(i)}$. The element in the i-th row and j-th column of a matrix $A \in \mathbb{R}^{n \times m}$ is written as $A_{(i,j)}$, the entire i-th row and j-th column are written as $A_{(i,\cdot)}$ and $A_{(\cdot,j)}$, respectively. The concatenation of A with a matrix $B \in \mathbb{R}^{n \times o}$ is denoted by $[A\ B] \in \mathbb{R}^{n \times (m+o)}$. We denote with I_n the identity matrix of dimension $n \in \mathbb{N}$. The symbols $\mathbf{0}$ and $\mathbf{1}$ refer to matrices with all zeros and ones of proper dimensions, respectively. Given $a \leq b \in \mathbb{N}_0$, then $[b]_a = \{a, a+1, \dots, b\}$ and $[b] = [b]_1$. The cardinality of a discrete set $\mathcal{D}$ is denoted by $|\mathcal{D}|$. Let $\mathcal{D} \subseteq [n]$, then $A_{(\mathcal{D},\cdot)}$ denotes all rows $i \in \mathcal{D}$; this is used analogously for columns. Let $\mathcal{S} \subset \mathbb{R}^n$ be a set and $f : \mathbb{R}^n \to \mathbb{R}^m$ be a function, then $f(\mathcal{S}) = \{f(x) \mid x \in \mathcal{S}\}$. An interval with bounds $a, b \in \mathbb{R}^n$ is denoted by $[a, b]$.

Set-based Computing

Next, we briefly introduce all required set operations to show our novel approach, starting with polynomial zonotopes:

Definition 1 (Polynomial Zonotope (Kochdumper and Althoff 2020)[1]). *Given an offset $c \in \mathbb{R}^n$, a generator matrix of dependent generators $G \in \mathbb{R}^{n \times h}$, an exponent matrix $E \in \mathbb{N}_0^{p \times h}$ with an identifier vector $\mathtt{id} \in \mathbb{N}^p$, and a generator matrix of independent generators $G_I \in \mathbb{R}^{n \times q}$, a polynomial zonotope $\mathcal{PZ} = \langle c, G, G_I, E \rangle_{PZ}$ is defined as*

$$\mathcal{PZ} \coloneqq \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1] \right\}.$$

If not otherwise stated, the variables n, h, p, q are as in Def. 1. In addition, we use the shorthand $\mathcal{PZ} = \{c + \Sigma\Pi G + \Sigma G_I \,|\, \alpha, \beta\}$ to make proofs more concise. Given two polynomial zonotopes $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1 \rangle_{PZ}, \mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2 \rangle_{PZ} \subset \mathbb{R}^n$, and $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$, then the affine map and the Minkowski sum can be computed as

$$A\mathcal{PZ}_1 + b = \langle Ac_1 + b, AG_1, AG_{I,1}, E_1 \rangle_{PZ}, \quad (1)$$

$$\mathcal{PZ}_1 \oplus \mathcal{PZ}_2 = \{x_1 + x_2 \mid x_1 \in \mathcal{PZ}_1, x_2 \in \mathcal{PZ}_2\} = \left\langle c_1 + c_2, [G_1\ G_2], [G_{I,1}\ G_{I,2}], \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & E_2 \end{bmatrix} \right\rangle_{PZ}. \quad (2)$$

If $\mathcal{PZ}_1, \mathcal{PZ}_2$ share the same identifier vector $\mathtt{id}$, we can compute an exact addition (Kochdumper and Althoff 2020, Prop. 10) using

$$\mathcal{PZ}_1 \boxplus \mathcal{PZ}_2 = \langle c_1 + c_2, [G_1 \quad G_2], [G_{I,1} \quad G_{I,2}], [E_1 \quad E_2] \rangle_{PZ}. \quad (3)$$

An interval $\mathcal{I} = [l, u] \subset \mathbb{R}^n$ can be equivalently represented by a polynomial zonotope using

$$\mathcal{I} = \langle 0.5(u + l), 0.5\,\mathtt{diag}(u - l), [\,], I_n \rangle_{PZ}. \quad (4)$$

[1] As in (Kochdumper 2022), we adapt the definition from (Kochdumper and Althoff 2020) and do not integrate the offset c into the generator matrix G and omit the identifier vector almost always for simplicity.

The main advantage of using polynomial zonotopes is that we can tightly approximate nonlinear functions (Althoff 2013; Ladner and Althoff 2023a) using the quadratic map operation as it can be computed exactly in favorable computation time.

Proposition 1 (Quadratic Map (Kochdumper 2022, Prop. 3.1.30)). *Given a $\mathcal{PZ} = \langle c, G, [\,], E \rangle_{PZ} \subset \mathbb{R}^n$, the result of the element-wise quadratic map is*

$$\mathtt{quadMap}(\mathcal{PZ}) = \left\langle \overline{c}, \begin{bmatrix} 2\overline{G}_c & \overline{G}_1 & \dots & \overline{G}_h \end{bmatrix}, [\,], \begin{bmatrix} E & \overline{E}_1 & \dots & \overline{E}_h \end{bmatrix} \right\rangle_{PZ},$$

where

$$\overline{c} = \begin{bmatrix} c_{(1)}^2 \\ \vdots \\ c_{(n)}^2 \end{bmatrix}, \ \overline{G}_c = \begin{bmatrix} c_{(1)} G_{(1,\cdot)} \\ \vdots \\ c_{(n)} G_{(n,\cdot)} \end{bmatrix}, \ \overline{G}_j = \begin{bmatrix} G_{(1,j)} G_{(1,\cdot)} \\ \vdots \\ G_{(n,j)} G_{(n,\cdot)} \end{bmatrix},$$

and the exponent matrix $\overline{E}_j = E + E_{(\cdot,j)} \cdot \mathbf{1},\ j \in [h]$.

For example, an element-wise evaluation of a polynomial zonotope $\mathcal{PZ} \subset \mathbb{R}^n$ on the quadratic polynomials $p_i(x) = a_{0(i)} + a_{1(i)} x_{(i)} + a_{2(i)} x_{(i)}^2,\ i \in [n]$, is given by

$$p(\mathcal{PZ}) = a_0 + A_1 \mathcal{PZ} \boxplus A_2 \mathtt{quadMap}(\mathcal{PZ}), \quad (5)$$

where a_0, $A_1 = \mathtt{diag}(a_1)$, and $A_2 = \mathtt{diag}(a_2)$ contain the polynomial coefficients for each dimension. Higher-order polynomials use multiple quadratic map operations. Unfortunately, some other operations are harder to compute using polynomial zonotopes. For example, previous order reduction in (Kochdumper 2022, Prop. 3.1.39) needs a zonotope enclosure that induces outer-approximations. We briefly introduce the zonotope enclosure to show that the outer-approximation is drastically reduced using our approach.

Definition 2 (Zonotope (Girard 2005, Def. 1)). *Given a center vector $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n \times q}$, a zonotope is defined as*

$$\mathcal{Z} = \langle c, G \rangle_Z \coloneqq \left\{ c + \sum_{j=1}^{q} \beta_j G_{(\cdot,j)} \;\middle|\; \beta_j \in [-1, 1] \right\}.$$

Proposition 2 (Zonotope Enclosure (Kochdumper 2022, Prop. 3.1.14)). *Given a $\mathcal{PZ} = \langle c, G, G_I, E \rangle_{PZ}$, then $\mathcal{PZ} \subseteq \mathcal{Z} = \mathtt{zonotope}(\mathcal{PZ})$ with*

$$\mathcal{Z} = \left\langle c + \sum_{i \in \mathcal{H}} 0.5 G_{(\cdot,i)}, \begin{bmatrix} 0.5 G_{(\cdot,\mathcal{H})} & G_{(\cdot,\mathcal{K})} & G_I \end{bmatrix} \right\rangle_Z,$$

where $\mathcal{H}$ contains the indices of the generators with all even exponents and $\mathcal{K} = [h] \backslash \mathcal{H}$.

Neural Network Verification

In this work, we focus on verifying feed-forward neural networks (Bishop and Nasrabadi 2006, Sec. 5.1), consisting of alternating linear and nonlinear layers.

Definition 3. *(Neural Networks (Bishop and Nasrabadi 2006, Sec. 5.1)) Let $x \in \mathbb{R}^{n_0}$ be the input of a neural network Φ, its output $y = \Phi(x) \in \mathbb{R}^{n_\kappa}$ is obtained as follows:*

$$\begin{aligned} h_0 &= x, \\ h_k &= L_k(h_{k-1}), \quad k \in [\kappa], \\ y &= h_\kappa, \end{aligned}$$

where

$$L_k(h_{k-1}) = \begin{cases} W_k h_{k-1} + b_k & \text{if layer } k \text{ is linear,} \\ \sigma_k(h_{k-1}) & \text{otherwise,} \end{cases}$$

with $W_k \in \mathbb{R}^{n_k \times n_{k-1}}$, $b_k \in \mathbb{R}^{n_k}$, and $\sigma_k(\cdot)$ is the respective continuous activation function (e.g., sigmoid or ReLU).

Neural network verification reasons about input sets $\mathcal{X} \subset \mathbb{R}^{n_0}$ instead of points. State-of-the-art verifiers compute an outer-approximation of the exact output set $\mathcal{Y}^* = \Phi(\mathcal{X}) \subset \mathbb{R}^{n_\kappa}$, including the ones using polynomial zonotopes (Kochdumper et al. 2023; Ladner and Althoff 2023a). On a high level, the verification procedure is shown in Alg. 1. Linear layers can be computed exactly using polynomial zonotopes (line 3). Nonlinear layers are enclosed by approximating them with a polynomial and finding an approximation error (line 4-5).

Algorithm 1: Simplified Neural Network Verification

Require: Input $\mathcal{X}$, neural network Φ
1: $\mathcal{H}_0 \leftarrow \mathcal{X}$
2: **for** $k = 1, 3, \ldots, \kappa - 1$ **do** ▷ κ layers of Φ
3: $\quad \mathcal{H}_k \leftarrow W_k\, \mathcal{H}_{k-1} + b_k$ ▷ Affine map (1)
4: $\quad \widetilde{\mathcal{H}}_{k+1} \leftarrow p_{k+1}(\mathcal{H}_k)$ ▷ Polynomial map (Prop. 1)
5: $\quad \mathcal{H}_{k+1} \leftarrow \widetilde{\mathcal{H}}_{k+1} \oplus \mathcal{I}_k$ ▷ Approx. error (4), (2)
6: **end for**
7: $\mathcal{Y} \leftarrow \mathcal{H}_\kappa$
8: **return** $\mathcal{Y} \supseteq \mathcal{Y}^*$

Problem Statement

Given a polynomial zonotope $\mathcal{PZ}$, e.g., obtained from Alg. 1, we want to restructure $\mathcal{PZ}$ to obtain a new, simpler polynomial zonotope $\overline{\mathcal{PZ}} \supseteq \mathcal{PZ}$ with fewer dependencies between the monomials and minor outer-approximations. The restructuring should benefit subsequent operations, e.g., interval bounds, intersection checking, and order reduction.

3 Exponent Relaxation

Polynomial zonotopes can be hard to deal with as they are defined over polynomials with multiple variables (Def. 1). Thus, many operations are computed via a zonotope enclosure (Prop. 2) in practice, e.g., computing interval bounds, order reduction, intersection checking, and plotting (Kochdumper 2022). However, we observe that the zonotope enclosure can be very outer-approximative, especially if there are many dependencies between the monomials. As the next example demonstrates, the outer-approximation can be quite significant even in simple cases:

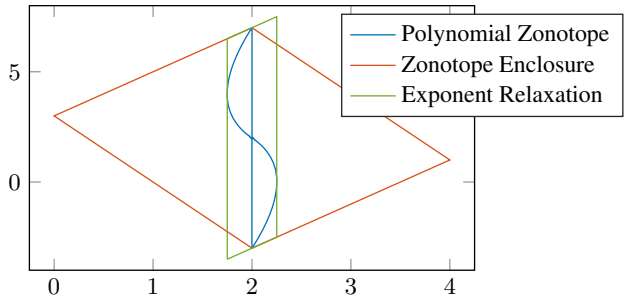

Figure 1: Outer-approximation of the zonotope enclosure.

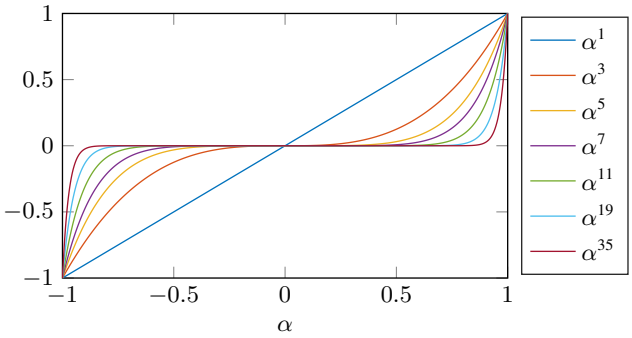

Figure 2: Dependent factors $\alpha \in [-1, 1]$ in Def. 1.

Example 1. *Given a polynomial zonotope*

$$\mathcal{PZ} = \left\langle \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ 2 & 3 \end{bmatrix}, \begin{bmatrix} \ \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 4 & 2 \end{bmatrix} \right\rangle_{PZ},$$

we obtain

$$\mathcal{Z} = \texttt{zonotope}\,(\mathcal{PZ}) = \left\langle \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ 2 & 3 \end{bmatrix} \right\rangle_{Z}$$

using Prop. 2, which is also depicted in Fig. 1. The zonotope enclosure makes the generators independent, thus, we lose the information that they cancel each other in the first dimension – resulting in a large outer-approximation.

Given a polynomial zonotope $\mathcal{PZ}$, we present a novel approach to construct a polynomial zonotope $\overline{\mathcal{PZ}} \supseteq \mathcal{PZ}$ with fewer dependencies between the monomials without inducing large outer-approximations by relaxing the exponents of the dependent factors. Fig. 2 shows that the maximum difference between two dependent factors with odd exponents is minor within $[-1, 1]$, especially for higher exponents. Analogously, this also holds for even exponents. We define the following operator to compute the maximum difference between two exponents for convenience.

Proposition 3 (Exponent Difference)**.** *Given two exponents $e_i, e_j \in \mathbb{N}_0$, then the maximum difference between the respective polynomials within $[-1, 1]$ is given by*

$$\texttt{eDiff}\,(e_i, e_j) = \max_{x \in [-1,1]} \left| x^{e_i} - x^{e_j} \right|.$$

Our novel approach merges generators whose monomials have similar exponents, which we define as follows:

Definition 4 (Similar Exponents)**.** *Given a polynomial zonotope $\mathcal{PZ} = \langle c, G, G_I, E\rangle_{PZ}$ with h dependent generators, then we say that the monomials of two generators $i, j \in [h]$ have similar exponents if and only if they differ at most in one index:*

$$\left|\left\{k \,\middle|\, E_{(k,i)} \neq E_{(k,j)},\ k \in [p]\right\}\right| = 1.$$

With that, we can present the first main theorem of our novel restructuring approach:

Theorem 1 (Exponent Relaxation)**.** *Given a polynomial zonotope $\mathcal{PZ} = \langle c, G, G_I, E\rangle_{PZ}$ with h dependent generators and the indices of two generators $i, j \in [h]$ with similar exponents, then we can relax the exponents of the monomials of these generators to obtain a new polynomial zonotope*

$$\overline{\mathcal{PZ}} = \left\langle \overline{c}, \overline{G}, \overline{G}_I, \overline{E}\right\rangle_{PZ} = \texttt{relax}\left(\mathcal{PZ}, i, j\right) \supseteq \mathcal{PZ}$$

with

$$\overline{c} = c + \xi_c G_{(\cdot,i)}, \qquad \overline{G} = \left[G_{(\cdot,[h]\setminus\{i,j\})}\ G_{(\cdot,j)}{+}G_{(\cdot,i)}\right],$$
$$\overline{G}_I = \left[G_I\ \xi_G G_{(\cdot,i)}\right], \quad \overline{E} = \left[E_{(\cdot,[h]\setminus\{i,j\})}\ E_{(\cdot,j)}\right],$$

where

$$\xi_c = \xi_E - \xi_G, \qquad \xi_G = \begin{cases} \xi_E/2 & \text{if } E_{(\cdot,i)} \text{ are all even,} \\ \xi_E & \text{otherwise,} \end{cases}$$

$$\xi_E = \texttt{eDiff}\left(E_{(k_0,i)}, E_{(k_0,j)}\right),$$

and $k_0 \in [p]$ is the index where $E_{(\cdot,i)}$ and $E_{(\cdot,j)}$ differ. The computational complexity to construct $\overline{\mathcal{PZ}}$ is in $\mathcal{O}((h+q)n + hp)$.

Proof. We only show the case where $E_{(\cdot,i)}$ does not have all even exponents, thus $\xi_c = 0$, $\xi_G = d$. In the case of all even exponents, the approximation error can be halved with the same argument as in Prop. 2. Without loss of generality, we assume $i = 1,\ j = 2$, and $k_0 = 1$. Thus,

$$\begin{aligned}
\mathcal{PZ} &= \{c + \Sigma\Pi G + \Sigma G_I \,|\, \alpha, \beta\} \\
&= \left\{ c{+}\Sigma\Pi G_{(\cdot,[h]_2)}{+}\Sigma G_I{+}\left(\prod_{k=1}^{p} \alpha_k^{E_{(k,1)}}\right) G_{(\cdot,1)} \,\middle|\, \alpha, \beta \right\} \\
&= \Big\{ c + \Sigma\Pi G_{(\cdot,[h]_2)} + \Sigma G_I \\
&\quad + \left(\alpha_1^{E_{(1,1)}} \prod_{k=2}^{p} \alpha_k^{E_{(k,1)}}\right) G_{(\cdot,1)} \,\middle|\, \alpha, \beta \Big\}.
\end{aligned}$$

Next, we relax the exponents of the monomial of the generator $i = 1$ to match the exponents from $j = 2$ and enclose the relaxation by introducing a new factor $\alpha_* \in [-1, 1]$:

$$\begin{aligned}
\mathcal{PZ} &\subseteq \Big\{ c + \Sigma\Pi G_{(\cdot,[h]_2)} + \Sigma G_I \\
&\quad + \left(\left(\alpha_1^{E_{(1,2)}} + \alpha_* \xi_E\right) \prod_{k=2}^{p} \alpha_k^{E_{(k,2)}}\right) G_{(\cdot,1)} \,\middle|\, \alpha_*, \alpha, \beta \Big\} \\
&= \left\{ c{+}\Sigma\Pi G_{(\cdot,[h]_2)}{+}\Sigma G_I{+}\left(\alpha_1^{E_{(1,2)}} \prod_{k=2}^{p} \alpha_k^{E_{(k,2)}}\right) G_{(\cdot,1)} \right. \\
&\quad \left. + \underbrace{\left(\alpha_* \prod_{k=2}^{p} \alpha_k^{E_{(k,2)}}\right)}_{\in[-1,1]} \xi_E G_{(\cdot,1)} \,\middle|\, \alpha_*, \alpha, \beta \right\} = \widehat{\mathcal{PZ}}.
\end{aligned}$$

As all dependent factors of $\xi_E G_{(\cdot,1)}$ are within $[-1, 1]$, we can replace them by a single independent factor $\beta_{q+1} \in [-1, 1]$:

$$\begin{aligned}
\widehat{\mathcal{PZ}} \subseteq \Bigg\{ & c + \Sigma\Pi G_{(\cdot,[h]_2)} + \Sigma G_I + \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,2)}}\right) G_{(\cdot,1)} \\
& + \beta_{q+1} \xi_E G_{(\cdot,1)} \,\Big|\, \alpha, \beta \Bigg\} = \widetilde{\mathcal{PZ}}.
\end{aligned}$$

As the monomials of the generators $G_{(\cdot,1)}, G_{(\cdot,2)}$ now have the same exponents, we can add the respective generators:

$$\begin{aligned}
\widetilde{\mathcal{PZ}} &= \Big\{ c + \Sigma\Pi \left[G_{(\cdot,[h]_3)} \quad G_{(\cdot,2)}{+}G_{(\cdot,1)}\right] \\
&\qquad {+}\Sigma \left[G_I \quad \xi_E G_{(\cdot,1)}\right] \,\Big|\, \alpha, \beta \Big\} \\
&= \left\{ \overline{c} + \Sigma\Pi\overline{G} + \Sigma\overline{G}_I \,\middle|\, \alpha, \beta \right\} = \overline{\mathcal{PZ}}.
\end{aligned}$$

Please note that α_* is only temporary and does not appear in $\overline{\mathcal{PZ}}$. The computational complexity follows directly from the construction of $\overline{G}, \overline{G}_I$, and $\overline{E}$. □

Example 2. *We revisit Example 1 and apply Thm. 1 to obtain*

$$\overline{\mathcal{PZ}} = \texttt{relax}\left(\mathcal{PZ}, 1, 2\right) = \left\langle \begin{bmatrix}2\\2\end{bmatrix}, \begin{bmatrix}0\\5\end{bmatrix}, \begin{bmatrix}0.25\\0.5\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix} \right\rangle_{PZ},$$

which represents the green zonotope in Fig. 1.

We want to stress that Thm. 1 is also applicable on generators i, j with more than one different exponent: This is achieved by implicitly adding a chain of zero-length generators to the polynomial zonotope, each changing one exponent such that we can iteratively apply Thm. 1:

Proposition 4 (Multi-Exponent Relaxation)**.** *Given a polynomial zonotope $\mathcal{PZ} = \langle c, G, G_I, E\rangle_{PZ}$ with h dependent generators and the indices of two generators $i, j \in [h], i \neq j$, then we can relax all exponents of the monomials of these generators to obtain a new polynomial zonotope*

$$\overline{\mathcal{PZ}} = \left\langle \overline{c}, \overline{G}, \overline{G}_I, \overline{E}\right\rangle_{PZ} = \texttt{relax}\left(\mathcal{PZ}, i, j\right) \supseteq \mathcal{PZ}$$

with identical construction as in Thm. 1, except that

$$\xi_E = \sum_{k \in \mathcal{K}} \texttt{eDiff}\left(E_{(k,i)}, E_{(k,j)}\right)$$

and $\mathcal{K} \subseteq [p]$ are the indices where $E_{(\cdot,i)}$ and $E_{(\cdot,j)}$ differ.

Proof. Again, we only consider the case where $E_{(\cdot,i)}$ does not have all even exponents and without loss of generality, we assume $i = 1,\ j = 2$, and $\mathcal{K} = [p_0] \subseteq [p]$. Then, without inducing any outer-approximation, we have that $\mathcal{PZ} = \widehat{\mathcal{PZ}} = \left\langle c, \widehat{G}, G_I, \widehat{E}\right\rangle_{PZ}$ with

$$\widehat{G} = \left[G_{(\cdot,1)} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad G_{(\cdot,2)} \quad G_{(\cdot,[h]_3)}\right],$$
$$\widehat{E} = \begin{bmatrix} E_{(\cdot,1)} & \begin{bmatrix} E_{([p_0-1],1)} \\ E_{([p]_{p_0},2)} \end{bmatrix} & \dots & \begin{bmatrix} E_{([1],1)} \\ E_{([p]_2,2)} \end{bmatrix} & E_{(\cdot,2)}\ E_{(\cdot,[h]_3)} \end{bmatrix}.$$

We apply Thm. 1 iteratively on the first $p_0 + 1$ generators of $\widehat{\mathcal{PZ}}$ as the monomials of subsequent generators have similar exponents. As the resulting independent generators are scaled generators of $G_{(\cdot,1)}$, they can be combined and we obtain ξ_E and $\overline{G}_I$. □

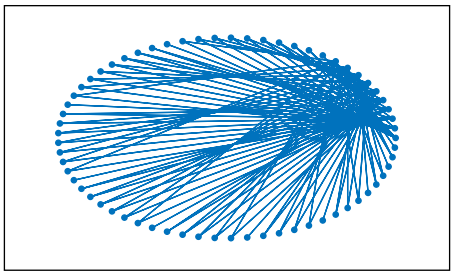

Figure 3: Resulting graph $\mathcal{G}_{\mathcal{PZ}^2,0}$ from applying a single quadratic map operation on a polynomial zonotope $\mathcal{PZ}$: Each generator has at least one other generator where the respective monomials have similar exponents.

Please note that Thm. 1 and Prop. 4 do not increase the number of generators. However, we removed one dependent generator and added one independent generator. Thus, the restructured polynomial zonotope is simpler in the sense that it is closer to a zonotope without inducing much outer-approximation due to Prop. 3 with reasonable exponents. The dimension of the polynomial zonotope does not affect the exponent relaxation directly, as the error due to the relaxation can always be enclosed by one independent generator; however, higher dimensional polynomial zonotopes usually have more dependent factors. Also, it is not apparent which generators we should merge by relaxing their exponents if one has multiple other generators with similar exponents. We discuss all three points in more detail in the next section.

4 Graph-Based Restructuring

For the exponent relaxation approach (Thm. 1) to be applicable, we need the polynomial zonotopes to have similar exponents (Def. 4). We show in this section that similar exponents are not uncommon when applying quadratic maps (Prop. 1) on polynomial zonotopes, where we view polynomial zonotopes as graphs to determine which exponents to relax. Each node in the graph represents a dependent generator of a polynomial zonotope, and each edge represents a choice to relax the exponents as follows: We want a) monomials of generators with connecting edges to have similar exponents (Def. 4), b) the dissimilar exponents to be both even, both odd, or $E_{(k,j)} = 0$, and c) do not relax exponents underneath a threshold η.

Definition 5 (Graph of Polynomial Zonotope)**.** *Let $\mathcal{PZ} = \langle c, G, G_I, E \rangle_{PZ}$ be a polynomial zonotope with h dependent generators and an exponent threshold $\eta \in \mathbb{N}_0$, then we define the directed graph $\mathcal{G}_{\mathcal{PZ},\eta} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$ from $\mathcal{PZ}$, where*

$$\begin{aligned}\mathcal{V} &= [h],\\ \mathcal{E} &= \{(i,j) \mid \textit{Def. 4 holds for } i,j \in [h]\\ &\quad \wedge \left(\texttt{mod}\left(E_{(k,i)},2\right) = \texttt{mod}\left(E_{(k,j)},2\right) \vee E_{(k,j)} = 0\right)\\ &\quad \wedge E_{(k,i)} > E_{(k,j)} \geq \eta,\ k \textit{ s.t. } E_{(k,i)} \neq E_{(k,j)}\}\,,\end{aligned}$$

with an edge weight $\texttt{eDiff}\left(E_{(k,i)}, E_{(k,j)}\right)$ for an edge $(i,j) \in \mathcal{E}$.

The computational complexity of constructing $\mathcal{G}_{\mathcal{PZ},\eta}$ is in $\mathcal{O}(h^2 p)$. With a higher value of the threshold η, the resulting polynomial zonotope remains closer to the original set, while a smaller value simplifies it more at the cost of larger outer-approximation. This tradeoff is favorable for our approach as the outer-approximation induced by the exponent relaxation is minor (Fig. 2).

Lemma 1. *The graph $\mathcal{G}_{\mathcal{PZ},\eta}$ does not contain cycles.*

Proof. The graph $\mathcal{G}_{\mathcal{PZ},\eta}$ does not contain cycles as we only add edges between the generators i, j if $E_{(k,i)} > E_{(k,j)}$ holds and all other exponents are equal (Def. 4). Thus, we cannot construct a path from generator j back to generator i as the k-th exponent cannot increase along a path in $\mathcal{G}_{\mathcal{PZ},\eta}$. □

We say that a node i is a leaf node of a directed graph if there are no incoming edges to node i but at least one outgoing edge. Monomials with similar exponents already appear after applying a single quadratic map by construction (Prop. 1), as shown in the following example.

Example 3. *Let $\mathcal{PZ} = \langle c, G, [\,], I_n \rangle_{PZ} \subset \mathbb{R}^n$ be a polynomial zonotope representing an interval* (4) *without any dependencies between the monomials of the generators. Then, all generators of $\mathcal{PZ}^2 = \texttt{quadMap}\left(\mathcal{PZ}\right)$ have at least two other generators, where the respective monomials have similar exponents as visible in the graph $\mathcal{G}_{\mathcal{PZ}^2,0}$ in Fig. 3 for $n = 10$.*

If a generator in $\mathcal{G}_{\mathcal{PZ},\eta}$ has multiple outgoing edges, we have to decide which generators are merged and in which order. We apply a greedy approach to merge the generators (Alg. 2):

Theorem 2 (Greedy Relaxation)**.** *Given a polynomial zonotope $\mathcal{PZ}$ and an exponent threshold $\eta \in \mathbb{N}_0$, then we denote the application of Alg. 2 by*

$$\texttt{relax}\left(\mathcal{PZ}, \eta\right) \supseteq \mathcal{PZ},$$

where $\texttt{greedy}\left(\mathcal{G}_{\mathcal{PZ},\eta}\right)$ only keeps the outgoing edge with the smallest edge weight for each node. The computational complexity of Alg. 2 is $\mathcal{O}((h+q)hn + h^2 p)$.

Proof. The construction of $\mathcal{G}_{\mathcal{PZ},\eta}$ is in $\mathcal{O}(h^2 p)$ (Def. 5), and the greedily selected subgraph can be constructed in $\mathcal{O}(h^2)$ by checking the weight of the outgoing edges for each node. The subgraph consists of multiple trees as there are no cycles in $\mathcal{G}_{\mathcal{PZ},\eta}$ (Lemma 1). We loop at most h-times as each node is only processed at most once. Updating the subgraph can be done in $\mathcal{O}(h)$ due to the tree structure. Then, Thm. 1 is applied in $\mathcal{O}((h+q)n + hp)$. Thus, the overall complexity is in $\mathcal{O}((h+q)hn + h^2 p)$. The greedy relaxation is sound and outer-approximative due to Thm. 1. □

We note that more advanced strategies can also be applied here, e.g., preferring generators facing in opposite directions. Fig. 4 shows how different choices for η can influence the result of $\texttt{relax}\left(\mathcal{PZ}, \eta\right)$ and a subsequent zonotope enclosure (Prop. 2). Notably, the first plot seems to show $\texttt{relax}\left(\mathcal{PZ}, \eta\right) \subset \mathcal{PZ}$, which is the opposite of what we claim in Thm. 2. However, plotting polynomial zonotopes

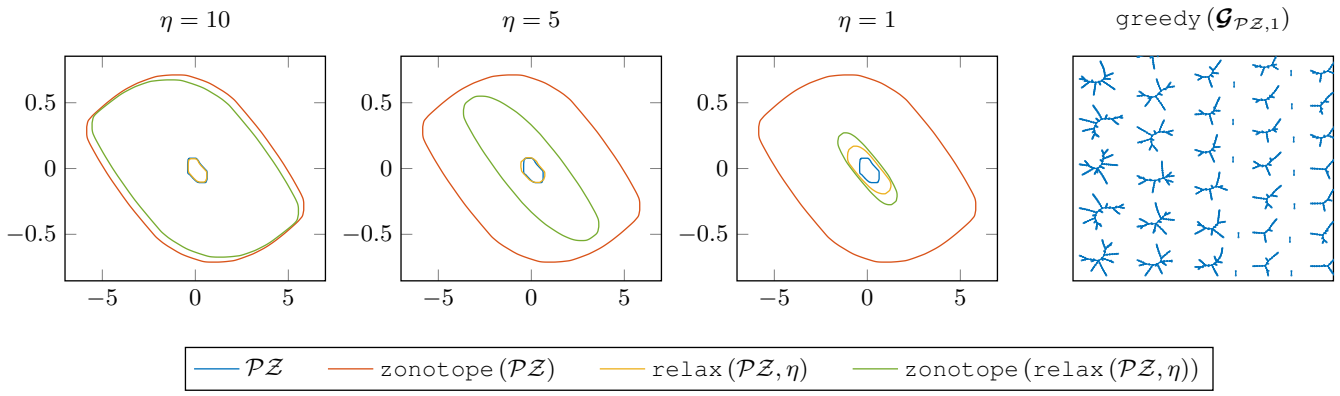

Figure 4: Resulting sets after applying Thm. 2 on a polynomial zonotope $\mathcal{PZ}$ with different values for η: The outer-approximation of the relaxed $\mathcal{PZ}$ grows barely while a subsequent zonotope enclosure shrinks drastically. The fourth plot shows the greedily selected subgraph of $\mathcal{G}_{\mathcal{PZ},1}$.

Algorithm 2: Greedy Relaxation relax $(\mathcal{PZ}, \eta)$

Require: Polynomial zonotope $\mathcal{PZ}$, threshold $\eta \in \mathbb{N}_0$
Construct $\mathcal{G}_{\mathcal{PZ},\eta}$ ▷ Def. 5
$\mathcal{G}_{\mathcal{PZ},\eta} \leftarrow$ greedy $(\mathcal{G}_{\mathcal{PZ},\eta})$
while $|\mathcal{G}_{\mathcal{PZ},\eta}.\mathcal{E}| > 0$ **do**
  $i \leftarrow$ leaf node of $\mathcal{G}_{\mathcal{PZ},\eta}$
  $j \leftarrow j \in [h]$ *s.t.* $(i,j) \in \mathcal{G}_{\mathcal{PZ},\eta}.\mathcal{E}$
  $\mathcal{PZ} \leftarrow$ relax $(\mathcal{PZ}, i, j)$ ▷ Thm. 1
  Update $\mathcal{G}_{\mathcal{PZ},\eta}$ ▷ Remove (i,j), update indices
end while
return $\mathcal{PZ}$

is outer-approximative and the plot for relax $(\mathcal{PZ}, \eta)$ is tighter as an order reduction and zonotope enclosures are involved. Any point sampled from $\mathcal{PZ}$ is, of course, also contained in relax $(\mathcal{PZ}, \eta)$. Finally, for $\eta = 1$ we obtain a tight zonotope enclosure using our novel approach. Note that the plotted polynomial zonotopes (blue and yellow) are obtained via recursive splitting (Kochdumper 2022, Proposition 3.1.44), while the zonotope enclosure of the relaxed set (green) is obtained without splitting.

5 Experiments

Finally, we show the results of our experiments using the MATLAB toolbox CORA (Althoff 2015) for the neural network verification and for all subsequent operations. All computations were performed on an Intel® Core™ Gen. 11 i7-11800H CPU @2.30GHz with 64GB memory. Our goal is to verify the networks without splitting the set, as splitting usually does not scale well with the dimension and is especially expensive in closed-loop settings (Lopez et al. 2022) due to the repeated evaluations of the neural network.

In our first experiment, we want to show the time improvement using our novel approach using an image (28×28 pixels) from the MNISTFC benchmark (Bak, Liu, and Johnson 2021). The neural network has 5 layers, 256 neurons in each hidden layer with ReLU activations and 10 output neurons. We increase the perturbation radius from 0.05 to 0.075

Approach	Split	Time	Result
Zonotope	✗	0.05s	UNKNOWN
Polynomial zonotope	✗	0.16s	UNKNOWN
Polynomial zonotope	✓	23.22s	VERIFIED
Our approach	✗	0.27s	VERIFIED

Table 1: Comparison of verification results using an image from the MNISTFC benchmark. The second column indicates whether the output set was recursively split.

such that it can no longer be verified using zonotopes. We use a neural network reduction technique (Ladner and Althoff 2023b) to construct a much smaller network with formal error bounds, where the verification of the reduced network entails the verification of the original network. Then, we apply the abstraction refinement approach using polynomial zonotopes (Ladner and Althoff 2023a, Alg. 3) on the reduced network. The specification is verified with three refinement steps, thus, resulting in three applications of the quadMap $(\cdot)$ operation (Prop. 1) creating dependencies between the monomials. Our novel exponent relaxation approach (Thm. 2) is applied using $\eta = 1$ before the final order reduction and checking against the specification. A comparison is shown in Tab. 1: The zonotope approach is as in (Singh et al. 2018) and the polynomial zonotope approach as in (Kochdumper et al. 2023). Our novel approach significantly speeds up the polynomial zonotope approach due to its polynomial time complexity and minor outer-approximations. The stated verification time of our approach also includes the time to restructure the polynomial zonotope.

In our second experiment, we analyze our approach more broadly regarding order reduction and subsequent enclosure operations, such as the zonotope enclosure, the interval bounds, and computing the support function in random directions. The neural networks Φ are generated randomly with varying sizes (number of layers κ, number of in-/output neurons $n_0 = n_\kappa$, number of hidden neurons), and we

Neural Network	OR	ZE		IB		SF	
Split Denominator	✓	✓	✗	✓	✗	✓	✗
– sigmoid –							
[2, 10, 2]	1.5905	3.1410	0.6977	2.0211	0.7311	1.5623	0.8325
[2, 10, 10, 2]	1.5421	3.1354	0.6181	2.0869	0.6600	1.4715	0.7821
[5, 10, 10, 5]	1.3258	3.9993	0.5253	36.3789	0.2400	1.9442	0.7145
[10, 20, 20, 10]	0.8578	1.0039	0.8066	1.6527	0.3827	0.9705	0.8953
[10, 20, 20, 20, 10]	0.8942	1.0308	0.8525	2.1742	0.5009	1.0063	0.9210
[10, 50, 50, 50, 10]	0.2909	0.2909	0.2568	0.0030	0.0009	0.5608	0.5056
– ReLU –							
[2, 10, 2]	2.5025	4.9590	0.3143	4.1043	0.3264	1.9836	0.5652
[2, 10, 10, 2]	1.3043	2.7052	0.5107	2.3776	0.5119	1.5761	0.7207
[5, 10, 10, 5]	0.9831	2.0318	0.7367	11.0079	0.5095	1.4071	0.8493
[10, 20, 20, 10]	1.4685	2.4273	0.2903	441.3038	0.0032	1.0696	0.5427
[10, 20, 20, 20, 10]	1.5591	2.4381	0.2934	263.2050	0.0026	1.0632	0.5276
[10, 50, 50, 50, 10]	2.0113	2.8469	0.1937	802.3418	0.0004	0.9019	0.4155

Table 2: Ratios of order reduction (OR), zonotope enclosure (ZE), interval bounds (IB), and support function (SF) using (6) on the output set $\mathcal{Y}$ of the respective neural network and $\eta = 1$. The second row indicates whether the value in the denominator of (6) is obtained via recursive splitting.

use intervals as input sets $\mathcal{X}$. We refine the abstraction five times, which creates the dependencies between the monomials. The output $\mathcal{Y} = \Phi(\mathcal{X})$ is used for our experiments. Each experiment is repeated 50 times, and we present the averaged results.

We want to stress that many operations on polynomial zonotopes are outer-approximative, thus, experiments need to be carefully crafted for valuable comparison. Additionally, the restructuring via exponent relaxation induces outer-approximation, however, we show the reduced outer-approximation of subsequently applied operations. To show that our novel approach produces tight outer-approximations with a subsequent operation ϕ on a polynomial zonotope $\mathcal{Y}$, we compute the ratio

$$\frac{\phi\left(\texttt{reduce}\left(\texttt{relax}\left(\mathcal{Y}, \eta\right)\right)\right)}{\phi\left(\texttt{reduce}\left(\mathcal{Y}\right)\right)}, \tag{6}$$

where $\texttt{reduce}\left(\cdot\right)$ computes an order reduction. For example, we compute $\phi(\cdot) = \texttt{vol}\left(\texttt{zonotope}\left(\cdot\right)\right)$ to measure the volume of the zonotope enclosure. Unfortunately, the volume is hard to compute for high-dimensional zonotopes and polynomial zonotopes. Thus, we project them to the two most dominant directions using singular value decomposition of the generator matrices prior to the volume computation. The volume of the interval bounds is computed considering all dimensions. Tab. 2 shows the averaged ratios in columns marked with a cross (✗). As all ratios are below 1, we obtain tighter results across all networks on average.

For comparison, in columns marked with a tick (✓), we show the ratio if the respective value in the denominator of (6) is obtained with recursive splitting. While it is expected that splitting obtains tighter results (ratio larger than 1), we can report that our approach often obtains very similar and, in some cases, even tighter results – with the main differences that our approach maintains a single set and runs in polynomial time complexity (Thm. 2).

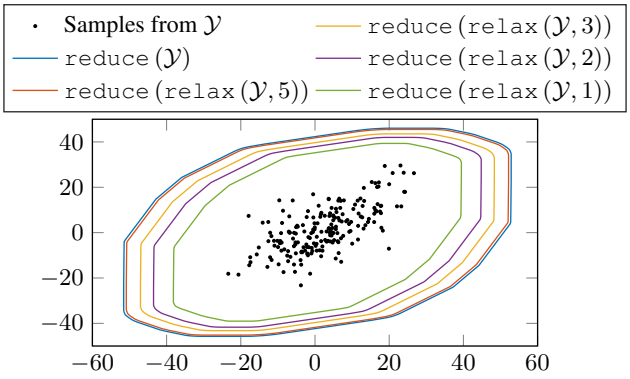

Figure 5: Improvements on the order reduction for different η using an output set $\mathcal{Y}$ of a neural network. Samples are taken from the unreduced output set $\mathcal{Y}$.

6 Conclusion

Polynomial zonotopes are used to verify neural networks due to their favorable computation time of the quadratic map operation. Unfortunately, the introduced dependencies between monomials of the generators make further processing challenging, such as intersection checking with an unsafe set. We present the first approach to consider these dependencies while restructuring the polynomial zonotope. As the resulting polynomial zonotope is simpler, subsequent operations such as order reduction and enclosure operations can be computed much tighter than without our novel approach. The restructuring has polynomial time complexity and thus can significantly decrease the verification time. While we focused on the verification of neural networks in this work, our approach is also applicable in other settings where polynomial zonotopes are used such as the verification of dynamic systems.

Acknowledgments

The authors gratefully acknowledge financial support from the project FAI funded by the German Research Foundation (DFG) under project number 286525601.

References

Althoff, M. 2013. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, 173–182.

Althoff, M. 2015. An introduction to CORA 2015. In *Proceedings of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 120–151.

Bak, S.; Liu, C.; and Johnson, T. 2021. The second international verification of neural networks competition (VNN-COMP 2021): Summary and results. *arXiv preprint arXiv:2109.00498*.

Bishop, C. M.; and Nasrabadi, N. M. 2006. *Pattern recognition and machine learning*, volume 4. Springer.

Bogomolov, S.; Forets, M.; Frehse, G.; Potomkin, K.; and Schilling, C. 2019. JuliaReach: A toolbox for set-based reachability. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 39–44.

Botoeva, E.; Kouvaros, P.; Kronqvist, J.; Lomuscio, A.; and Misener, R. 2020. Efficient verification of relu-based neural networks via dependency analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3291–3299.

Ferrari, C.; Muller, M. N.; Jovanovic, N.; and Vechev, M. 2022. Complete verification via multi-neuron relaxation guided branch-and-bound. *arXiv preprint arXiv:2205.00263*.

Girard, A. 2005. Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*, 291–305. Springer.

Goodfellow, I.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Henriksen, P.; and Lomuscio, A. 2020. Efficient neural network verification via adaptive refinement and adversarial search. In *European Conference on Artificial Intelligence*, 2513–2520. IOS Press.

Huang, Y.; Luo, E.; Bak, S.; and Sun, Y. 2023. On the difficulty of intersection checking with polynomial zonotopes. *arXiv preprint arXiv:2305.09901*.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, 97–117. Springer.

Katz, G.; Huang, D. A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljić, A.; et al. 2019. The Marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, 443–452. Springer.

Kiran, B. R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A. A.; Yogamani, S.; and Pérez, P. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23: 4909–4926.

Kochdumper, N. 2022. *Extensions of polynomial zonotopes and their application to verification of cyber-physical systems*. Ph.D. thesis, Technische Universität München.

Kochdumper, N.; and Althoff, M. 2020. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *IEEE Transactions on Automatic Control*, 4043–4058.

Kochdumper, N.; Schilling, C.; Althoff, M.; and Bak, S. 2023. Open-and closed-loop neural network verification using polynomial zonotopes. In *NASA Formal Methods Symposium*, 16–36. Springer.

Ladner, T.; and Althoff, M. 2023a. Automatic abstraction refinement in neural network verification using sensitivity analysis. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*, 18, 1–13.

Ladner, T.; and Althoff, M. 2023b. Specification-driven neural network reduction for scalable formal verification. *arXiv preprint arXiv:2305.01932*.

Lopez, D. M.; Althoff, M.; Benet, L.; Chen, X.; Fan, J.; Forets, M.; Huang, C.; Johnson, T. T.; Ladner, T.; Li, W.; et al. 2022. ARCH-COMP22 category report: Artificial intelligence and neural network control systems (AINNCS) for continuous and hybrid systems plants. In *Proceedings of 9th International Workshop on Applied*, volume 90, 142–184.

Schilling, C.; Forets, M.; and Guadalupe, S. 2022. Verification of neural-network control systems by integrating Taylor models and zonotopes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8169–8177.

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. 2018. Fast and effective robustness certification. *Advances in Neural Information Processing Systems*, 31.

Wang, S.; Zhang, H.; Xu, K.; Lin, X.; Jana, S.; Hsieh, C.-J.; and Kolter, J. Z. 2021. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34: 29909–29921.

Xu, K.; Zhang, H.; Wang, S.; Wang, Y.; Jana, S.; Lin, X.; and Hsieh, C.-J. 2020. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. *arXiv preprint arXiv:2011.13824*.