Robustness Certification for Point Cloud Models

Tobias Lorenz*[1], Anian Ruoss[2], Mislav Balunović[2], Gagandeep Singh[3], Martin Vechev[2]

[1]CISPA Helmholtz Center for Information Security
[2]Department of Computer Science, ETH Zurich
[3]University of Illinois at Urbana-Champaign and VMware Research

tobias.lorenz@cispa.de, ggnds@illinois.edu
{anian.ruoss, mislav.balunovic, martin.vechev}@inf.ethz.ch

Abstract

*The use of deep 3D point cloud models in safety-critical applications, such as autonomous driving, dictates the need to certify the robustness of these models to real-world transformations. This is technically challenging, as it requires a scalable verifier tailored to point cloud models that handles a wide range of semantic 3D transformations. In this work, we address this challenge and introduce 3DCertify, the first verifier able to certify the robustness of point cloud models. 3DCertify is based on two key insights: (i) a generic relaxation based on first-order Taylor approximations, applicable to any differentiable transformation, and (ii) a precise relaxation for global feature pooling, which is more complex than pointwise activations (*e.g.*, ReLU or sigmoid) but commonly employed in point cloud models. We demonstrate the effectiveness of 3DCertify by performing an extensive evaluation on a wide range of 3D transformations (*e.g.*, rotation, twisting) for both classification and part segmentation tasks. For example, we can certify robustness against rotations by $\pm 60°$ for 95.7% of point clouds, and our max pool relaxation increases certification by up to 15.6%.*

1. Introduction

Deep learning has achieved remarkable success in tasks involving 3D objects such as autonomous driving [8, 9, 26]. As such applications are typically safety-critical, recent work has investigated methods for quantifying the robustness of these systems via adversarial attacks, demonstrating the vulnerability of state-of-the-art point cloud models to semantic transformations and noise-based perturbations [28, 53, 60, 62, 66]. Another line of work introduced defenses [62, 65, 67], aiming to improve a models' robustness against these attacks. However, as demonstrated in the image recognition domain, such defenses are usually broken by more powerful attacks [1, 49], resulting in an arms race between stronger defenses and even stronger attacks.

To break this cycle, one ideally needs a *proof* that a deep learning model is robust against *any* adversarial attack, under some threat model. This proof is usually obtained by invoking a neural network verifier on a deep learning model and a model input, where the verifier attempts to provide a certificate that the model is robust to any transformation of this input. While plenty of verifiers have been proposed in the image recognition domain [5, 19, 42, 44, 48, 56, 64], no such verifier exists for 3D point cloud models.

This work: certification of 3D point clouds In this work we propose the first verifier for 3D point cloud models, called 3DCertify. As point cloud models are too complex for exact verification (*e.g.*, MILP [48]), the key challenge one must address is designing scalable and precise convex relaxations capturing all point clouds that could result from transforming the original (input) point cloud. In our work, we address this challenge and introduce such relaxations for a large family of common differentiable 3D transformations, including rotation, twisting, tapering, shearing, and arbitrary compositions of these transformations.

Robustness certification is achieved by propagating our 3D relaxations through the network using existing verifiers (*e.g.*, DeepPoly [44] or LiRPA [61]). This modular design allows our method to benefit from future advances in verification. For instance, we observe that existing verifiers incur a significant loss of precision at the max pool layer, often overlooked when designing verifiers for image classifiers, yet a critical feature of 3D point cloud models [36]. We address this issue by designing a more precise max pool relaxation which is more precise than existing solutions in practice while being applicable beyond point cloud models.

*This work was done while the author was at ETH Zurich.

Using 3DCertify, we are able to certify, for the first time, the robustness of the PointNet [36] model to semantic transformations on two challenging tasks: object classification on the ModelNet40 [59] dataset and part segmentation on the ShapeNet [7] dataset. We consider PointNet since, despite its relatively simple architecture, it performs well in terms of classification, segmentation, and certification, and provides the basis for more complex models [37, 52], which could be certified with future advances in verification.

Key contributions Our main contributions are:

- A novel framework based on first-order Taylor approximations for fast computation of linear relaxations of semantic 3D transformations of point clouds.
- A scalable linear relaxation for max pool that is provably more precise than prior work while being widely applicable to certification tasks beyond point clouds.
- The identification of inherent accuracy-robustness trade-offs in point cloud networks and a detailed ablation study of robustness-enhancing methods.
- The first robustness verifier of 3D point cloud models for object classification and part segmentation.
- A comprehensive experimental evaluation of our method on the PointNet [36] architecture and different datasets using a variety of semantic transformations. We make our implementation publicly available at `https://github.com/eth-sri/3dcertify`.

2. Background & Related Work

We now provide the necessary background on 3D point clouds and neural network certification, and give an overview of work closely related to ours in these areas.

3D point cloud models 3D data can be represented in several different ways (*e.g.*, 2D projections, 3D voxels, or 3D meshes), each with distinct advantages for specific applications. In this work, we consider 3D point clouds, which represent an object as an unordered set of points. Point clouds are sparse, requiring less space than, *e.g.*, voxels, and are the natural representation of 3D sensors such as LIDAR devices. The key challenge for neural networks handling point clouds is to be invariant to permutations in the order of 3D points. Qi *et al*. [36] were the first to address this challenge with their novel PointNet architecture, which aggregates local, pointwise features with a symmetric function (usually max pool) to obtain a permutation-invariant global feature vector that can be processed by any task-specific network, including classification and part segmentation models. In an effort to boost model performance, the PointNet architecture has been extended to include local information [37, 52], or rotation-invariant features [25].

Adversarial attacks on point cloud models Qi *et al*. [36] performed an initial investigation of the robustness of the PointNet architecture by introducing the concepts of critical points and upper bound shapes. They demonstrated that all sampled point clouds lying between critical points and upper bound shapes generate the same global feature vectors and thus obtain the same classification. However, this form of robustness quantification only holds for the concrete samples and does not consider adversarial transformations. The latter problem was addressed by a recent line of work that extended the well-studied problem of adversarial attacks for images to the 3D point cloud domain by considering adversarial point perturbation and generation [20, 23, 28, 29, 53, 60, 62], real-world adversarial objects for LIDAR sensors [6], occlusion attacks [55], and adversarial rotations [66]. The adversarial vulnerability of 3D point cloud models has spurred the development of corresponding defense methods, based on perturbation measurement [62], outlier removal and upsampling [67], and adversarial training [28, 65]. However, similar to the image setting, many of these adversarial defenses were later broken by stronger attacks [46], illustrating the need for provable robustness guarantees of 3D point cloud models.

Neural network certification Existing certifiers prove the robustness of image and NLP models: they compute a certificate which guarantees that no attack in a given range can change the predicted label (local robustness). These approaches generally focus on ℓ_p-norm threat models (*i.e.*, changing pixel intensities) and are based on the following three-step process: (i) compute a convex shape that encloses the space of all inputs obtainable from a given threat model, implying that the shape is *sound*, (ii) propagate the input shape through the network to obtain an output shape on the logits, and (iii) check that all concrete outputs within that output shape are classified to the correct class. For smaller networks, certification can be performed exactly with SMT solvers [19] or mixed-integer linear programming [48], but scaling to bigger networks requires computing an overapproximation of the output shape which can cause false positives, *i.e.*, the verifier fails to prove robustness even though it holds. A variety of such methods have been proposed based on semi-definite relaxations [38], linear relaxations [4, 5, 14, 27, 41, 43, 44, 51, 54, 56, 61, 64], or combinations of solvers and the above [34, 42, 45, 50]. These overapproximation methods compute convex relaxations for nonlinear network operations (*e.g.*, ReLU or max pool). The main challenge in the design of these relaxations is balancing their cost and precision. Certification can also be performed via randomized smoothing [10, 22, 40], which, however, provides probabilistic guarantees for a smoothed version of the original model and also incurs overhead during inference (as it requires additional sampling).

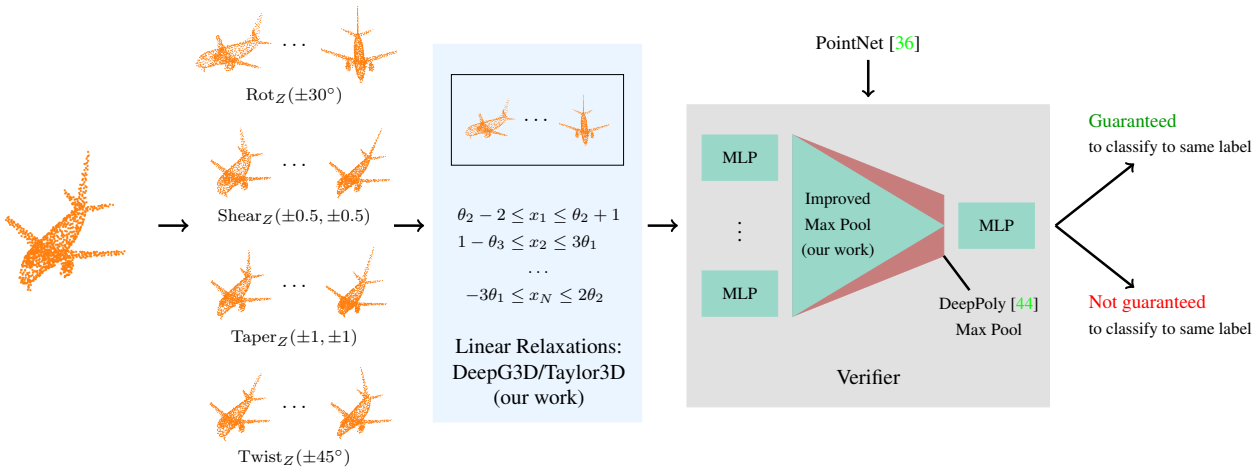

Figure 1. Overview of 3DCertify, consisting of two components: (i) a method to compute linear relaxations of 3D transformations for point clouds, and (ii) an improved network verifier. The first stage receives a point cloud and a transformation as input and computes a linear relaxation around all possible transformed point clouds. This relaxation is passed to the second stage, where the verifier tries to prove that a given network correctly classifies the input object under all (infinitely many) transformations. The verifier leverages our improved relaxation for the global max pool commonly employed in 3D models, enabling certification of significantly more objects than prior work.

Certification of semantic transformations Beside ℓ_p-norm threat models, neural network certification against semantic image transformations (*e.g.*, rotation or translation) have been recently considered. These approaches use enumeration [35], intervals [44], linear relaxations [2, 33, 39], or randomized smoothing [11, 24]. Inspired by randomized smoothing, concurrent work [30] computes probabilistic certificates for point cloud models against modifying a few individual points (addition or deletion), but cannot handle the semantic transformations considered in our work (*e.g.*, rotations or shearing). Another concurrent work [12] introduces an alternative approach to certify segmentation based on randomized smoothing. To the best of our knowledge, no prior certification method can handle the type of transformations on point cloud models which we consider. To achieve our goal, we leverage the high-level idea of DeepG [2], *i.e.*, we compute linear bounds on the transformed point positions in terms of the transformation parameters (*e.g.*, rotation angle). DeepG computes these bounds via a combination of sampling and optimization, which is asymptotically optimal but can unfortunately require exponential time for 3D transformations with many variables. Accordingly, this severely affects the practicality of the method, as we demonstrate experimentally. Thus, in our work, we first show how to generalize DeepG to the 3D point cloud setting and then introduce a more efficient (constant time) relaxation framework specifically tailored to point clouds. The asymptotic benefit is also reflected in practice, where we demonstrate 1000x speed-ups over the DeepG generalization, with minimal or no loss of precision.

We achieve end-to-end certification by providing our linear bounds as input to neural network verifiers, such as DeepPoly [44] and LiRPA [61], which work by overapproximating common functions (*e.g.*, ReLU) with a linear upper and lower bound for every output in terms of its inputs. To further improve on our results, we develop a novel max pool relaxation compatible with these verifiers (LiRPA does not support max pool, and DeepPoly's relaxation is imprecise as we will show experimentally). Finally, we show that our relaxations are generally applicable and of interest beyond the point cloud setting.

3. 3DCertify: 3D Point Cloud Verifier

We now present 3DCertify, our novel system for certifying the robustness of deep learning models for point clouds against perturbations on the input data. A high-level overview of 3DCertify is shown in Fig. 1. Given a neural network (*e.g.*, PointNet), a point cloud (*e.g.*, airplane), and a transformation (*e.g.*, $\pm 60°$ rotation), the goal of our system is to prove that all transformed point clouds are classified correctly by the network. The original point cloud and all considered transformations are shown in the left part of Fig. 1. Verification using our system consists of two main parts: (i) a general method for computing linear relaxations on a semantic transformation function, and (ii) an improved network verifier to certify the robustness of point cloud models based on these linear relaxations. Our system is highly modular: various convex relaxations can be used in the first part, and a wide range of network verifiers (both complete and incomplete) can be used in the second part.

For the first part of 3DCertify, we propose two different methods for computing linear relaxations of the transformation function, each with different benefits and drawbacks: (a) a generalization of DeepG [2] from images to 3D point clouds, and (b) a novel framework based on first-

order Taylor approximations. The DeepG generalization (Section 3.1) computes asymptotically optimal constraints but is more expensive, especially for transformations with many parameters. In contrast, our Taylor relaxations (Section 3.2) are fast to compute (in constant time), with minimal precision loss over the constraints from DeepG.

For the second part, we leverage the state-of-the-art verifiers DeepPoly [44] and LiRPA [61], which scale well to large networks in terms of computational complexity and certification performance. The key challenge in applying such verifiers to point cloud models is to precisely overapproximate the large, global max pool layer. Current 2D verifiers were designed for small pooling sizes and thus are imprecise for large pooling layers with thousands of inputs. We address this challenge by designing a more precise max pool relaxation, which we present in Section 3.3.

Notation We define a point cloud P of size n as $P = \{\boldsymbol{p}^{(j)} \mid \boldsymbol{p}^{(j)} \in \mathbb{R}^3,\ j \in \{1, 2, \dots, n\}\}$, where each $\boldsymbol{p}^{(j)} = (x, y, z)^T$ is a point in 3D space. We denote all vectors $\boldsymbol{v}$ in bold and their i-th entry as v_i. A point cloud transformation is a function which changes the position of the point cloud's points based on global transformation parameters. Formally, we define a transformation function as $f : \mathbb{R}^{3\times n} \times \mathbb{R}^k \to \mathbb{R}^{3\times n}$, where k is the number of parameters. f is applied to the point cloud prior to passing it to the neural network. Thus, with slight abuse of notation, given the network $N(P)$ and the transformation function $f(P, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^k$, we try to certify the robustness of N under transformation f for P by showing that the network output $N(f(P, \boldsymbol{\theta}))$ is invariant for a range of transformation parameters $\boldsymbol{l_\theta} \le \boldsymbol{\theta} \le \boldsymbol{u_\theta}$. To achieve this, we will compute precise linear relaxations of the transformation function $f(P, \boldsymbol{\theta})$, consisting of upper and lower constraints f_u and f_l. These constraints are linear in the transformation parameters $\boldsymbol{\theta}$, *i.e.*, $f_l(P, \boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{w_l} + b_l$ and $f_u(P, \boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{w_u} + b_u$, and must ensure that for the parameters $\boldsymbol{l_\theta} \le \boldsymbol{\theta} \le \boldsymbol{u_\theta}$ the inequality $f_l(P, \boldsymbol{\theta}) \le f(P, \boldsymbol{\theta}) \le f_u(P, \boldsymbol{\theta})$ holds (both point- and coordinatewise).

3.1. Generalizing DeepG

DeepG [2] computes sound and asymptotically optimal linear relaxations for any composition of semantic transformations, but is limited to the 2D image domain. Here, we show how to generalize DeepG to the 3D point cloud setting. We only show the necessary changes, treating the remaining parts of DeepG as a black box, and we refer interested readers to the original work for further details. To compute the linear relaxations, DeepG relies on a combination of sampling and optimization, and guarantees asymptotic optimality with increasing samples and tolerance parameter ϵ. The optimization procedure performs reverse-mode automatic differentiation and thus requires the Jacobian of each transformation, both with respect to inputs and parameters. Since each point $\boldsymbol{p} \in P$ is transformed independently of the other points, we can state the transformation in terms of $\boldsymbol{p}$ and apply it to each point individually. We provide the corresponding Jacobians for the 3D transformations rotation, shearing, tapering, and twisting in App. A.

3.2. Taylor Approximations

In the previous section, we showed that we can compute precise linear relaxations for point clouds by generalizing DeepG. However, DeepG needs to solve an optimization problem for each individual point coordinate. While this works well for quasilinear functions with a small parameter space, processing highly nonlinear functions with many parameters is exponentially more expensive. We therefore propose a novel, alternative framework to compute linear constraints in constant time based on first-order Taylor approximations. Our method can be applied to any twice continuously differentiable transformation function and also extends to compositions of multiple such transformations.

Linear approximation For any function $f(P, \boldsymbol{\theta})$ that is differentiable on the interval $\boldsymbol{l_\theta} \le \boldsymbol{\theta} \le \boldsymbol{u_\theta}$, the first-order Taylor polynomial [47] provides a linear approximation of f around the point $\boldsymbol{t} = (\boldsymbol{u_\theta} + \boldsymbol{l_\theta})/2$ with approximation error $R(P, \boldsymbol{\theta})$:

$$f(P, \boldsymbol{\theta}) = f(P, \boldsymbol{t}) + \sum_{i=1}^{k} (\theta_i - t_i) \frac{\partial f}{\partial \theta_i}(P, \boldsymbol{t}) + R(P, \boldsymbol{\theta}). \quad (1)$$

Our key insight is that we can use Eq. (1) to compute sound linear upper and lower bounds for $f(P, \boldsymbol{\theta})$, provided that we can bound the approximation error $R(P, \boldsymbol{\theta})$ with two constant terms $L_R \le R(P, \boldsymbol{\theta}) \le U_R$, for $L_R, U_R \in \mathbb{R}^{3\times n}$.

Error bounds For twice continuously differentiable functions, the Lagrange form of the approximation error is

$$R(P, \boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} (\theta_i - t_i)(\theta_j - t_j) \frac{\partial^2 f}{\partial \theta_i \partial \theta_j}(P, \boldsymbol{\xi}) \quad (2)$$

for some $\boldsymbol{\xi}$ between $\boldsymbol{t}$ and $\boldsymbol{\theta}$. Using the closed intervals $\boldsymbol{\theta} = [\boldsymbol{l_\theta}, \boldsymbol{u_\theta}]$ and $\boldsymbol{\xi} = [\boldsymbol{l_\theta}, \boldsymbol{u_\theta}]$ as inputs, we can compute the upper and lower bounds U_R and L_R for $R(P, \boldsymbol{\theta})$ using standard interval arithmetic. This means that we evaluate the functions on closed intervals instead of concrete values, resulting in an output interval containing the entire range of possible output values given the input intervals. Consequently, the effect of the individual functions on the input interval is generally overapproximated, so that the resulting interval bounds are not necessarily exact but always sound.

Therefore, the resulting upper linear constraint

$$f_u(P, \boldsymbol{\theta}) = f(P, \boldsymbol{t}) + \sum_{i=1}^{k} (\theta_i - t_i) \frac{\partial f}{\partial \theta_i}(P, \boldsymbol{t}) + U_R \quad (3)$$

and lower linear constraint

$$f_l(P, \boldsymbol{\theta}) = f(P, \boldsymbol{t}) + \sum_{i=1}^{k} (\theta_i - t_i) \frac{\partial f}{\partial \theta_i}(P, \boldsymbol{t}) + L_R \quad (4)$$

create a sound overapproximation of f on $[\boldsymbol{l_\theta}, \boldsymbol{u_\theta}]$.

Rotation example We demonstrate this computation with rotation around the z-axis Rot_Z, and we provide the relaxations for rotation around multiple axes, as well as shearing, tapering and twisting in App. A. The transformation function for rotation around the z-axis is

$$\mathrm{Rot}_Z(\boldsymbol{p}, \theta) = \begin{pmatrix} x\cos(\theta) - y\sin(\theta) \\ x\sin(\theta) + y\cos(\theta) \\ z \end{pmatrix}. \quad (5)$$

Recall that each point $\boldsymbol{p} \in P$ is transformed independently, and we can thus apply our method to each point individually. We consider the rotation interval $\theta = [-\pi/2, \pi/2]$, yielding $t = 0$ and the first-order Taylor approximation

$$\mathrm{Rot}_Z(\boldsymbol{p}, \theta) = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} -y \\ x \\ 0 \end{pmatrix} \theta + R(\boldsymbol{p}, \theta) \quad (6)$$

with approximation error

$$R(\boldsymbol{p}, \theta) = \frac{1}{2} \begin{pmatrix} -x\cos(\xi) + y\sin(\xi) \\ -x\sin(\xi) - y\cos(\xi) \\ 0 \end{pmatrix} \theta^2 \quad (7)$$

for $\xi \in [-\pi/2, \pi/2]$. To bound this term with constant bounds $L_R \leq R(\boldsymbol{p}, \theta) \leq U_R$, we evaluate the entire function by replacing $\xi = [-\pi/2, \pi/2]$ and $\theta = [-\pi/2, \pi/2]$ with their valid interval range:

$$[L_R, U_R] = \frac{1}{2} \begin{pmatrix} -x[0,1] + y[-1,1] \\ -x[-1,1] - y[0,1] \\ 0 \end{pmatrix} [0, \pi^2/4]. \quad (8)$$

Considering $\boldsymbol{p} = (1,1,1)^T$ as an example, we obtain

$$[l_R, u_R] = \begin{pmatrix} [-\pi^2/4, \pi^2/8] \\ [-\pi^2/4, \pi^2/8] \\ 0 \end{pmatrix}. \quad (9)$$

Composition of transformations Given twice continuously differentiable transformations $f(P, \boldsymbol{\phi})$ and $g(P, \boldsymbol{\psi})$, the composition $h(P, \boldsymbol{\theta}) = g(f(P, \boldsymbol{\phi}), \boldsymbol{\psi})$ is also twice continuously differentiable in $\boldsymbol{\theta} = (\boldsymbol{\phi}, \boldsymbol{\psi})^T$ (proof in App. B). Thus, we can directly compute its linear relaxation with the chain rule, using the first- and second-order partial derivatives of f and g. Our framework therefore naturally extends to the composition of multiple transformation functions.

3.3. Improved Max Pool Relaxation

Many deep learning models for point clouds, including PointNet, use a global or semi-global max pool layer to aggregate local, pointwise features into permutation-invariant global features. Consequently, these pooling layers operate on thousands of input variables, orders of magnitude more than image classification models. Since state-of-the-art relaxations for max pool are designed for the 2D case with few inputs, applying these methods out-of-the-box to point cloud models causes substantial precision loss (App. E).

Addressing the above problem by designing more precise linear constraints for max pool is inherently difficult, as it requires reasoning in much higher dimensions compared to most univariate activation functions (*e.g.*, ReLU). The state-of-the-art linear relaxations from DeepPoly for the max function $y = \max_i x_i$ over input neurons x_i with corresponding upper and lower bounds u_i and l_i are: $y \geq x_j$ for $j = \arg\max_i l_i$ as a lower bound and $y \leq u_{\max} = \max_i u_i$ as the upper bound. For the lower bound, any x_i would give a sound bound, since we know $y \geq x_i$. However, we have no such guarantees for the upper bound. DeepPoly simply uses the constant $u_{\max}$, which does not preserve relationship between neurons, causing significant precision loss. We therefore propose a scalable and more precise upper bound which preserves this relation.

As a first step, we check if we can prove that there exists an input neuron x_j which is always greater than the rest, that is: $x_j > x_{i \neq j}$. If this is the case, we simply return $y \leq x_j$ as the upper bound. For example, consider the simplified case $y := \max\{x_1, x_2\}$ with $x_1 \in [-1, -0.1]$ and $x_2 \in [0, 1]$. Then $x_1 < x_2$ for all inputs and our result $y = x_2$ is exact.

Otherwise, we compute an upper bound based on the convex hull of all possible cases $y = x_j, x_j \geq x_i$ for all possible j. In the two-variate example with $x_1 \in [-1, 0.25]$ and $x_2 \in [0, 1]$, we cannot prove $x_2 < x_1$ or $x_1 < x_2$ for all inputs and instead must consider both cases $\mathcal{S}_1 = \{y = x_1, x_1 \geq x_2, x_1 \in [-1, 0.25], x_2 \in [0, 1]\}$ and $\mathcal{S}_2 = \{y = x_2, x_2 \geq x_1, x_1 \in [-1, 0.25], x_2 \in [0, 1]\}$. Certifying all cases separately is prohibitive in practice, as it scales poorly with increasing input dimension and network depth. Instead, we compute the convex hull of $\mathcal{S}_1$ and $\mathcal{S}_2$ via the double description method [13], the state-of-the-art for high-dimensional convex hull computation. The double description method represents a polytope both with the set of constraints and vertices and uses a conversion algorithms to convert between the two representations.

For vertex representations $\mathcal{V}_1$ and $\mathcal{V}_2$ of $\mathcal{S}_1$ and $\mathcal{S}_2$, the convex hull vertices are $\mathcal{V}_1 \cup \mathcal{V}_2$, and the conversion computes output constraints $\mathcal{S} = \{y \in [0, 1], x_1 \in [-1, 0.25], x_2 \in [0, 1], y \geq x_1, y \geq x_2, y \leq 0.2 + 0.2x_1 + x_2, y \leq 0.25 + 0.75x_2\}$. Certification with $\mathcal{S}$ is faster than considering the two cases $\mathcal{S}_1$ and $\mathcal{S}_2$ separately, but it can still be expensive due to multiple linear constraints. There-

fore, we relax $\mathcal{S}$ further by computing its DeepPoly [44] relaxation, keeping the interval bounds and one upper and lower polyhedral constraint for y. Prior work does not compute the convex hull and obtains $y \in [0, 1]$ as interval bounds and $x_2 \leq y \leq 1$ as polyhedral bounds. Note that the upper polyhedral bound is the same as the upper interval bound. In contrast, we obtain polyhedral bounds $y \leq 0.2 + 0.2x_1 + x_2$ and $y \leq 0.25 + 0.75x_2$ from $\mathcal{S}$. To choose one among them, we compute the upper bound of the right-hand-side by replacing x_1 and x_2 with their interval bounds, obtaining 1.25 and 1, respectively. We choose $0.25 + 0.75x_2$ with the smaller right-hand-side as our polyhedral bound, and our result is $y \in [0, 1]$ and $x_2 \leq y \leq 0.25 + 0.75x_2$. While in this example our relaxation is strictly more precise than prior work, one cannot always theoretically establish increased precision. However, extensive experiments (Section 4.2) show that our method is significantly more precise in practice.

While this linear upper bound is more precise than state-of-the-art, it cannot be directly employed to certify point cloud models, since the complexity of convex hull computation grows exponentially in the number of input neurons, rendering the above computation infeasible for the thousands of neurons pooled in PointNet. Our key insight is to decompose the max pool operation into small groups of up to 10 neurons and to apply the max function recursively, making the computation tractable. For example, to take the max over 16 neurons $\{x_1, \ldots, x_{16}\}$, we first compute $y_1 = \max\{x_1, \ldots, x_8\}$ and $y_2 = \max\{x_9, \ldots, x_{16}\}$ and then merge both groups to obtain $y = \max\{y_1, y_2\}$. This allows us to scale our refined relaxation to large point cloud models, achieving significant improvements over the current state-of-the-art. We investigate the trade-off between running time and certification precision for different group sizes in App. D.1.

4. Experimental Evaluation

To illustrate the broad applicability of 3DCertify, we perform extensive experiments on different transformations, models, and tasks. Section 4.1 evaluates 3DCertify on a wide range of semantic transformations, comparing our generalization of DeepG, denoted as DeepG3D, and our Taylor framework, denoted as Taylor3D, to different baselines and analyzing the trade-off between our two methods. We investigate the impact of our improved max pool relaxation in Section 4.2, and we conduct an ablation study of the impact of various robustness-enhancing methods, such as adversarial training, in Section 4.3. Finally, we demonstrate the broad applicability of 3DCertify by performing the first robustness certification for part segmentation in Section 4.4. We make all of our code and models publicly available at https://github.com/eth-sri/3dcertify.

Experimental setup We use PointNet [36] models for all experiments since, despite their relatively simple architecture, they perform well for classification, part segmentation, and certification. Furthermore, PointNet is the basis for more complex, state-of-the-art models [37, 52], which could be certified with future verification advancements.

For classification, we evaluate 3DCertify on the 3D objects from the ModelNet40 [59] dataset, which are represented as point clouds and assigned to one of 40 different categories, such as tables or airplanes. We use the PointNet classification architecture introduced by Qi *et al.* [36] without T-Nets and perform experiments for different point cloud sizes, for which we train separate versions of the model. We apply the standard pre-processing pipeline [36] to the point clouds: centering, scaling to the unit sphere, and rotating randomly around the z(up)-axis. These normalizations render the model invariant to translation and scaling, which is why we do not consider these transformations, even though our system could easily handle them.

For certification of classification models, we report the percentage of correctly classified objects for which we can guarantee correct classification under all input transformations. All reported numbers are averages over a random subset of 100 objects from the test set (consistent with prior work [2, 33, 39]). We use the same random subset for all experiments, and, unless otherwise noted, we run all experiments on point clouds with 64 points using our improved max pool relaxation and the DeepPoly verifier [44].

Splitting To increase certification precision for transformations with few parameters (*e.g.*, rotation with one angle), one can split the parameter space into multiple smaller subsets (*e.g.*, split rotation by $[-\theta, \theta]$ into $[-\theta, 0]$ and $[0, \theta]$) and certify correct classification on each subset individually. Then, if we can certify correct classification for all subsets, we can infer correct classification for their union. While this technique has been successfully applied to extend certification to larger parameter ranges [2, 33, 44], it can only be employed for transformations with few parameters since the cost grows exponentially in the number of parameters. Both DeepG3D and Taylor3D naturally support splitting.

4.1. Semantic Transformations

Using 3DCertify, we evaluate the robustness of PointNet to several semantic transformations: rotation around one, two, and all three axes, shearing, twisting, and tapering, as well as compositions of these transformations. We compare the precision of our relaxations with two baselines: simple interval bounds, as well as refined bounds using implicit splitting introduced by Mohapatra *et al.* [33] with 15 625 splits, which we extend to the 3D domain. Finally, we analyze the trade-off between certification performance and computational complexity for DeepG3D and Taylor3D.

	Rot_Z		Rot_{ZX}		Rot_{ZYX}	
$\pm\theta$	20°	60°	5°	10°	2°	5°
Interval	70.7	54.3	6.5	3.3	1.1	0.0
Mohapatra *et al.* [33]	84.8	80.4	10.9	4.3	1.1	1.1
Ours (DeepG3D)	**96.7**	**95.7**	**89.1**	**73.9**	**72.8**	**58.7**
Ours (Taylor3D)	**96.7**	**95.7**	**89.1**	**73.9**	69.6	**58.7**

Table 1. Certification for rotation around one, two, and three axes.

Rotation Rotation is one of the most common transformations of 3D objects. In particular, an object's rotation around the up-axis is often arbitrary and thus any model should be robust to changes in upright orientation. Additionally, a model should not be fooled by small rotations around the other axes. We perform thorough experiments for robustness certification of different rotation angles around one, two, and all three axes, displaying the results in Table 1. All rotations are performed for $\pm\theta$, *i.e.*, the total rotation angle is 2θ, with splits of 2° along each dimension. For rotations around a single axis, both our relaxations enable certification of almost all objects for up to $\pm 60°$, which is equivalent to one third of all possible object orientations and significantly improves over both baselines. For two and three rotation axes, we certify robustness up to $\pm 5°$ and $\pm 3°$ respectively (again outperforming both baselines), since the number of splits required scales exponentially, as discussed above. Finally, to demonstrate that our system is independent of the concrete verifier, we also instantiate the LiRPA verifier [61] with Taylor3D to certify robustness against Rot_Z for $\pm 1°$ without splitting. DeepPoly certifies 97.8% and LiRPA certifies 95.7% of the objects, showing that 3DCertify can effectively certify robustness of PointNet independent of the concrete verifier used.

Additional transformations In addition to rotation, 3DCertify also handles shearing, twisting, and tapering, as well as compositions thereof. We compare certification accuracy and running time for DeepG3D and Taylor3D in Table 2 and observe that the relaxations are almost identical in terms of certification precision, although DeepG3D is slightly more precise in some cases such as $\mathrm{Twist} \circ \mathrm{Rot}_Z$. However, Taylor3D is orders of magnitude faster, particularly for transformations with multiple parameters. For example, Taylor3D requires 65ms to achieve the same certification for $\mathrm{Twist} \circ \mathrm{Taper} \circ \mathrm{Rot}_Z$ as DeepG3D, which needs 65447ms (a 1000x speed-up). Unlike DeepG3D, Taylor3D could thus be employed in settings requiring instant certification feedback (*e.g.*, real-time applications). We further demonstrate the effortless scaling of Taylor3D to real-world point clouds with up to 300k points [15] in App. D.2.

4.2. Improved Max Pool Relaxation

Scaling certification to large point clouds poses significant challenges for verifiers due to the much larger number of neurons involved in global feature pooling, underlining the importance of a precise max pool relaxation such as the one we introduced in Section 3.3. We compare our new max pool relaxation to the DeepPoly relaxation [44], as it is the basis upon which we improve. We also integrate the linear relaxation proposed by Boopathy *et al.* [5] into DeepPoly as an additional baseline. We compare certification for rotation around one axis for $\pm 3°$ without splitting on point clouds of different sizes in Table 3. Our improved relaxation provides the best results in all settings, improving by up to 15.6% over the current state-of-the-art and with particular benefits for large point cloud sizes, which are essential in the context of 3D point cloud models. We further demonstrate the broad applicability of our max pool relaxation with experiments on computer vision models in App. D.1.

		Certified (%)		Time (ms)	
$f(P,\theta)$	$\pm\theta$	Taylor3D	DeepG3D	Taylor3D	DeepG3D
Rot_{ZYX}	1°	73.9	73.9	8.87	391
	1.5°	8.7	8.7	6.88	405
Shear	0.02	93.5	93.5	0.07	230
	0.03	70.7	70.7	0.07	228
Taper	0.1	81.5	81.5	0.46	232
	0.2	28.3	28.3	0.47	232
Twist	10	76.1	76.1	0.70	209
	20	23.9	23.9	0.64	211
$\mathrm{Twist} \circ \mathrm{Rot}_Z$	10, 1°	57.6	60.9	3.04	580
	20, 1°	7.6	16.3	3.21	515
$\mathrm{Taper} \circ \mathrm{Rot}_Z$	0.1, 1°	69.6	68.5	4.64	531
	0.2, 1°	20.7	20.7	4.54	918
$\mathrm{Twist} \circ \mathrm{Taper} \circ \mathrm{Rot}_Z$	10, 0.1, 1°	20.7	20.7	64.08	32216
	20, 0.2, 1°	5.4	5.4	65.04	65447

Table 2. Certification of different transformations and compositions thereof. DeepG3D provides slightly more precise certification, while Taylor3D is orders of magnitude more efficient.

Points	16	32	64	128	256	512	1024
Boopathy *et al.* [5]	3.7	3.6	3.3	2.2	4.4	5.6	6.7
DeepPoly [44]	95.1	**94.0**	91.3	72.2	51.1	39.3	28.1
Ours (Taylor3D)	**97.5**	**94.0**	**93.5**	**81.1**	**66.7**	**49.4**	**37.1**

Table 3. Certification of Rot_Z with $\theta = \pm 3°$ for different max pool relaxations. Our relaxation is significantly more precise, particularly for large point clouds (*i.e.*, large pooling layers).

4.3. Boosting Certified Robustness

It is well established that certified robustness of image classification models can be significantly increased via adversarial and provable training [3, 17, 32, 58, 63]. Here, we generalize such robust training methods to the 3D point cloud domain. Moreover, we identify that common point cloud network components lead to substantial trade-offs in terms of accuracy and certified robustness, providing essential insights to guide future architecture design.

In the 2D domain, adversarial threat models are generally formulated in terms of ℓ_p-norms quantifying the pixel intensity perturbation. Such ℓ_p-norm threat models extend naturally to the 3D point cloud setting, where they capture

Global Pooling	Training	Accuracy (%) 64	Accuracy (%) 256	Certified (%) 64	Certified (%) 256
Max Pool	Natural	**85.7**	**86.1**	0.0	0.0
	Adversarial	84.4	85.6	22.2	3.2
	Provable	77.6	79.1	**91.4**	**84.7**
Average Pool	Natural	81.9	84.0	0.0	0.0
	Adversarial	80.1	80.7	47.7	43.7
	Provable	72.7	72.6	85.3	76.7

Table 4. Ablation study for robustness-enhancing methods with pointwise ℓ_∞-perturbations on 64 and 256 points with $\epsilon = 0.01$.

the spatial perturbation of every point. We consider the ℓ_∞-norm, which restricts perturbations to an ϵ-box around the original point coordinates and corresponds to the commonly studied Hausdorff distance in the 3D adversarial attack literature [29, 53, 60, 62]. Note that perturbations shift each point independently from all others, unlike *e.g.*, rotation.

We conduct an ablation study for various robustness-enhancing methods, including adversarial (FGSM) [16, 57] and provable (IBP) [17] training, and using different symmetric feature-aggregation functions (average pool and max pool). We use the fast adversarial training variant of FGSM by Wong *et al*. [57] since it is orders of magnitude faster than PGD [31] (and thus applicable to point cloud models) but provides similar performance in terms of certified robustness. We compare the accuracy and certified robustness in Table 4 and observe that average pool leads to a notable drop in accuracy (roughly 5%), confirming the results by Qi *et al*. [36]. While naturally trained models are not provably robust against perturbations, adversarial training significantly improves the robustness, particularly for average pool, which DeepPoly can encode exactly regardless of the point cloud size. In contrast, certification with max pool is challenging (particularly for larger point clouds) due to its nonlinearity – even with our improved relaxation. Provable training further increases robustness, although less for average pool than for max pool since IBP training uses intervals that are less precise for average pool than for max pool.

Running the same experiment with max pool and IBP training for point clouds with 2048 points results in a model with 77.4% accuracy and 94.9% certification, confirming the scalability of our approach to large point cloud sizes.

4.4. Part Segmentation

In addition to object classification, we consider the safety-relevant task of part segmentation. For part segmentation, a model tries to predict which part of an object a point belongs to, *e.g.*, the wings of an airplane or the legs of a chair. We show that 3DCertify, powered by our relaxations, is the first certification system to successfully handle this task, proving its usefulness beyond object classification.

For part segmentation, we use the ShapeNet-Part [7] dataset, which contains different classes of 3D objects with part annotations for each point. We train a part segmentation version of PointNet [36] on 64 points, with an IoU score of 0.82. The model architecture is shown in App. C. Table 5 shows the percentage of correctly classified points remaining invariant under Rot_Z with $5°$ splits, showing that DeepG3D and Taylor3D can certify robustness for most points, even for large rotation angles up to $\pm 10°$. Moreover, the method scales to larger point clouds with 1024 points, with 95.5% of points certifiably robust to rotations of $\pm 5°$.

$\pm\theta$	Interval	Mohapatra *et al*. [33]	DeepG3D	Taylor3D
5°	44.1	50.5	**96.6**	96.5
10°	32.2	46.5	**95.7**	95.6

Table 5. Ratio of certified points for Rot_Z on part segmentation.

4.5. Discussion

Point cloud models achieve high accuracy on natural datasets, and 3DCertify shows that such models are also highly robust against 1D rotations, even for large angles. However, in line with our results, Zhao *et al*. [66] showed that 3D rotations achieve a 95% attack success rate with angles as small as $\pm 2.81°$, implying that certification on the same model cannot scale beyond such small angles. In this light, our results for 3D rotations of $\pm 5°$ are meaningful though they highlight a need for further research on robust architectures. Moreover, our guarantees for ℓ_∞-perturbations of $\epsilon = 0.01$, roughly 1% of the object's size, indicate that 3DCertify is applicable beyond semantic transformations. Our robust training results demonstrate that max pool is the best symmetric feature-aggregation function in terms of accuracy and provable robustness, confirming its importance for point cloud models. Finally, we note that current certifiers based on linear relaxations cannot compute efficient bounds for PointNet's T-Nets, which are known to boost the model's accuracy [36]. We thus consider investigating approaches to certify T-Nets without sacrificing precision an interesting direction for future work.

5. Conclusion

We presented 3DCertify, the first scalable verifier able to certify robustness of 3D point cloud models against a wide range of semantic 3D transformations including rotations, shearing, twisting and tapering. The key insight of 3DCertify is a novel method which enables efficient and precise computation of linear relaxations for these transformations. Combined with an improved relaxation for max pool layers, our extensive evaluation on two datasets with object classification and part segmentation illustrates the effectiveness and broad applicability of 3DCertify.

Acknowledgements This work was partially supported by the ERC Starting Grant 680358. We thank the anonymous reviewers for their valuable feedback.

References

[1] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. 1

[2] Mislav Balunovic, Maximilian Baader, Gagandeep Singh, Timon Gehr, and Martin T. Vechev. Certifying geometric robustness of neural networks. In *Advances in Neural Information Processing Systems 32*, 2019. 3, 4, 6, 15

[3] Mislav Balunovic and Martin T. Vechev. Adversarial training and provable defenses: Bridging the gap. In *8th International Conference on Learning Representations*, 2020. 7

[4] Gregory Bonaert, Dimitar I. Dimitrov, Maximilian Baader, and Martin T. Vechev. Fast and precise certification of transformers. In *42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2021. 2

[5] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, 2019. 1, 2, 7

[6] Yulong Cao, Chaowei Xiao, Dawei Yang, Jing Fang, Ruigang Yang, Mingyan Liu, and Bo Li. Adversarial objects against lidar-based autonomous driving systems. *CoRR*, abs/1907.05418, 2019. 2

[7] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015. 2, 8

[8] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington. 3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. *IEEE Signal Process. Mag.*, 38, 2021. 1

[9] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1

[10] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. 2

[11] Marc Fischer, Maximilian Baader, and Martin T. Vechev. Certified defense to image transformations via randomized smoothing. In *Advances in Neural Information Processing Systems 33*, 2020. 3

[12] Marc Fischer, Maximilian Baader, and Martin T. Vechev. Scalable certified segmentation via randomized smoothing. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. 3

[13] Komei Fukuda and Alain Prodon. Double description method revisited. In *Combinatorics and Computer Science*, 1995. 5

[14] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. AI2: safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy*, 2018. 2

[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *Int. J. Robotics Res.*, 32, 2013. 7, 15

[16] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations*, 2015. 8

[17] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Arthur Mann, and Pushmeet Kohli. Scalable verified training for provably robust image classification. In *IEEE/CVF International Conference on Computer Vision*. IEEE, 2019. 7, 8

[18] Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. 15

[19] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification - 29th International Conference*, 2017. 1, 2

[20] Itai Lang, Uriel Kotlicki, and Shai Avidan. Geometric adversarial attacks and defenses on 3d point clouds. *CoRR*, abs/2012.05657, 2020. 2

[21] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 14

[22] Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy*, 2019. 2

[23] Kibok Lee, Zhuoyuan Chen, Xinchen Yan, Raquel Urtasun, and Ersin Yumer. Shapeadv: Generating shape-aware adversarial 3d point clouds. *CoRR*, abs/2005.11626, 2020. 2

[24] Linyi Li, Maurice Weber, Xiaojun Xu, Luka Rimanic, Tao Xie, Ce Zhang, and Bo Li. Provable robust learning based on transformation-specific smoothing. *CoRR*, abs/2002.12398, 2020. 3

[25] Xianzhi Li, Ruihui Li, Guangyong Chen, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. A rotation-invariant framework for deep point cloud analysis. *CoRR*, abs/2003.07238, 2020. 2

[26] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Computer Vision - 15th European Conference*, 2018. 1

[27] Wang Lin, Zhengfeng Yang, Xin Chen, Qingye Zhao, Xiangkun Li, Zhiming Liu, and Jifeng He. Robustness verification of classification deep neural networks via linear programming. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[28] Daniel Liu, Ronald Yu, and Hao Su. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. In *IEEE International Conference on Image Processing*, 2019. 1, 2

[29] Daniel Liu, Ronald Yu, and Hao Su. Adversarial shape perturbations on 3d point clouds. In *Computer Vision - ECCV Workshops*, 2020. 2, 8

[30] Hongbin Liu, Jinyuan Jia, and Neil Zhenqiang Gong. Pointguard: Provably robust 3d point cloud classification. *CoRR*, abs/2103.03046, 2021. 3

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations*, 2018. 8

[32] Matthew Mirman, Timon Gehr, and Martin T. Vechev. Differentiable abstract interpretation for provably robust neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. 7

[33] Jeet Mohapatra, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Towards verifying robustness of neural networks against A family of semantic perturbations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 3, 6, 7, 8

[34] Mark Niklas Müller, Gleb Makarchuk, Gagandeep Singh, Markus Püschel, and Martin T. Vechev. Precise multi-neuron abstractions for neural network certification. *CoRR*, abs/2103.03638, 2021. 2

[35] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Towards practical verification of machine learning: The case of computer vision systems. *CoRR*, abs/1712.01785, 2017. 3

[36] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 3, 6, 8, 13, 14

[37] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30*, 2017. 2, 6

[38] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems 31*, 2018. 2

[39] Anian Ruoss, Maximilian Baader, Mislav Balunovic, and Martin T. Vechev. Efficient certification of spatial robustness. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021. 3, 6

[40] Hadi Salman, Jerry Li, Ilya P. Razenshteyn, Pengchuan Zhang, Huan Zhang, Sébastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems 32*, 2019. 2

[41] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems 32*, 2019. 2

[42] Gagandeep Singh, Rupanshu Ganvir, Markus Püschel, and Martin T. Vechev. Beyond the single neuron convex barrier for neural network certification. In *Advances in Neural Information Processing Systems 32*, 2019. 1, 2

[43] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems 31*, 2018. 2

[44] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3, 2019. 1, 2, 3, 4, 6, 7, 14

[45] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. Boosting robustness certification of neural networks. In *7th International Conference on Learning Representations*, 2019. 2

[46] Jiachen Sun, Karl Koenig, Yulong Cao, Qi Alfred Chen, and Z. Morley Mao. On the adversarial robustness of 3d point cloud classification. *CoRR*, abs/2011.11922, 2020. 2

[47] Brook Taylor. *Methodus incrementorum directa & inversa / auctore Brook Taylor.* Typis Pearsonianis: prostant apud Gul. Innys, 1715. 4

[48] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *7th International Conference on Learning Representations*, 2019. 1, 2

[49] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems 33*, 2020. 1

[50] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems 31*, 2018. 2

[51] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In *27th USENIX Security Symposium*, 2018. 2

[52] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.*, 38, 2019. 2, 6

[53] Yuxin Wen, Jiehong Lin, Ke Chen, and Kui Jia. Geometry-aware generation of adversarial and cooperative point clouds. *CoRR*, abs/1912.11171, 2019. 1, 2, 8

[54] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S. Boning, and Inderjit S. Dhillon. Towards fast computation of certified robustness for relu networks. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. 2

[55] Matthew Wicker and Marta Kwiatkowska. Robustness of 3d deep learning in an adversarial setting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[56] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. 1, 2

[57] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *8th International Conference on Learning Representations*, 2020. 8

[58] Eric Wong, Frank R. Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems 31*, 2018. 7

[59] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2, 6

[60] Chong Xiang, Charles R. Qi, and Bo Li. Generating 3d adversarial point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 8

[61] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. In *Advances in Neural Information Processing Systems 33*, 2020. 1, 2, 3, 4, 7

[62] Jiancheng Yang, Qiang Zhang, Rongyao Fang, Bingbing Ni, Jinxian Liu, and Qi Tian. Adversarial attack and defense on point sets. *CoRR*, abs/1902.10899, 2019. 1, 2, 8

[63] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane S. Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *8th International Conference on Learning Representations*, 2020. 7

[64] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems 31*, 2018. 1, 2

[65] Yu Zhang, Gongbo Liang, Tawfiq Salem, and Nathan Jacobs. Defense-pointnet: Protecting pointnet against adversarial attacks. In *IEEE International Conference on Big Data*, 2019. 1, 2

[66] Yue Zhao, Yuwei Wu, Caihua Chen, and Andrew Lim. On isometry robustness of deep 3d point cloud models under adversarial attacks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2, 8

[67] Hang Zhou, Kejiang Chen, Weiming Zhang, Han Fang, Wenbo Zhou, and Nenghai Yu. Dup-net: Denoiser and upsampler network for 3d adversarial point clouds defense. In *IEEE/CVF International Conference on Computer Vision*, 2019. 1, 2

A. 3D Transformations and Their Relaxations

In this section, we define the semantic transformations that we consider (App. A.1) and provide the taylor relaxations for the transformations (App. A.2), as well as their Jacobians for DeepG3D (App. A.4). We also give some background on the interval arithmetic used to compute bounds on the approximation error (App. A.3).

A.1. Semantic Transformations

3DCertify can handle a wide range of semantic transformations, including 3D rotation around any axis with $\theta \in \mathbb{R}$ as defined in Eq. (5). We can also certify shearing, twisting, and tapering of a point cloud, defined pointwise (since each point is transformed independently) for a point $\boldsymbol{p} = (x, y, z)^T$ as

$$\text{Shear}(\boldsymbol{p}, \boldsymbol{\theta}) = \begin{pmatrix} \theta_1 z + x \\ \theta_2 z + y \\ z \end{pmatrix}$$

$$\text{Twist}(\boldsymbol{p}, \theta) = \begin{pmatrix} x\cos(\theta z) - y\sin(\theta z) \\ x\sin(\theta z) + y\cos(\theta z) \\ z \end{pmatrix}$$

$$\text{Taper}(\boldsymbol{p}, \boldsymbol{\theta}) = \begin{pmatrix} (\frac{1}{2}\theta_1^2 z + \theta_2 z + 1)x \\ (\frac{1}{2}\theta_1^2 z + \theta_2 z + 1)y \\ z \end{pmatrix}$$

or any composition of these functions.

A.2. Taylor Relaxations

In Section 3.2, we presented the general form of our linear bounds $f_l(P, \boldsymbol{\theta})$ and $f_u(P, \boldsymbol{\theta})$ for any twice continuously differentiable transformation function $f(P, \boldsymbol{\theta})$, as well as the relaxation for Rot_Z as an example. Rotation around the other two axes, *i.e.*, Rot_X and Rot_Y can be computed analogously. Here, we list the linear relaxations for the remaining transformation functions we use in our experiments.

All transformations can be applied to each point individually, allowing us to denote them for a single point as $f(\boldsymbol{p}, \boldsymbol{\theta})$ with $\boldsymbol{p} = (x, y, z)^T \in P$. For each transformation, we list the first-order taylor polynomial $Q(\boldsymbol{p}, \boldsymbol{\theta})$ and remainder $R(\boldsymbol{p}, \boldsymbol{\theta})$, such that $f(\boldsymbol{p}, \boldsymbol{\theta}) = Q(\boldsymbol{p}, \boldsymbol{\theta}) + R(\boldsymbol{p}, \boldsymbol{\theta})$. As described in Section 3.2, we use interval arithmetic (App. A.3) to get real-valued bounds $\boldsymbol{l}_R \leq R(\boldsymbol{p}, \bar{\boldsymbol{\theta}}) \leq \boldsymbol{u}_R$ for the interval $\bar{\boldsymbol{\theta}} = [\boldsymbol{l}_{\boldsymbol{\theta}}, \boldsymbol{u}_{\boldsymbol{\theta}}]$ with $\boldsymbol{t} = (\boldsymbol{l}_{\boldsymbol{\theta}} + \boldsymbol{u}_{\boldsymbol{\theta}})/2$ and therefore the lower constraint $f_l(\boldsymbol{p}, \boldsymbol{\theta}) = Q(\boldsymbol{p}, \boldsymbol{\theta}) + \boldsymbol{l}_R$ and upper constraint $f_u(\boldsymbol{p}, \boldsymbol{\theta}) = Q(\boldsymbol{p}, \boldsymbol{\theta}) + \boldsymbol{u}_R$.

Shearing:

$$\begin{aligned} Q_{\text{Shear}}(\boldsymbol{p}, \boldsymbol{\theta}) &= \begin{pmatrix} t_1 z + x \\ t_2 z + y \\ z \end{pmatrix} \\ &\quad + \begin{pmatrix} z \\ 0 \\ 0 \end{pmatrix} (\theta_1 - t_1) + \begin{pmatrix} 0 \\ z \\ 0 \end{pmatrix} (\theta_2 - t_2) \\ R_{\text{Shear}}(\boldsymbol{p}, \bar{\boldsymbol{\theta}}) &= 0 \end{aligned}$$

Twisting:

$$\begin{aligned} Q_{\text{Twist}}(\boldsymbol{p}, \theta) &= \begin{pmatrix} \cos(tz)x - \sin(tz)y \\ \sin(tz)x + \cos(tz)y \\ z \end{pmatrix} \\ &\quad + \begin{pmatrix} -z(\sin(tz)x + \cos(tz)y) \\ z(\cos(\theta z)x - \sin(\theta z)y) \\ 0 \end{pmatrix} (\theta - t) \\ R_{\text{Twist}}(\boldsymbol{p}, \bar{\theta}) &= \frac{1}{2} \begin{pmatrix} -z^2(\cos(\bar{\theta} z)x - \sin(\bar{\theta} z)y) \\ -z^2(\sin(\bar{\theta} z)x + \cos(\bar{\theta} z)y) \\ 0 \end{pmatrix} (\bar{\theta} - t)^2 \end{aligned}$$

Tapering:

$$\begin{aligned} Q_{\text{Taper}}(\boldsymbol{p}, \boldsymbol{\theta}) &= \begin{pmatrix} (\frac{1}{2}t_1^2 z + t_2 z + 1)x \\ (\frac{1}{2}t_1^2 z + t_2 z + 1)y \\ z \end{pmatrix} \\ &\quad + \begin{pmatrix} t_1 zx \\ t_1 zy \\ 0 \end{pmatrix} (\theta_1 - t_1) + \begin{pmatrix} zx \\ zy \\ 0 \end{pmatrix} (\theta_2 - t_2) \\ R_{\text{Taper}}(\boldsymbol{p}, \bar{\boldsymbol{\theta}}) &= \frac{1}{2} \begin{pmatrix} zx \\ zy \\ 0 \end{pmatrix} (\bar{\theta_1} - t_1)^2 \end{aligned}$$

A.3. Interval Arithmetic

When evaluating functions such as $R(P, \boldsymbol{\theta})$ on intervals, we use the following standard operators:

$$\begin{aligned} -[x_l, x_u] &= [-x_u, -x_l] \\ [x_l, x_u] + [y_l, y_u] &= [x_l + y_l, x_u + y_u] \\ [x_l, x_u] - [y_l, y_u] &= [x_l - y_u, x_u - y_l] \\ [x_l, x_u] \cdot [y_l, y_u] &= [\min(x_l y_l, x_l y_u, x_u y_l, x_u y_u), \\ &\qquad \max(x_l y_l, x_l y_u, x_u y_l, x_u y_u)] \end{aligned}$$

For mixed operations with scalars, *i.e.*, $a * [x_l, x_u]$, we can treat the scalar as an interval with one element: $[a, a] * [x_l, x_u]$.

In addition to these basic operators, we use the following

sine function:

$$\sin([x_l, x_u]) = [y_l, y_u], \text{ where}$$

$$y_l = \begin{cases} -1 & -\frac{\pi}{2} + 2k\pi \in [x_l, x_u] \\ \min(\sin(x_l), \sin(x_u)) & \text{otherwise} \end{cases}$$

$$y_u = \begin{cases} 1 & \frac{\pi}{2} + 2k\pi \in [x_l, x_u] \\ \max(\sin(x_l), \sin(x_u)) & \text{otherwise} \end{cases}$$

with $k \in \mathbb{Z}$. Similarly, we can define the cosine function:

$$\cos([x_l, x_u]) = [y_l, y_u], \text{ where}$$

$$y_l = \begin{cases} -1 & \pi + 2k\pi \in [x_l, x_u] \\ \min(\cos(x_l), \cos(x_u)) & \text{otherwise} \end{cases}$$

$$y_u = \begin{cases} 1 & 2k\pi \in [x_l, x_u] \\ \max(\cos(x_l), \cos(x_u)) & \text{otherwise.} \end{cases}$$

To compute the square x^2 of $x = [x_l, x_u]$, we could simply use $x \cdot x$. While sound, we can compute a tighter interval for some cases:

$$[x_l, x_u]^2 = \begin{cases} [x_l^2, x_u^2] & x_l \geq 0 \\ [x_u^2, x_l^2] & x_u \leq 0 \\ [0, \max(x_l^2, x_u^2)] & \text{otherwise.} \end{cases}$$

Using these operators and functions, we can evaluate all of our relaxations and their derivatives with intervals as input.

A.4. Jacobian Matrices

Computing linear relaxations using Deepg3D (Section 3.1) requires the Jacobians of our 3D transformations, both with respect to the transformation parameters and with respect to the point cloud inputs. For example, for 3D rotation around the z-axis with $\theta \in \mathbb{R}$, as defined in Eq. (5), we compute

$$\partial_{\boldsymbol{p}} \mathrm{Rot}_Z(\boldsymbol{p}, \theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{10}$$

and

$$\partial_{\theta} \mathrm{Rot}_Z(\boldsymbol{p}, \theta) = \begin{pmatrix} -x\sin(\theta) - y\cos(\theta) \\ x\cos(\theta) - y\sin(\theta) \\ z \end{pmatrix}. \tag{11}$$

The corresponding Jacobians for Shear, Twist and Taper (Section 4.1) are given by:

$$\partial_{\boldsymbol{p}} \mathrm{Shear}(\boldsymbol{p}, \boldsymbol{\theta}) = \begin{pmatrix} 1 & 0 & \theta_1 \\ 0 & 1 & \theta_2 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\partial_{\boldsymbol{\theta}} \mathrm{Shear}(\boldsymbol{p}, \boldsymbol{\theta}) = \begin{pmatrix} z & 0 \\ 0 & z \\ 0 & 0 \end{pmatrix}$$

$$\partial_{\boldsymbol{p}} \mathrm{Twist}(\boldsymbol{p}, \theta) = \begin{pmatrix} \cos(\theta z) & -sin(\theta z) & -\theta(\sin(\theta z)x + \cos(\theta z)y) \\ \sin(\theta z) & \cos(\theta z) & \theta(\cos(thetaz)x - \sin(\theta z)y) \\ 0 & 0 & 1 \end{pmatrix}$$

$$\partial_{\theta} \mathrm{Twist}(\boldsymbol{p}, \theta) = \begin{pmatrix} -z(\sin(\theta z)x + \cos(\theta z)y) \\ z(\cos(\theta z)x - \sin(\theta z)y) \\ 0 \end{pmatrix}$$

$$\partial_{\boldsymbol{p}} \mathrm{Taper}(\boldsymbol{p}, \boldsymbol{\theta}) = \begin{pmatrix} \frac{1}{2}\theta_1^2 z + \theta_2 z + 1 & 0 & (\frac{1}{2}\theta_1^2 + \theta_2)x \\ 0 & \frac{1}{2}\theta_1^2 z + \theta_2 z & (\frac{1}{2}\theta_1^2 + \theta_2)y \\ 0 & 0 & 1 \end{pmatrix}$$

$$\partial_{\boldsymbol{\theta}} \mathrm{Taper}(\boldsymbol{p}, \boldsymbol{\theta}) = \begin{pmatrix} \theta_1 zx & zx \\ \theta_1 zy & zy \\ 0 & 0 \end{pmatrix}$$

B. Proof for Composition of Transformations

To show that our taylor approximations introduced in Section 3.2 can be applied to the composition of multiple transformations, we show that the composition of two twice continuously differentiable functions is itself twice continuously differentiable. That is, given two twice continuously differentiable functions $f : \mathbb{R}^n \mapsto \mathbb{R}^p$ and $g : \mathbb{R}^m \mapsto \mathbb{R}^n$, we want to show that $h = f \circ g$ is also twice continuously differentiable.

To simplify notation, we define $\boldsymbol{y} = f(\boldsymbol{u})$ and $\boldsymbol{u} = g(\boldsymbol{x})$. Using the chain rule, we know that the first-order derivatives exist and can, with slight liberties in notation, be written as:

$$\frac{\partial \boldsymbol{y}}{\partial x_i} = \sum_k \frac{\partial \boldsymbol{y}}{\partial u_k} \frac{\partial u_k}{\partial x_i}. \tag{12}$$

Furthermore, we know that f and g are twice differentiable, hence Eq. (12) consists of compositions, products and sums of differentiable functions and thus is again differentiable. We therefore conclude that $f \circ g$ is itself twice differentiable.

It remains to be shown that the second-order derivatives are continuous. Using Faà di Bruno's formula, we can write the second-order derivatives as:

$$\frac{\partial^2 \boldsymbol{y}}{\partial x_i \partial x_j} = \sum_k \frac{\partial y}{\partial u_k} \frac{\partial^2 u_k}{\partial x_i \partial x_j} + \sum_k \sum_l \frac{\partial^2 \boldsymbol{y}}{\partial u_k \partial u_l} \frac{\partial u_k}{\partial x_i} \frac{\partial u_l}{\partial x_j}. \tag{13}$$

We know that f, g and their first- and second-order derivatives are continuous. Equation (13) is therefore a combination of compositions, products and sums of continuous functions, which means it is itself continuous. This means $f \circ g$ is twice continuously differentiable and we can therefore calculate Taylor bounds for any composition of twice continuously differentiable transformations.

C. PointNet Architectures

For both object classification and part segmentation, we use PointNet [36] models. Below we present the exact layer configurations used.

Object Classification

For object classification, we use the following network architecture:

No	Type	Normalization	Activation	Features
1	Linear	BatchNorm	ReLU	64
2	Linear	BatchNorm	ReLU	64
3	Linear	BatchNorm	ReLU	64
4	Linear	BatchNorm	ReLU	128
5	Linear	BatchNorm	ReLU	1024
6	MaxPool			1024
7	Linear	BatchNorm	ReLU	512
8	Linear	BatchNorm	ReLU	256
9	Linear		SoftMax	num classes

The first block of linear (fully connected) layers (no 1 to 5) is executed on each point individually, but sharing weights across all points. We implement this via 1D convolution layers with stride 1 as in the original work by Qi *et al*. [36]. Layer 6 pools each feature across points. During training, a dropout of 30% is applied for layer 8.

Part Segmentation

No	Type	Normalization	Activation	Features
1	Linear	BatchNorm	ReLU	64
2	Linear	BatchNorm	ReLU	128
3	Linear	BatchNorm	ReLU	256
4	Linear	BatchNorm		128
5	MaxPool			128
6	Repeat			128
7	Concatenate (1, 2, 3, 6)			576
8	Linear	BatchNorm	ReLU	256
9	Linear	BatchNorm	ReLU	128
10	Linear		SoftMax	num parts

The architecture for part segmentation differs in some ways, since it needs to predict a label for each point individually. Again, the first block of linear layers (1-4) is applied to each point individually with shared weights and max pool combines per-point features to one global feature vector. Layer 7 concatenates the local features of layers 1 to 3 with the global feature from layer 5 for each point by simple concatenation. The last 3 linear layers are again applied individually for each point on the combined local and global feature and predict the part the particular point belongs to.

Since DeepPoly [44] cannot handle this architecture for part segmentation, we implement novel relaxations for the concatenation and repeat layers. In particular, the transformer for concatenation requires the verifier to handle layers with multiple predecessors, which is out of reach for current state-of-the-art verifiers. We provide our implementation in the accompanying code.

ϵ	0.005	0.010	0.015
DeepPoly [44]	72.8	33.3	3.7
Ours	**79.0**	**38.3**	**6.2**

Table 6. Percentage of certified images with different max pool relaxations for two different image classification tasks for ℓ_∞ noise perturbations.

Group Size	Certified (%)	Time (s)
4	93.5	56
8	93.5	76
12	93.5	119

Table 7. Percentage of certified point clouds with 64 points for different max pool group sizes for $\pm 3°$ rotation.

D. Additional Experiments

In this section, we present additional empirical evidence for the benefits of our improved max pool relaxations in App. D.1, and investigate the effect different max pool group sizes have on certification results. We also show that Taylor3D efficiently scales to real-world point cloud sizes in App. D.2, with running times of only a few milliseconds.

D.1. Improved Max Pool Relaxation

Applications beyond point clouds In Section 4.2, we show that our improved max pool relaxation, introduced in Section 3.3, significantly improves certification for PointNet models compared to the previous state-of-the-art, especially for models with larger pooling layers. Here, we demonstrate that our new relaxations are useful beyond PointNet, *i.e.*, for any network architecture containing max pool layers. To that end, we show certification results for a convolutional image classification model with nine conv/-linear layers and two max pool layers for the MNIST [21] dataset in Table 6, comparing our improved relaxations with the best baseline. Our improved max pool relaxations consistently outperform the previous state-of-the-art across all ϵ-values, significantly increasing the number of images for which we can certify correct classification. These results demonstrate that models beyond the 3D point cloud domain benefit from our new relaxations.

Max pool group size Our improved max pool relaxation, introduced in Section 3.3, requires computing the convex hull of the polyhedral relaxation, for which the running time grows exponentially in the number of input neurons. This

	Rot_Z		Twist	
Points	Taylor3D	DeepG3D	Taylor3D	DeepG3D
100 000	0.036	393	0.070	366
200 000	0.070	887	0.169	848
300 000	0.104	1502	0.266	1496

Table 8. Running time in seconds to compute the relaxations for different real-world point cloud sizes with Taylor3D and DeepG3D. Taylor3D achieves speed-ups of up to 14 442x for Rot_Z and 5 624x for Twist.

is why we recursively split the max pool operation into sub groups. Table 7 shows the certification accuracy and average running time of DeepPoly with the improved max pool relaxation for different group sizes. Increasing the group size beyond this range is impractical (*i.e.*, more than 3h per point clouds) due to the exponential scaling behavior of convex hull computation. Nevertheless, our experiments indicate that our recursively partitioned relaxation is not impeded by this constraint since the different group sizes do not influence certification performance (while, in theory, computing the relaxation over all inputs should be most precise), allowing us to optimize for improved running time.

D.2. Scaling

3D processing of LIDAR point cloud data is an active area of research. The main challenge is the huge size of point clouds (in the order of 100k points [15]) that have to be processed in real-time for most applications. Both DeepG3D and Taylor3D scale linearly with point cloud size and can be parallelized perfectly across points. Table 8 shows the running time of computing linear relaxations for large point cloud sizes using Taylor3D and Deepg3D respectively. All experiments are run on an AMD EPYC 7601 processor with 2.2 GHz. Taylor3D is efficiently implemented as vectorized operations using Numpy [18] and run on a single thread. DeepG3D uses the original parallelized implementation [2] and runs in parallel with 16 threads. The results show that, while both implementations scale linearly in the point cloud size, Taylor3D is significantly more efficient, computing relaxations in only a few milliseconds event for large point cloud sizes on a single thread, thereby achieving speed-ups of up to 14 442x over DeepG3D. This enables easy and efficient scaling to real-world applications.

E. Max Pool Analysis

The state-of-the-art linear relaxations for max pool are imprecise, especially for many inputs to the pooling layer. We demonstrate this by plotting the mean divergence between DeepPoly's upper and lower bounds for each of PointNet's layers in Fig. 2, where we observe that the bounds start to significantly diverge after the max pool layer.

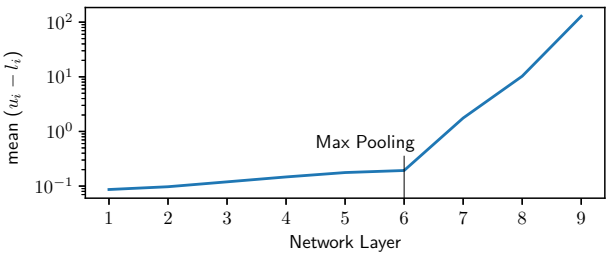

Figure 2. Plotting the mean difference between upper and lower bounds of neurons after each layer shows that the precision significantly decreases after the max pool and therefore motivates the need for improvement. Note the logarithmic scaling of the y-axis.