Robustness Verification of Semantic Segmentation Neural Networks Using Relaxed Reachability

Hoang-Dung Tran[1(✉)], Neelanjana Pal[2], Patrick Musau[2], Diego Manzanas Lopez[2], Nathaniel Hamilton[2], Xiaodong Yang[2], Stanley Bak[3], and Taylor T. Johnson[2]

[1] University of Nebraska-Lincoln, Lincoln, USA
[2] Vanderbilt University, Nashville, USA
[3] Stony Brook University, Stony Brook, USA

Abstract. This paper introduces robustness verification for semantic segmentation neural networks (in short, semantic segmentation networks [SSNs]), building on and extending recent approaches for robustness verification of image classification neural networks. Despite recent progress in developing verification methods for specifications such as local adversarial robustness in deep neural networks (DNNs) in terms of scalability, precision, and applicability to different network architectures, layers, and activation functions, robustness verification of semantic segmentation has not yet been considered. We address this limitation by developing and applying new robustness analysis methods for several segmentation neural network architectures, specifically by addressing reachability analysis of up-sampling layers, such as transposed convolution and dilated convolution. We consider several definitions of robustness for segmentation, such as the percentage of pixels in the output that can be proven robust under different adversarial perturbations, and a robust variant of intersection-over-union (IoU), the typical performance evaluation measure for segmentation tasks. Our approach is based on a new relaxed reachability method, allowing users to select the percentage of a number of linear programming problems (LPs) to solve when constructing the reachable set, through a relaxation factor percentage. The approach is implemented within NNV, then applied and evaluated on segmentation datasets, such as a multi-digit variant of MNIST known as M2NIST. Thorough experiments show that by using transposed convolution for up-sampling and average-pooling for down-sampling, combined with minimizing the number of ReLU layers in the SSNs, we can obtain SSNs with not only high accuracy (IoU), but also that are more robust to adversarial attacks and amenable to verification. Additionally, using our new relaxed reachability method, we can significantly reduce the verification time for neural networks whose ReLU layers dominate the total analysis time, even in classification tasks.

A. Silva and K. R. M. Leino (Eds.) CAV 2021, LNCS 12759, pp. 263–286, 2021.
https://doi.org/10.1007/978-3-030-81685-8_12

1 Introduction

Image segmentation is the process of partitioning an image into multiple portions, or segments, which are sets of pixels, and in short is referred to as segmentation [30]. Segmentation has broad applications, ranging from perception in autonomous cyber-physical systems (e.g., identifying pedestrians, lanes, vehicles, etc. in images) and medical imaging (e.g., identifying tumors, measuring tissue, etc. in X-rays and other medical scans) [31]. *Semantic segmentation* additionally classifies each pixel into a *class* from a set of classes, and hence, can be viewed as a generalization of image classification, the robustness of which has been studied deeply in recent years.

State-of-the-art segmentation approaches typically rely on neural networks, known as *semantic segmentation networks (SSNs)*. Typically SSN architectures take an image as input and are composed of two major portions: a sequence of *down-sampling* layers to extract features from the input image into a latent space, followed by another sequence of *up-sampling* layers, which in essence map the features (roughly corresponding to the classes) from the latent space to the image's pixels, such that each pixel is associated with a class. However, just as neural networks for image classification are well-known to be vulnerable to adversarial perturbations, so too are SSNs [45]. Although deep neural networks (DNN) verification is emerging into an established research area with many tools and techniques proposed to verify safety and robustness specifications of DNNs [22,43] and neural network controlled systems [15,17,34,37], most state-of-art verification techniques for robustness verification of DNNs focus on variants of classification[1], frequently for images [1,5,7,11,19,24,26,29,32,33,46].

To our knowledge, there are no existing methods that can verify robustness of SSNs, which perform a more complex task than image classification, as the output space dimensionality is (typically) of the same order of size as that of the input space (e.g., the output is an image with the width and height of the input image, but with identified classes in the output instead of color bit depth; see Figs. 5 and 7 for examples). We review some existing testing-based robustness evaluation methods in our related work section.

Overview and Contributions. In this paper, we present the first formal approach for verifying SSN robustness using reachability analysis. Our approach's central idea is, if an input image is attacked (perturbed) with some bounded disturbance, we construct a reachable output set that contains all possible classes for each pixel. From the reachable output set, we can formally guarantee an SSN's robustness at the pixel-level, i.e., each pixel is provably classified correctly. Our approach focuses on two effective SSN architectures, including dilated CNNs and transposed CNNs, which to our knowledge, are not supported in any other existing neural network verification approaches. We evaluate our approach on a

[1] The ACAS-Xu benchmarks [18] common in neural network verification can be viewed as a form of classifier: the networks produce advisories (weak left, etc.), which in essence are classes, but do not have images as their inputs.

set of SSNs trained with different architectures on the MNIST [21] and M2NIST data sets, the latter of which is a multi-digit variant of MNIST suitable for segmentation evaluation. Additionally, we define and evaluate several metrics for robustness, as the robustness evaluation is more sophisticated for segmentation.

Our reachability-based approach builds on ImageStars, which are an efficient data structure for verifying convolutional neural networks (CNNs) [33], to construct the input set and compute the reachable set layer-by-layer throughout the SSN. The ImageStar approach offers both exact and approximate reachability schemes for analyzing the robustness of CNNs. Although the approximate scheme obtains a tighter reachable set in comparison with the zonotope [28] and new polytope methods [29] by using optimized ranges, in practice, we do not need a tight reachable set in many cases. Indeed, we only need a "tight enough" reachable set to verify a property. Therefore, it is reasonable to let users have the freedom to choose an appropriate level of relaxation in constructing the reachable set for their applications. More relaxation comes with a coarser reachable set and vice versa. To fulfill this need, we also present a new relaxed ImageStar approach to allow users to choose a specific relaxation level defined by a *relaxation factor* (RF) percentage when constructing the reachable set for their applications. This relaxed reachability method can help reduce the verification time of SSNs significantly (up to 99%) in some cases.

In summary, the main contributions of this paper are: 1) the first formal approach for robustness verification of SSNs, 2) a new relaxed ImageStar reachability method, 3) the implementation of the approach in a prototype software tool, 4) thorough assessment of these methods on different network architectures, and 5) insight on how to train robust SSNs that are amenable to verification.

2 Preliminaries and Problem Formulation

2.1 ImageStars

In this section, we review the ImageStar data structure and its properties [33].

Definition 1. *An* ImageStar Θ *is a tuple* $\langle c, V, P, l, u\rangle$ *where* $c \in \mathbb{R}^{h\times w\times nc}$ *is the* anchor image, $V = \{v_1, v_2, \cdots, v_m\}$ *is a set of* m *images in* $\mathbb{R}^{h\times w\times nc}$ *called* generator images, $P : \mathbb{R}^m \rightarrow \{\top, \bot\}$ *is a* predicate, l *and* u *are the* lower bound *and* upper bound *vectors of the* predicate variables, *and* h, w, nc *are the* height, width, *and* number of channels *of the images, respectively. The generator images are arranged to form the ImageStar's* $h \times w \times nc \times m$ basis array. *The set of images represented by the ImageStar is given as:*

$$[\![\Theta]\!] = \{x \mid x = c + \Sigma_{i=1}^{m}(\alpha_i v_i) \text{ such that } P(\alpha_1, \cdots, \alpha_m) = \top, l_i \leq \alpha_i \leq u_i\}.$$

We may refer to both the tuple Θ *and the set of states* $[\![\Theta]\!]$ *as* Θ. *In this work, we restrict the predicates to be a conjunction of linear constraints,* $P(\alpha) \triangleq C\alpha \leq d$ *where, for* p *linear constraints,* $C \in \mathbb{R}^{p\times m}$, α *is the vector of* m*-variables, i.e.,* $\alpha = [\alpha_1, \cdots, \alpha_m]^T$, *and* $d \in \mathbb{R}^{p\times 1}$. *An ImageStar is the empty set if and only if* $P(\alpha)$ *subject to* $l \leq \alpha \leq u$ *is empty.*

Lemma 1 (Affine mapping of an ImageStar). *An affine mapping of an ImageStar* $\Theta = \langle c, V, P, l, u \rangle$ *with a scale factor* γ *and an offset image* β *is another ImageStar* $\Theta' = \langle c', V', P', l', u' \rangle$ *in which the new anchor, generators and predicate are as follows:*

$$c' = \gamma \cdot c + \beta, \quad V' = \gamma \cdot V, \quad P' \equiv P, \quad l' \equiv l, \quad u'. \equiv u.$$

Note that, the scale factor γ *can be a scalar or a vector containing scalar scale factors in which each factor is used to scale one color channel in the ImageStar.*

2.2 Range of a Specific Input in an ImageStar

We slightly alter the original definition of an ImageStar, [33], by introducing lower bound and upper bound vectors to the predicate variables. Specifically, if we want to find the range of an input $x(i, j, k)$ (where $1 \leq i \leq h$, $1 \leq j \leq w$, $1 \leq k \leq nc$) in an ImageStar Θ, we need to solve the following LP problem.

$$x_{min} = \min(c(i,j,k) + \Sigma_{p=1}^{m} \alpha_i v_p(i,j,k)) \; s.t. \; C\alpha \leq d, l \leq \alpha \leq u, \tag{1}$$

$$x_{max} = \max(c(i,j,k) + \Sigma_{p=1}^{m} \alpha_i v_p(i,j,k)) \; s.t. \; C\alpha \leq d, l \leq \alpha \leq u. \tag{2}$$

However, if we only want to estimate roughly the range of the neuron without solving the LP optimization problem, we can compute the estimated range quickly as follows.

$$x_{min}^{est} = c(i,j,k) + \Sigma_{p=1}^{m} l_p \max(v_p(i,j,k), 0) + \Sigma_{q=1}^{m} u_q \min(v_q(i,j,k), 0), \tag{3}$$

$$x_{max}^{est} = c(i,j,k) + \Sigma_{p=1}^{m} u_p \max(v_p(i,j,k), 0) + \Sigma_{q=1}^{m} l_q \min(v_q(i,j,k), 0). \tag{4}$$

2.3 Semantic Segmentation Networks and Reachability

Definition 2. *A* semantic segmentation network (SSN) f *is a nonlinear function that maps each pixel* $x(i, j)$ *of a multichannel input image* x *to a target class* $y(i, j)$ *from a set of classes* $\mathcal{L} = \{1, 2, \ldots, L\}$*:*

$$\begin{aligned} f: \; x \in \mathbb{R}^{h \times w \times nc} &\rightarrow y \in \mathcal{L}^{h \times w} \\ x(i,j) &\rightarrow y(i,j) \in \mathcal{L}, \end{aligned} \tag{5}$$

where h, w, nc *are the height, width, and number of channels of the input image, respectively, and* $(i, j) \in \{1, \ldots, h\} \times \{1, \ldots, w\}$ *are the pixel height and width indices, respectively.*

Definition 3. *Reachability analysis (or shortly, Reach) of a SSN* f *on an ImageStar input set* I *is the process of computing all possible classes corresponding to every pixel in all input images* x *in the ImageStar input set* I*:*

$$\begin{aligned} Reach(f, I): \; I &\rightarrow \mathcal{R}_f \\ x &\rightarrow y = f(x). \end{aligned} \tag{6}$$

We call $\mathcal{R}_f(I)$ *the* pixel-class reachable set *of the SSN corresponding to the input set* I *(or just* $\mathcal{R}_f$ *when* I *is clear from context), in which each pixel-class* $pc(i,j) \in \mathcal{R}_f$ *at each pixel* $(i,j) \in \{1,\ldots,h\} \times \{1,\ldots,w\}$ *may contain more than one class, i.e.,* $pc(i,j) = \{l_1,\ldots,l_m\} \subseteq \mathcal{L}$*, for* $L \geq m \geq 1$.

2.4 Adversarial Attacks and Robustness

Definition 4. *An adversarial attack is where a set of* n *noise images* $x_{noise} = [x_1^{noise}, \ldots, x_n^{noise}]$ *and corresponding coefficient vector* $\epsilon = [\epsilon_1,\ldots,\epsilon_n]^T$ *are added to input image* x *to change the classification result of a network.*

Mathematically, an adversarial attack is a linear parameterized function $g_{\epsilon,x^{noise}}(\cdot)$ *that takes an image as an input and produces the corresponding adversarial image.*

$$x^{adv} = g_{\epsilon,x^{noise}}(x) = x + \Sigma_{i=1}^{n} \epsilon_i \cdot x_i^{noise} \tag{7}$$

In this paper, we focus on the robustness analysis of SSNs under adversarial attacks. We refer readers to [45] for a survey of state-of-art attack and defenses approaches, mostly for classification.

Definition 5. *An* unknown, bounded adversarial attack (UBAA) *is an adversarial attack where the value of the coefficient vector* ϵ *is unknown but bounded in a range* $[\underline{\epsilon}, \overline{\epsilon}]$*, i.e.,* $\underline{\epsilon_i} \leq \epsilon_i \leq \overline{\epsilon_i}$*. An UBAA can be defined formally as a tuple* $\mathcal{A} = \langle \underline{\epsilon}, \overline{\epsilon}, x^{noise} \rangle$.

Proposition 1 (UBAA as an ImageStar). *Applying an UBAA* $\mathcal{A} = \langle \underline{\epsilon}, \overline{\epsilon}, x^{noise} \rangle$ *on an image* x *creates a set of images, which can be represented as an ImageStar* $I = \langle c \equiv x, V \equiv x^{noise}, P(\alpha) \equiv P(\epsilon) \equiv \underline{\epsilon} \leq \epsilon \leq \overline{\epsilon} \rangle$.

Definition 6. *Given a SSN* f *and an input image* x*, a pixel* $x(i,j) \in x$ *is called* robust *to an UBAA* $\mathcal{A}$ *if and only if:* $\forall\ g_{\epsilon,x^{noise}} \in \mathcal{A}$*,* $f(x^{adv}(i,j)) = f(x(i,j))$*, where* $x^{adv}(i,j) \in x^{adv} = g_{\epsilon,x^{noise}}(x)$*. If* $\exists\ g_{\epsilon,x^{noise}} \in \mathcal{A}$ *such that* $f(x^{adv}(i,j)) \neq f(x(i,j))$*, the pixel* $x(i,j)$ *is called* non-robust.

Definition 7. *The* robustness value (RV) *of a SSN corresponding to an UBAA applied to an input image is defined as* $RV = \frac{N_{robust}}{N_{pixels}} \times 100\%$*, where* N_{robust} *is the total number of robust pixels under the attack, and* $N_{pixels} = h \cdot w$ *is the total number of pixels of the input image.*

Definition 8. *The* robustness sensitivity (RS) *of a SSN corresponding to an UBAA applied to an input image is defined as* $RS = \frac{N_{nonrobust} + N_{unknown}}{N_{attackedpixels}}$*, where* $N_{nonrobust}$ *is the total number of non-robust pixels under the attack,* $N_{unknown}$ *is the total number of pixels whose robustness is unknown (may or may not be robust), and* $N_{attackedpixels}$ *is the total number of attacked pixels of the input image.*

Definition 9. *The* robust IoU (Intersection-over-Union) *(*R_{IoU}*) of a SSN corresponding to an UBAA applied to an input image is defined as the average IoU of all labels that are robust under the attack. Let* x *be a segmentation*

ground-truth image, y be the verified segmentation image under the attack, and IoU_p *be the IoU (also known as Jaccard index) of the* p^{th} *label in the label images* x *and* y*, then the robust IoU of the SSN is computed by:*

$$R_{IoU} = \frac{\Sigma_{p=1}^{L} IoU_p}{L}. \tag{8}$$

The robust IoU definition is quite similar to traditional IoU, which is a core metric to evaluate the accuracy in training SSNs. However, instead of assessing the accuracy, we use the robust IoU concept in combination with the robustness value and robustness sensitivity as core metrics to evaluate the robustness of a SSN under adversarial attack in the verification context.

2.5 Robustness Verification Problem Formulation

We consider two robustness verification problems.

Problem 1. Given a SSN f, an image x, and an UBAA $\mathcal{A}$, prove for every pixel $x(i,j) \in x$ that $x(i,j)$ is robust or non-robust to the attack $\mathcal{A}$.

Problem 2. Given a SSN f, a set of N test images $\{x_1, \ldots, x_N\}$, and an UBAA $\mathcal{A}$, compute the average robustness value $\overline{RV}$, the average robustness sensitivity $\overline{RS}$, and the average robust IoU $\overline{R}_{IoU}$ of the SSN (corresponding to $\mathcal{A}$).

The core step in solving these problems is to prove the robustness of a SSN f under an UBAA $\mathcal{A}$ at the pixel-level, i.e., Problem 1, which can be solved using reachability analysis computing the "pixel-class reachable set" $\mathcal{R}_f = Reach(f, I)$ that contains all possible classes of every pixel in the input set I constructed by applying the attack $\mathcal{A}$ on an image x (Proposition 1). Next, we investigate a new relaxed ImageStar reachability method for the ReLU layer, the up-sampling layers, including transposed convolution, dilated convolution, and pixel-classification. We note that the softmax layer can be neglected in the analysis [33].

3 Reachability of SSNs Using Relaxed ImageStars

In this section, we build on the original ImageStar method to develop reachability analysis for the transposed convolution and dilated convolution layers, and propose a new relaxed ImageStar reachability method for the ReLU and pixel-classification layers. The reachability algorithms for other layers can be handled using existing methods, such as those in [33]. Thus, we highlight handling the up-sampling layers, which requires overcoming significant challenges, and has not previously been done. Handling up-sampling layers is necessary for SSN robustness verification.

3.1 Reachability of a Transposed (Dilated) Convolutional Layer

Transposed (dilated) convolutions are frequently used for up-sampling in image segmentation applications to generate an output feature map that has a spatial

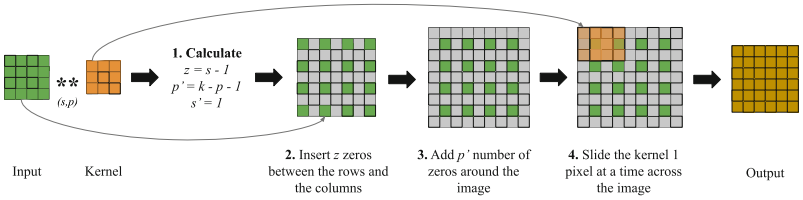


Fig. 1. An example of a transposed convolution operation.

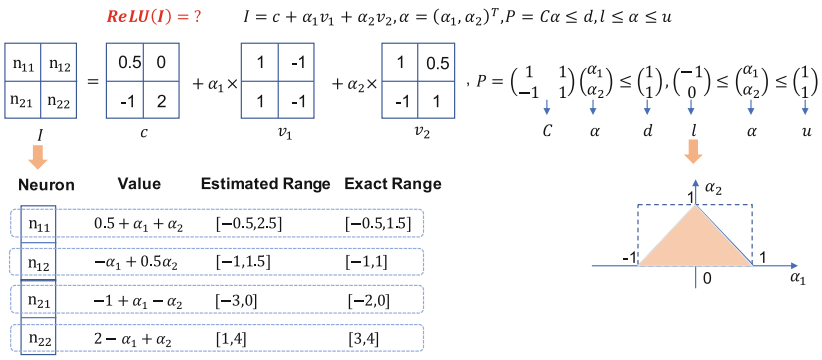


Fig. 2. Example 1.

dimension greater than that of the input feature map. A transposed convolution operation consists of four main steps, depicted in Fig. 1, and is defined by its kernel size k, padding p, and stride s. A dilated convolution operation is defined by its kernel size k, padding p, stride s and dilation factor d.

Lemma 2. *The reachable set of a transposed (dilated) convolutional layer with an ImageStar input set $\mathcal{I} = \langle c, V, P \rangle$ is another ImageStar, specifically $\mathcal{I}' = \langle c', V', P \rangle$ where $c' = TConv(c)$ ($c' = DConv(c)$) is the transposed (dilated) convolution operation applied to the anchor image, $V' = \{v'_1, \ldots, v'_m\}$, $v'_i = TConvZeroBias(v_i)$ ($v'_i = DConvZeroBias(v_i)$) is the transposed (dilated) convolution operation with zero bias applied to the generator images, i.e., using only the weights of the layer. Each of these are affine operations, see [30] for details, and as shown in Lemma 1, ImageStars are closed under affine operations.*[2]

3.2 Relaxed Reachability of a ReLU Layer

In this section, we present the relaxed ImageStar reachability of a ReLU layer. Like the original approximate reachability method [33], the relaxed ImageStar approach computes an overapproximate reachable set of a ReLU layer. However,

[2] In most neural network frameworks, transposed and dilated convolution are implemented as convolution with particular choices of padding, stride, and dilation factor as illustrated in Fig. 1 for transposed convolution, which is well-known to be affine.

it allows users to construct a "tight enough" reachable set sufficient to prove properties for their applications via a user-specified relaxation factor scaled from 0% to 100% that reduces verification time. In this paper, we focus on this process for ReLU layers. We use a small example depicted in Fig. 2 to illustrate the reachability of a ReLU layer using the relaxed ImageStar method. In this example, we have a 2×2 (4 neurons) ImageStar input set I with the anchor image c and two generator images v_1 and v_2, and we want to compute an overapproximation of $ReLU(I)$. To do that, we apply the triangle overapproximation rule [10,36] for the ReLU activation function at each neuron of the input set in the following.

Lemma 3. *For any input* $x \in [l,\ u]$*, the output set* $Y = \{y|\ y = ReLU(x)\}$ *satisfies: (1) If* $l \geq 0$*, then* $y = x$*; (2) If* $u \leq 0$*, then* $y = 0$*; or (3) If* $l < 0$ *and* $u > 0$*, then* $Y \subset \bar{Y} = \{y|\ y \geq 0,\ y \leq \frac{u(x-l)}{u-l},\ y \geq x\}$.

Using the predicate variable's bounds, we can quickly estimate the ranges of all neurons in the ImageStar set in Fig. 2 without solving any linear programming (LP) optimization problems (by using Eq. 3). From the estimated ranges, we see $ReLU(n_{21}) = 0$ ($n_{21} \leq 0$) and $ReLU(n_{22}) = 2 - \alpha_1 + \alpha_2$ ($n_{22} > 0$). Therefore, to overapproximate $ReLU(I)$, we need only perform the overapproximation rule on neurons n_{11} and n_{12}, which is where the user-defined relaxation can be applied. In the original approximate reachability approach [33], we use the exact ranges to construct the triangle overapproximation of the ReLU activation function, which requires solving 4 LPs to find the exact ranges for n_{11} and n_{12}, which are $[-0.5, 1.5]$ and $[-1, 1]$ respectively in this example. Now, if users want to *reduce the number of LPs* solved in constructing the overapproximate reachable set to speed up verification, which LPs should be chosen to solve to construct a sufficiently tight overapproximate reachable set? For example, if the users want to relax 50% number of LPs for Example 1, then only $4 - (50\% \times 4) = 2$ LPs are solved to construct an overapproximate reachable set. So, which two LPs should be chosen?

The answer is found by combining the exact ranges obtained by solving LPs and the estimated ranges to construct the overapproximate reachable set. This can be done using on of the following heuristic approaches. These approaches select which neurons and their corresponding lower (upper) bounds should be obtained exactly to construct an as-tight-as-possible overapproximate reachable set with a given allowable number of LPs. Some of these heuristic approaches are based on the estimated ranges information.

3.2.1 Randomly Relaxed Reachability

This approach randomly selects some LPs in the LPs pool to solve to obtain the lower (upper) bounds for some (random) neurons. For Example 1, the LPs pool is as follows.

$$LP_{pool} = \{\min(n_{11}), \max(n_{11}), \min(n_{22}), \max(n_{22}),\\ subject\ to: P = C\alpha \leq d, l \leq \alpha \leq u\}.$$

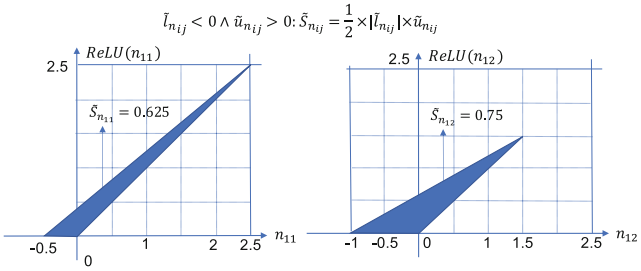


Fig. 3. Overapproximation areas at neurons n_{11} and n_{12} using estimated ranges.

If users relax 50% of the LPs, then the randomly relaxed reachability algorithm selects aimlessly two LPs in the LP pool to solve, and then combines the obtained lower (upper) ranges with the estimated ranges to construct an overapproximate reachable set using the triangle overapproximation rule, i.e., Lemma 3.

From Fig. 2, we can see that the estimated lower ranges of neurons n_{11} and n_{12} are the same as the exact ones. Therefore, if the randomly relaxed reachability algorithm selects $\min(n_{11})$ and $\min(n_{12})$ to solve, the final ranges used for constructing the reachable set exactly match the estimated ranges. This means solving $\min(n_{11})$ and $\min(n_{12})$ wastes time and does not reduce the conservativeness of the overapproximate reachable set, as no tighter ranges are obtained. In another case, if the algorithm selects $\max(n_{11})$ and $\max(n_{12})$, then we can obtain the exact ranges of two neurons by solving only two LPs (instead of four LPs), when combining the estimated lower ranges, i.e., -0.5 for n_{11} and -1 for n_{12} with the optimized upper ranges, i.e., 1.5 for n_{11} and 1 for n_{12}. In this case, the randomly relaxed algorithm can obtain the tightest overapproximate reachable set by solving only 50% of the LPs.

3.2.2 Area-Based Relaxed Reachability

The area-based relaxed reachability approach finds and optimizes the ranges of neurons with the potentially largest triangle overapproximation areas. Figure 3 illustrates the areas of the triangle overapproximation at neurons n_{11} and n_{12} using the estimated ranges. We see the overapproximation area of n_{12} ($\tilde{S}_{n_{12}} = 0.75$) is larger than that of n_{11} ($\tilde{S}_{n_{11}} = 0.625$). Therefore, if users relax 50% of the LPs, the area-based relaxed reachability algorithm will use two LPs to optimize the range of neuron n_{12}, i.e., solving $\min(n_{12})$ and $\max(n_{12})$. With this optimized range, the overapproximation area of the neuron n_{12} reduces from $\tilde{S}_{n_{12}} = 0.75$ to $S_{n_{12}} = 0.5$. If users relax 75% of the LPs, then the algorithm will use two LPs to optimize the range of the neuron n_{12} and one LP to optimize the upper bound of the neuron n_{11}, because $\tilde{u}_{11} = 2.5 > |\tilde{l}_{11}| = 0.5$.

3.2.3 Range-Based Relaxed Reachability

The range-based relaxed reachability approach finds the neurons with the potentially widest ranges to optimize their ranges. For Example 1, unlike the

area-based approach, the range-based approach will use two LPs to optimize the range of neuron n_{11}, i.e., solving $\min(n_{11})$ and $\max(n_{11})$, whose estimated range (ER) is widest ($ER_{n_{11}} = |\tilde{u}_{n_{11}} - \tilde{l}_{n_{11}}| = |2.5 - (-0.5)| = 3 > ER_{n_{12}} = 2.5$). After optimizing the range of neuron n_{11}, the overapproximation area at this neuron reduces from $\tilde{S}_{n_{11}} = 0.625$ to $S_{n_{11}} = 0.375$. The improvement in terms of overapproximation area reduction of the range-based method is equivalent to the above area-based approach in this case, i.e., $\Delta S_{n_{11}} = \Delta S_{n_{12}} = 0.25$.

3.2.4 Bound-Based Relaxed Reachability

The bound-based relaxed reachability approach finds neurons with the potentially largest (lower or upper) bounds to optimize their bounds. For Example 1, the algorithm will use two LPs to optimize the upper bounds of the neurons n_{11} and n_{12}, i.e., solving $\max(n_{11})$ and $\max(n_{12})$, because their estimated upper bounds are the ones with largest absolute values. Thus, $|\tilde{u}_{n_{11}}| = 2.5 > |\tilde{u}_{n_{12}}| = 1.5 > |\tilde{l}_{n_{12}}| = 1 > |\tilde{l}_{n_{11}}| = 0.5$. After optimizing these upper bounds, the overapproximation areas at neurons n_{11} and n_{12} reduces to 0.375 and 0.5 respectively. In this case, we can see that the bound-based relaxed approach is the best approach compared to the others since it reduces the overapproximation errors at both neurons n_{11} and n_{12}, effectively reducing the overapproximation areas by $\Delta S_{n_{11}} = \Delta S_{n_{12}} = 0.25$. It is worth noting the obtained overapproximate reachable set is the same as the one obtained by the original approximate ImageStar reachability because the estimated and optimized lower bounds are the same.

3.3 Reachability of a Pixel-Classification Layer

The last layer in an SSN is a pixel-classification layer, which assigns a specific class (label) to each pixel of an input image. Given an $h \times w \times nc$ input image, the size of the input x to the pixel-classification layer is $h \times w \times L$, where L is the number of classes (labels) of the network (we neglect the softmax layer in the analysis). To assign a specific class l, $1 \leq l \leq L$ to a pixel $x(i,j) \in x$, $1 \leq i \leq h, 1 \leq j \leq w$, the value of the pixel $x(i,j)$ at channel l, i.e., $x(i,j,l)$, needs to be the maximum one among L channels. When the input to the network is an ImageStar set instead of a single image, the input to the pixel-classification layer is a $h \times w \times L$ ImageStar set. Depending on the value of the predicate variables in the input set, a pixel $x(i,j)$ in the set may be assigned to more than one class. For example, if $l_1, \ldots, l_m$ are the cross-channel max-point candidates of the pixel $x(i,j)$ in L channels, the pixel-class reachable set of the layer at the considered pixel is $pc(i,j) = \{l_1, \ldots, l_m\}$. By determining all cross-channel max-point candidates of all pixels in the input set, we can obtain the pixel-class reachable set of the layer, which is also the reachable set of the SSN, $\mathcal{R}_f = [pc(i,j)]_{h \times w}$, i.e., the collection of pixel classes at every index (i,j).

Similar to the max-pooling layer [33], determining all cross-channel max-point candidates of all pixels in the input set can be done via solving linear programming (LP) optimization problems, which is time-consuming due to the number of LPs required (or equivalently the size of the LP). To reduce computation time, we estimate the lower and upper bounds of the ImageStar input to

the layer using only the ranges of the predicate variables. These bounds are then used to predict all possible cross-channel max-point candidates of all pixels.

4 Verification Algorithm

Our reachability-based verification algorithm for SSNs is presented in Algorithm 4.1. The algorithm takes an SSN f, an input image x, an UBAA $\mathcal{A}$, and a reachability method (exact or approximate) as inputs, then returns the pixel-class reachable set $\mathcal{R}_f$, the robustness value RV, sensitivity RS, and robust IoU R_{IoU} of the SSN. The algorithm works as follows. First, it constructs the input set corresponding to the attack using Proposition 1 (line 2). Then, it computes the pixel-class reachable set of the SSN using reachability analysis layer-by-layer (line 3). Using the pixel-class reachable set, it verifies the robustness of each pixel in the reachable set by comparing its classes with the non-attacked (ground truth) output segmentation image, i.e., $y = f(x)$. If $\mathcal{R}_f(i,j) = y(i,j)$, the pixel $x(i,j)$ is robust under the attack (line 10). If $\mathcal{R}_f(i,j) \neq y(i,j) \wedge y(i,j) \notin \mathcal{R}_f(i,j)$, the pixel $x(i,j)$ is non-robust under the attack (line 12). Otherwise, the robustness of the pixel $x(i,j)$ is *unknown* (may be robust or non-robust), due to overapproximation. Beyond verifying the robustness of each pixel in the reachable set, it also counts the numbers of 1) robust pixels N_{robust} (line 10), 2) non-robust pixels $N_{nonrobust}$ (line 12), and 3) pixels with unknown robustness $N_{unknown}$ (line 13). Finally, it computes the robustness value, sensitivity and robust IoU of the SSN (lines 12, 13 and 14). The robustness of a SSN under an UBAA should be evaluated on a set of test images (Problem 2).

Algorithm 4.1. Robustness verification of a semantic segmentation network.

Input: $f, x, \mathcal{A}, RF, method$ ▷ SSN, input image, attack, relaxation factor, relaxation method
Output: $\mathcal{R}_f, RV, RS$ ▷ pixel-class reachable set, robustness value, robustness sensitivity

1: **procedure** $[\mathcal{R}_f, RV, RS]$ = VERIFY$(f, x, \mathcal{A}, RF, method)$
2: $I = constructInputSet(x, \mathcal{A})$ ▷ construct an ImageStar input set
3: $\mathcal{R}_f = Reach(f, I, method)$ ▷ compute the pixel-class reachable set
4: $y = f(x)$ ▷ compute non-attacked output segmentation image
5: $h = x.Height,\ w = x.Width$
6: $N_{robust} = 0,\ N_{nonrobust} = 0,\ N_{unknown} = 0,\ N_{attackedpixels} = 0$
7: **for** $i = 1 : h$ **do**
8: **for** $j = 1 : w$ **do**
9: **if** $\mathcal{A}.x^{noise}(i,j) \neq 0$ **then** $N_{attackedpixels} = N_{attackedpixels} + 1$
10: **if** $\mathcal{R}_f(i,j) = y(i,j)$ **then** $N_{robust} = N_{robust} + 1$ ▷ pixel $x(i,j)$ is robust
11: **else**
12: **if** $y(i,j) \notin \mathcal{R}_f(i,j)$ **then** $N_{nonrobust} = N_{nonrobust} + 1$ ▷ pixel $x(i,j)$ is non-robust
13: **else** $N_{unknown} = N_{unknown} + 1$ ▷ pixel $x(i,j)$ robustness is unknown
14: $RV = (N_{robust}/(h \cdot w)) \cdot 100\%)$ ▷ robustness value
15: $RS = (N_{nonrobust} + N_{unknown})/N_{attackedpixels}$ ▷ robustness sensitivity
16: $R_{IoU} = getAverageIoU(y, \mathcal{R}_f)$ ▷ robust IoU

5 Evaluation

Experimental Setup. The approach is implemented in the NNV software tool for verification of deep neural networks[3]. We evaluate our approach by verifying the robustness of a set of SSNs trained on the MNIST [21] and M2NIST datasets shown in Table 1, where class "ten" corresponds to the background, and the other classes to the corresponding digits. The experiments were performed on a computer with an Intel Core i7-6700 CPU at 3.4 GHz with 8 cores and 64 GB Memory running Windows 10. The over-approximating reachability method and 6 cores are used for computing the pixel-class reachable sets.

We randomly selected 100 MNIST images (of size 28×28) and 100 M2NIST images (of size 64×84) to evaluate the robustness of the trained SSNs. We attack each image x in these two test sets using an UBAA brightening attack. Particularly, we darken a pixel $x(i,j)$ in the image if its value is larger than a threshold d, i.e. if $x(i,j) > d \rightarrow x^{adv}(i,j) = a \ll d$. Mathematically, the adversarial darkening attack on an image x can be described as:

$$x^{adv} = x + \epsilon \cdot x^{noise},\ 1 - \Delta_\epsilon \leq \epsilon \leq 1,$$
$$x^{noise}(i,j) = -x(i,j),\ \text{if } x(i,j) > d,\ \text{otherwise } x^{noise}(i,j) = 0.$$

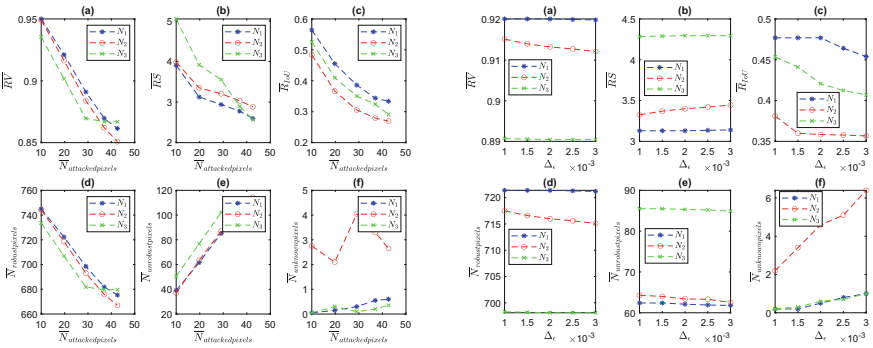
For $\epsilon = 1$, we completely darken all the pixels whose values are larger than d (=150 in our experiments), i.e., $x^{adv}(i,j) = 0$. The size of the input set caused by the attack is defined by Δ_ϵ. Generally, we have a large input set when Δ_ϵ is large. To evaluate the average robustness values ($\overline{RV}$) and sensitivities ($\overline{RS}$) of the SSNs (on the test sets) in the connection with the number of attacked pixels, we further restrict the maximum allowable number of attacked pixels by N_{max}.

We focus our evaluation and discussion on three aspects: 1) the robustness and sensitivity of different SSN architectures under adversarial attacks, 2) the effect of SSN architectures and input size on verification performance, and 3) the improvement of the new relaxed reachability method in terms of verification results and performance. For the first two aspects, we use the relaxed reachability method with relaxation factor $RF = 0\%$, i.e., no relaxation, to construct the reachable sets of the SSNs.

[3] The examples and tool are available: https://github.com/verivital/nnv/tree/cav2021/code/nnv/examples/Submission/CAV2021. An archival version is available on Zenodo: https://doi.org/10.5281/zenodo.4726346.

Table 1. Semantic Segmentation Network Benchmarks. Notation: 'I': input, 'C': convolution, 'TC': transposed convolution, 'DC': dilated convolution, 'R': ReLU, 'B': batch normalization, 'AP': average-pooling, 'MP': max-pooling, 'S': softmax, 'L': label (pixel classification).

ID	Name	Accuracy(IoU)	Down-sampling	Up-sampling	Input size	Layers
N_1	*mnist_ap_tc*	0.87	C+AP	TC	28×28	21 (1I, 7C, 3R, 4B, **2AP**, **2TC**, 1S, 1L)
N_2	*mnist_mp_tc*	0.85	C+MP	TC	28×28	21 (1I, 7C, 3R, 4B, **2MP**, **2TC**, 1S, 1L)
N_3	*mnist_dc*	0.83	C	DC	28×28	21 (1I, 3C, 3R, 3B, **9DC**, 1S, 1L)
N_4	*m2nist_ap_dc*	0.62	C+AP	DC	64×84	16 (1I, 4C, 3R, **3AP**, **3DC**, 1S, 1L)
N_5	*m2nist_ap_tc*	0.75	C+AP	TC	64×84	22 (1I, 7C, 8R, **2AP**, **2TC**, 1S, 1L)
N_6	*m2nist_dc*	0.72	C	DC	64×84	24 (1I, 1C, 5R, 5B, **10DC**, 1S, 1L)

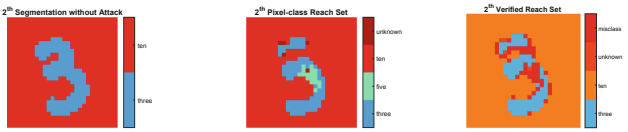


(a) $\Delta_\epsilon = 0.001$. (b) $N_{max} = 20$.

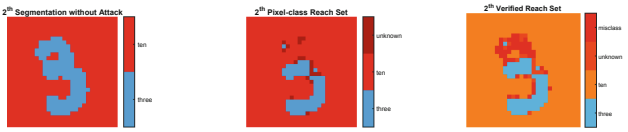
Fig. 4. The average robustness value, sensitivity, and IoU of MNIST SSNs.

5.1 Robustness and Sensitivity of Different Network Architectures

Max-Pooling vs. Average-Pooling. Max-pooling is the preferred choice over average-pooling for training SSNs because of its nonlinear characteristics. We investigate whether max-pooling is actually better than average-pooling in terms of accuracy and robustness of deep SSN. Figure 4 illustrates the average robustness and sensitivities of MNIST SSNs under different numbers of attacked pixels (Fig. 4a, 20 images are used) and input sizes (Fig. 4b, 10 images are used). We focus on the first two SSNs, i.e. N_1 and N_2. These SSNs have the same architectures (with 21 layers). The only difference is N_1 uses average-pooling for down-sampling while N_2 uses max-pooling for the same task (both SSNs use two transposed convolutional layers for up-sampling). With training, we experienced that N_1 is more accurate than N_2, (0.87 IoU vs. 0.85 IoU, see Table 1). Interestingly, N_1 is also more robust than N_2 since it has a larger average robustness value (Figs. 4a-a, 4b-a), a higher average robust IoU (Figs. 4a-c, 4b-c), and more robust pixels (Figs. 4a-d, 4b-d). One can also see that the average-pooling-based SSN is less sensitive to the attack than the max-pooling-based SSN (Figs. 4a-(b, e, f), 4b-(b, e, f)). Notably, when more pixels are attacked or larger input sizes are used, the max-pooling-based SSN (i.e., N_2) produces more pixels with unknown



(a) $R_f(N_1), N_{unknown} = 6$ ($N_{max} = 50, \Delta_\epsilon = 0.003$).

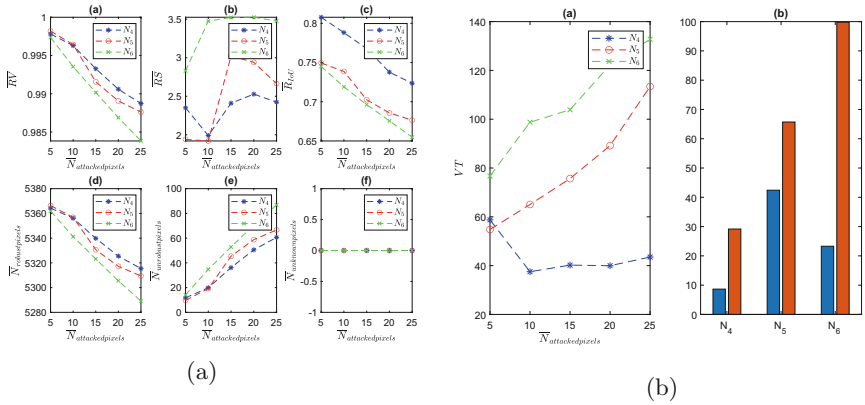


(b) $R_f(N_2), N_{unknown} = 19$ ($N_{max} = 50, \Delta_\epsilon = 0.003$).

Fig. 5. Example pixel-class reachable sets of MNIST SSNs. The max-pooling-based SSN N_2 produces more unknown pixels than the average-pooling-based SSN N_1 (19 vs. 6).

robustness (Figs. 4a-f, 4b-f, and 5). Lastly, when the input size increases, the robustness of the max-pooling-based SSN drops more quickly than the average-pooling-based SSNs (Fig. 4b (a,d)) and its sensitivity increases faster (Fig. 4b -b). We believe the main reason causing the max-pooling-based SSN to be more sensitive to the attack is its high nonlinearity using max-pooling layers. It is quite interesting that even the max-pooling-based SSN N_2 has a higher accuracy (0.85) than the non-max-pooling SSN N_3 (0.83), the average robust IoU of the SSN N_2 is smaller than the one of N_3 (Figs. 4a-c, 4b-c).

Accuracy vs. Robustness; Deeper Networks and ReLU Layer Robustness. Accuracy (and for segmentation, IoU) is one of the most important factors for evaluating deep neural networks. We investigate whether more accurate and deeper SSNs are more robust compared to other architectures. To determine this, we analyze the robustness of two SSNs with different architectures and accuracy trained on the M2NIST data set. The first SSN N_4 is based on dilated convolution with 16 layers and 0.62 (IoU) accuracy (Table 1). The second SSN N_5 is based on transposed convolution with 22 layers and 0.75 (IoU) accuracy. Here, the second SSN is deeper and more accurate than the first SSN. We run the robustness analysis on these two SSNs on a set of 20 M2NIST images. The results are depicted in Fig. 6. In terms of robustness, the more accurate and deeper SSN N_5 is worse than the less accurate one N_4 as it has a smaller average robustness value and IoU (Figs. 6-(a,c), 7). Additionally, N_5 is also more sensitive to the attack than N_4 (Fig. 6-(b,e)) when we increase the number of attacked pixels. The main reason for this result is, the more accurate SSN contains many ReLU layers (8 ReLU layers) compared with the less accurate one (3 ReLU layers). Similar to the max-pooling layer, using many ReLU layers increases the nonlinearity of the SSN to capture complex features of images. Unfortunately, it also makes the SSN more sensitive to the attack.



(a) (b)

Fig. 6. The average robustness value, sensitivities, IoU, verification time ($\Delta_\epsilon = 10^{-5}$) and reachability times (blue for ReLU layers and orange for others, $\Delta_\epsilon = 6 \times 10^{-5}$) of M2NIST SSNs. (Color figure online)

(a) $R_f(N_4), N_{nonrobust} = 43$ ($N_{max} = 25, \Delta_\epsilon = 0.00001$).

(b) $R_f(N_5), N_{nonrobust} = 51$ ($N_{max} = 25, \Delta_\epsilon = 0.00001$).

Fig. 7. Example pixel-class reachable sets of M2NIST SSNs. The more accurate and deeper SSN N_5 produces more non-robust pixels than the less accurate SSN N_4 (51 vs. 43).

Dilated Convolution vs. Transposed Convolution. Dilated convolution and transposed convolution are typical choices for semantic segmentation tasks. We compare these techniques in terms of accuracy and robustness. On MNIST SSNs, although the transposed-convolution SSNs N_1 and the dilated-convolution SSN N_3 have the same number of layers (21 layers with 3 ReLU), N_3 is less accurate than N_1 (0.83 vs. 0.87 IoU, see Table 1). In terms of robustness, N_3 is also less robust and more sensitive to the attack than N_1, as it has smaller average RV and IoU, and larger sensitivities (Fig. 4). On M2NIST SSNs, by considering 21-layer (8 ReLU) transposed-convolution SSN N_5 and 24-layer (4 ReLU) dilated-convolution SSN N_6, one can see that even with more layers, N_6 is less accurate than N_5 (0.72 vs. 0.75 IoU, see Table 1). Also, N_6 is less robust and more sensitive to the attack than N_5, since it has smaller average RV and IoU, and larger sensitivities (Fig. 6).

5.2 Verification Performance

Dilated Convolution vs. Transposed Convolution. In general, more attacked pixels and larger input size leads to greater verification time, as depicted in Figs. 8a, 8b and 6b-(a). Interestingly, these show that the dilated-convolution-based SSNs require greater verification time than the ones using transposed convolution. For example, the verification time of N_3 is larger than N_2 when they have the same number of layers.

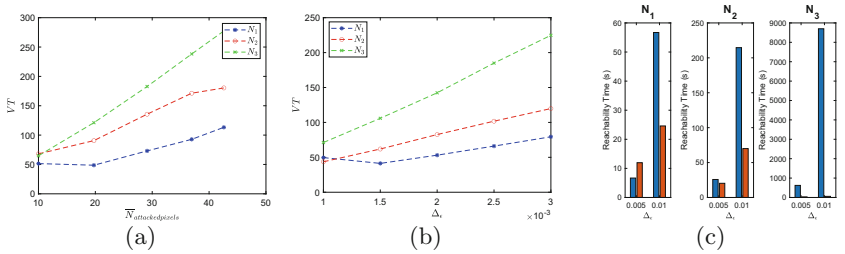


Fig. 8. Verification time is proportional to the number of attacked pixels and input size. The max-pooling-based N_2 and dilated convolution-based N_3 SSNs require more verification time than the average-pooling and transposed convolution-based SSN N_1. The reachability times of ReLU layers (blue) dominates the total reachability time (other layers reachability times are in orange). (Color figure online)

Max-Pooling and ReLU Layers. Using max-pooling layer for down sampling not only decreases the robustness of an SSN but also causes a dramatic increase in time and memory consumption in verification. Figure 8 shows that the verification time (in seconds) of the max-pooling-based SSN N_2 grows significantly compared with the average-pooling-based SSN N_1 when increasing the number of attacked pixels $N_{attackedpixels}$ or the input size Δ_ϵ. When dealing with more number of attacked pixels or larger input size, the max-pooling layer introduces more predicate variables to overapproximate the reachable set, which causes the increase both in computation time and memory usage [33]. Similar to the max-pooling layer, the ReLU layer is also the main source of robustness degradation. Additionally, it may also dominate the reachability time of a SSN, as shown in Fig. 8c. This leads to an increase in the verification time for SSNs with many ReLU layers.

5.3 Reducing Verification Time with Relaxation

When ReLU layer analysis dominates the total verification time significantly, as in the case of MNIST SSNs shown in Fig. 8c and not in the case of M2NIST SSNs depicted in Fig. 6b-(b), we can use the relaxed ImageStar reachability methods to speed up the verification process. Table 2 presents the decrease in

the verification times in percentage when applying different relaxation heuristics for ReLU layers. We note that due to the small input size and a small number of attacked pixels, we do not see any changes in the robustness value, sensitivity, and IoU compared with the non-relaxation method, i.e., the original approximate ImageStar method. However, there is a significant improvement in verification time when we apply the relaxed ImageStar reachability for non-max-pooling SSNs N_1 and N_3. More relaxation leads to a higher reduction in the verification time: up to 99% of the verification time can be reduced with 100% relaxation in the reachability of ReLU layers.

Interestingly, using relaxation for the max-pooling-based SSN N_2 decreases the verification performance, i.e., leading to higher verification time. The main reason is that the relaxed reachable sets after ReLU layers become increasingly conservative. At the max-pooling layer, a more conservative reachable set leads to more local max-point candidates that need to be determined via solving more LPs, which causes an increase in the verification time. Additionally, if a local region has more than one max-point candidate, a new predicate variable and its corresponding generator image are introduced [33]. The increase in the number of predicate variables and generator images causes the explosion in the memory

Table 2. The relaxed ImageStar reachability methods can reduce significantly the verification time (in seconds) of MNIST SSN networks except for the one containing max-pooling layers, i.e., N_2. The maximum allowable number of attacked pixels is $N_{max} = 50$ for N_1 and N_2 and $N_{max} = 20$ for N_3.

ID	RF	$\Delta_\epsilon = 0.005$				$\Delta_\epsilon = 0.01$				$\Delta_\epsilon = 0.02$			
		Rand	Area	Range	Bound	Rand	Area	Range	Bound	Rand	Area	Range	Bound
N_1	0.00	20.56	20.23	19.43	19.05	82.57	81.04	76.48	83.51	860.86	861.69	734.99	862.03
	0.25	19.5(↓ 5%)	20.7(↓ −2%)	18.8(↓ 3%)	20.7(↓ −9%)	72.2(↓ 13%)	75.8(↓ 7%)	69.5(↓ 9%)	84.4(↓ −1%)	734.1(↓ 15%)	770.2(↓ 11%)	665.0(↓ 10%)	978.1(↓ −13%)
	0.50	17.7(↓ 14%)	18.6(↓ 8%)	18.3(↓ 6%)	19.0(↓ 0%)	58.3(↓ 29%)	67.0(↓ 17%)	62.1(↓ 19%)	69.5(↓ 17%)	587.8(↓ 32%)	613.6(↓ 29%)	530.7(↓ 28%)	779.5(↓ 10%)
	0.75	17.0(↓ 17%)	17.7(↓ 13%)	16.7(↓ 14%)	16.9(↓ 11%)	47.4(↓ 43%)	49.9(↓ 38%)	51.0(↓ 33%)	53.0(↓ 37%)	347.6(↓ 60%)	389.2(↓ 55%)	361.1(↓ 51%)	439.0(↓ 49%)
	1.00	15.2(↓ 26%)	16.4(↓ 19%)	16.2(↓ 17%)	15.2(↓ 20%)	34.4(↓ 58%)	34.4(↓ 58%)	36.0(↓ 53%)	36.0(↓ 57%)	90.5(↓ 89%)	90.1(↓ 90%)	94.4(↓ 87%)	92.9(↓ 89%)
N_2	0.00	45.13	44.38	43.72	45.19	281.30	285.60	254.02	281.31	*MemErr*	*MemErr*	*MemErr*	*MemErr*
	0.25	53.1(↓ −18%)	53.5(↓ −21%)	51.8(↓ −19%)	69.9(↓ −55%)	308.0(↓ −9%)	294.7(↓ −3%)	255.6(↓ −1%)	378.1(↓ −34%)	*MemErr*	*MemErr*	*MemErr*	*MemErr*
	0.50	64.3(↓ −42%)	66.4(↓ −50%)	62.9(↓ −44%)	86.5(↓ −91%)	302.2(↓ −7%)	312.7(↓ −10%)	295.2(↓ −16%)	481.0(↓ −71%)	*MemErr*	*MemErr*	*MemErr*	*MemErr*
	0.75	72.9(↓ −62%)	75.8(↓ −71%)	72.5(↓ −66%)	93.0(↓ −106%)	306.0(↓ −9%)	309.0(↓ −8%)	344.4(↓ −36%)	448.5(↓ −59%)	*MemErr*	*MemErr*	*MemErr*	*MemErr*
	1.00	79.6(↓ −76%)	79.5(↓ −79%)	79.8(↓ −83%)	79.9(↓ −77%)	364.4(↓ −30%)	325.4(↓ −14%)	322.1(↓ −27%)	318.5(↓ −13%)	*MemErr*	*MemErr*	*MemErr*	*MemErr*
N_3	0.00	119.63	118.74	112.48	120.66	1119.16	1116.85	996.56	1116.66	17699.81	17651.30	17260.00	17780.00
	0.25	95.9(↓ 20%)	100.4(↓ 15%)	95.3(↓ 15%)	107.9(↓ 11%)	920.7(↓ 18%)	1020.7(↓ 9%)	874.9(↓ 12%)	1157.0(↓ −4%)	15474.4(↓ 13%)	17222.3(↓ 2%)	14700.0(↓ 15%)	17201.0(↓ 3%)
	0.50	72.7(↓ 39%)	77.8(↓ 35%)	72.2(↓ 36%)	78.5(↓ 35%)	648.4(↓ 42%)	759.4(↓ 32%)	644.2(↓ 35%)	797.8(↓ 29%)	11976.3(↓ 32%)	14566.7(↓ 17%)	11902.0(↓ 31%)	14729.0(↓ 17%)
	0.75	45.5(↓ 62%)	50.2(↓ 58%)	48.5(↓ 57%)	49.6(↓ 59%)	352.6(↓ 68%)	424.9(↓ 62%)	378.1(↓ 62%)	416.8(↓ 63%)	6720.0(↓ 62%)	8556.8(↓ 52%)	7217.0(↓ 58%)	7942.0(↓ 55%)
	1.00	22.0(↓ 82%)	23.0(↓ 81%)	22.9(↓ 80%)	22.3(↓ 81%)	47.6(↓ 96%)	45.7(↓ 96%)	45.7(↓ 95%)	45.2(↓ 96%)	116.1(↓ 99%)	115.7(↓ 99%)	115.4(↓ 99%)	115.2(↓ 99%)

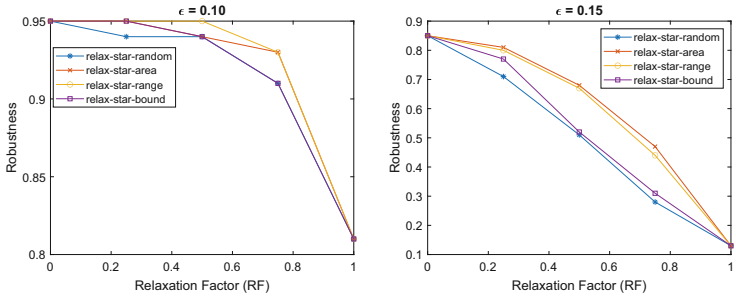


Fig. 9. The conservativeness of different relaxation heuristics. The area-based and range-based relaxation strategies outperform others in terms of conservativeness.

usage for the analysis. In the worst case, it can lead to a memory error as shown in Table 2. Therefore, it is important to have relaxation strategies for max-pooling layers, which will be investigated in our future work.

5.4 Conservativeness of Different Relaxation Heuristics

We have four relaxation heuristics that can be used in the reachability analysis of ReLU layers. The verification time improvement of these methods is quite similar, as shown in Table 2. It is interesting to see how good they are in terms of conservativeness. Unfortunately, we cannot see it clearly via verification of SSNs. Although increasing the number of attacked pixels and input size can eventually show the difference in conservativeness of these methods, it requires a more powerful computer with massive memory for verification. Therefore, to determine the best relaxation heuristic in terms of conservativeness, we evaluate image classification robustness that has been studied extensively recently, and illustrates the benefits of the relaxation method beyond SSN verification. We apply our four relaxation heuristics to verify robustness of an MNIST classification network [29] that is trained by the DiffAI robust training framework under the L_∞-norm attack, where all pixels of an input image are attacked independently by a bounded disturbance defined by ϵ[4]. The robustness of the network is quantified in percentage stating how many images of 100 randomly selected images are provably robust under the attack, i.e., classified correctly.

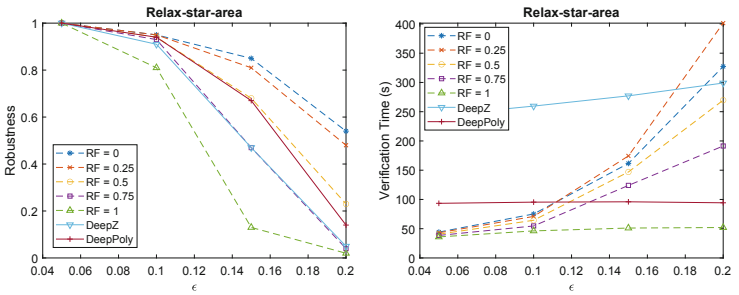


Fig. 10. When the relaxation factor (RF) ≤ 0.5, the area-based relaxed reachability is less conservative than DeepZono [28] and DeepPoly [29]. It is also faster than these approaches when the disturbance is small, i.e., $\epsilon \leq 0.11$.

Figure 9 illustrates the conservativeness of different relaxation methods. One can see that the area-based and range-based relaxation strategies consistently outperform others in terms of conservativeness since their provable numbers of robust images (in 100 images) under the different sizes of the L_∞ norm attacks are higher than others in all cases. Figure 10 illustrates the conservativeness and verification time of our area-based relaxed reachability (with different relaxation

[4] These benchmarks were used in VNN-COMP'20.

factors (RF)) in comparison with DeepZono [28] and DeepPoly [29]. In terms of conservativeness, the area-based relaxed reachability is better than DeepZono and DeepPoly when we choose a relaxation factor $RF \leq 0.5$. When the disturbance is large, DeepZono and DeepPoly may become very conservative. For example, when the disturbance bound $\epsilon = 0.2$, the only 5 and 14 (over 100) images are proved robust by DeepZono and DeepPoly, respectively. Meanwhile, without relaxation, i.e., relaxation factor $RF = 0$, the area-based relaxed reachability can prove 54 images are robust under the attack. It can prove robustness of 48 and 23 images when the relaxation factors are 0.25 and 0.5, respectively. In terms of verification time, when the disturbance is small, i.e., $\epsilon \leq 0.11$, the area-based relaxed reachability is faster than DeepZono and DeepPoly. It is slower than DeepPoly for larger disturbance (except for the case when the relaxation factor is 1). This increase in the verification time is apparent since DeepZono and DeepPoly do not solve any LPs for constructing the overapproximate reachable set of the network while our approach does. Due to using only estimated ranges of the neurons in constructing the reachable set, DeepZono and DeepPoly are overly conservative for a large disturbance, proving only a few images are robust. This reflects the fact that more computation time for optimization is needed to prove more images robust.

6 Related Work

To enable neural networks use in safety-critical scenarios, many methods have recently been proposed to improve their robustness and temper their susceptibility to adversarial attacks. The following section surveys the landscape of these approaches in order to better contextualize our work.

SSN Robustness. SSNs are used in visual understanding systems in numerous contexts, recent works aim to improve the robustness of these models [13,20,23,25], albeit none that provide worst-case guarantees, as our approach does. For instance, recent work develops rigorous testing-based approaches to evaluate the robustness of SSNs, considering a wide range of architectures, and offering an insightful discussion about the comparative robustness of these modalities against various adversarial attacks [2]. Kamann et al. conducted an extensive evaluation of a state-of-the-art SSN using over 400,000 images and issued a series of recommendations aimed at improving robustness to common perturbations. Zhou et al. presented an automated method for evaluating robustness of SSNs within visual systems for autonomous vehicles, which leverages an additional sensor to generate ground truth labels so that an examination of the classification accuracy of an SSN can be evaluated at runtime[47]. Robust training techniques that incorporate image corruptions and architecture modalities have also been developed for SSNs [20]. Even though such works provide better understanding, potential defenses against adversarial perturbations, run-time evaluation, and comparative robustness measures, they cannot provide formal verification guarantees for SSN robustness as our work does.

Neural Network Verification and Falsification. The bulk of neural network verification approaches have been aimed at verifying input-output properties of DNNs. These methods include SMT [18,19], polyhedral [35,44], mixed integer linear programming (MILP) [9], interval arithmetic [38], zonotope [28], linearization[39], and abstract-domain [29] approaches. There have also been a number of works aimed at testing the robustness of networks with respect to bounded input perturbations such as feature-guided search, global optimization, and game theory [16,42]. One such example is the work of Dreossi et al. where the authors proposed a general definition of robustness for DNNs [8]. Their work categorizes the existing literature into approaches that consider local robustness properties [6], and those that focus on verifying the global robustness of the networks [14]. Most of the existing research in this area focuses on robustness of classification neural networks, specifically image classification. While many approaches aim at verification, methods also exist for falsification of system specifications, in which robustness properties are included [12]. However, to the best of our knowledge, no existing approaches consider verification for SSNs, as we do in this paper.

Sequence Model Verification and Robustness Analysis. Aside from classification tasks, there are several verification approaches for sequence models. Unlike SSN and classification networks, the output of sequence models such as recurrent neural networks (RNNs) depends on spatially or temporally ordered data [4,41]. While some of these efforts are similar in spirit to our work in expanding the classes of problems and models for verification, the verification tasks and approaches differ.

Scalability and Specifications. Finally, verification of DNNs is challenging, and presently the most complex networks remain inaccessible to the majority of methods. However, several recent approaches have focused on improving the efficiency of existing methods via parallelization and other techniques [3,35,40]. As verification work is only meaningful when paired with high-quality specifications, there has been significant work on the importance of semantics when defining system specifications against adversarial attacks [27], and our paper contributes to this direction through our formulation of robustness specifications and metrics for segmentation tasks.

7 Conclusion

We present the first formal approach to verify robustness of SSNs using relaxed reachability analysis. Our evaluation has analyzed the robustness and sensitivity under adversarial attacks on a set of SSNs with typical architectures. From our experiments, we show that while max-pooling and ReLU layers are useful in training highly accurate SSNs, they are also the main sources of robustness and verification performance degradation. SSNs using average-pooling for down-sampling and transposed convolution for up-sampling seem to be an optimal choice for achieving high accuracy, robustness, and verification performance.

Additionally, our relaxed reachability approach can help to reduce significantly the total verification time for networks where the reachability time of ReLU layers dominates the network's reachability time, and are applicable to other networks, such as CNNs used for classification. In the future, we will investigate new relaxation heuristics for the max-pooling layer and extend this work to cope with the encoder-decoder SSN architecture where max-unpooling layers are used for up-sampling operations, instead of dilated/transposed convolution as we considered in this paper.

Acknowledgments. The material presented in this paper is based upon work supported the Defense Advanced Research Projects Agency (DARPA) through contract number FA8750-18-C-0089, the Air Force Office of Scientific Research (AFOSR) award FA9550-19-1-0288, and the National Science Foundation (NSF) through grant numbers 1910017 and 2028001. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of DARPA, AFOSR or NSF.

References

1. Anderson, G., Pailoor, S., Dillig, I., Chaudhuri, S.: Optimization and abstraction: A synergistic approach for analyzing neural network robustness. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, pp. 731–744. Association for Computing Machinery, New York (2019)
2. Arnab, A., Miksik, O., Torr, P.H.: On the robustness of semantic segmentation models to adversarial attacks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 888–897 (2018)
3. Bak, S., Tran, H.-D., Hobbs, K., Johnson, T.T.: Improved geometric path enumeration for verifying ReLU neural networks. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12224, pp. 66–96. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53288-8_4
4. Baluta, T., Shen, S., Shinde, S., Meel, K.S., Saxena, P.: Quantitative verification of neural networks and its security applications. CoRR arXiv:1906.10395 (2019)
5. Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., Misener, R.: Efficient verification of relu-based neural networks via dependency analysis. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, pp. 3291–3299 (2020)
6. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57 (2017)
7. Dathathri, S., et al.: Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming (2020)
8. Dreossi, T., et al.: VerifAI: a toolkit for the formal design and analysis of artificial intelligence-based systems. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 432–442. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_25
9. Dutta, S., Jha, S., Sanakaranarayanan, S., Tiwari, A.: Output range analysis for deep neural networks. arXiv preprint arXiv:1709.09130 (2017)

10. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: D'Souza, D., Narayan Kumar, K. (eds.) ATVA 2017. LNCS, vol. 10482, pp. 269–286. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_19
11. Fazlyab, M., Morari, M., Pappas, G.J.: Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. IEEE Trans. Autom. Control 1 (2020)
12. Fremont, D.J., Chiu, J., Margineantu, D.D., Osipychev, D., Seshia, S.A.: Formal analysis and redesign of a neural network-based aircraft taxiing system with VerifAI. In: 32nd International Conference on Computer Aided Verification (CAV) (July 2020)
13. Full, P.M., Isensee, F., Jäger, P.F., Maier-Hein, K.: Studying robustness of semantic segmentation under domain shift in cardiac MRI (2020)
14. Gopinath, D., Katz, G., Păsăreanu, C.S., Barrett, C.: DeepSafe: a data-driven approach for assessing robustness of neural networks. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 3–19. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01090-4_1
15. Huang, C., Fan, J., Li, W., Chen, X., Zhu, Q.: Reachnn: reachability analysis of neural-network controlled systems. ACM Trans. Embed. Comput. Syst. (TECS) **18**(5s), 1–22 (2019)
16. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 3–29. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_1
17. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: verifying safety properties of hybrid systems with neural network controllers. In: Hybrid Systems: Computation and Control (HSCC) (2019)
18. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: an efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 97–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_5
19. Katz, G., et al.: The marabou framework for verification and analysis of deep neural networks. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 443–452. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_26
20. Klingner, M., Bar, A., Fingscheidt, T.: Improved noise and attack robustness for semantic segmentation by using multi-task training with self-supervised depth estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (June 2020)
21. LeCun, Y.: The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/ (1998)
22. Liu, C., Arnon, T., Lazarus, C., Barrett, C., Kochenderfer, M.J.: Algorithms for verifying deep neural networks. arXiv preprint arXiv:1903.06758 (2019)
23. Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., Terzopoulos, D.: Image segmentation using deep learning: A survey (2020)
24. Mohapatra, J., Weng, T.W., Chen, P.Y., Liu, S., Daniel, L.: Towards verifying robustness of neural networks against a family of semantic perturbations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
25. Oliveira, G., Bollen, C., Burgard, W., Brox, T.: Efficient and robust deep networks for semantic segmentation. Int. J. Rob. Res. **37**, 027836491771054 (2017)
26. Ruan, W., Wu, M., Sun, Y., Huang, X., Kroening, D., Kwiatkowska, M.: Global robustness evaluation of deep neural networks with provable guarantees for the l_0 norm. arXiv preprint arXiv:1804.05805 (2018)

27. Seshia, S.A., et al.: Formal specification for deep neural networks. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 20–34. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01090-4_2
28. Singh, G., Gehr, T., Mirman, M., Püschel, M., Vechev, M.: Fast and effective robustness certification. In: Advances in Neural Information Processing Systems, pp. 10825–10836 (2018)
29. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proc. ACM Program. Lang. vol. 3(POPL), p. 41 (2019)
30. Szeliski, R.: Computer Vision: Algorithms and Applications. 2nd edn. Springer, New York (2021) https://doi.org/10.1007/978-1-84882-935-0
31. Thoma, M.: A survey of semantic segmentation. arXiv preprint arXiv:1602.06541 (2016)
32. Tjeng, V., Xiao, K.Y., Tedrake, R.: Evaluating robustness of neural networks with mixed integer programming. In: International Conference on Learning Representations (2019)
33. Tran, H.-D., Bak, S., Xiang, W., Johnson, T.T.: Verification of deep convolutional neural networks using ImageStars. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12224, pp. 18–42. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53288-8_2
34. Tran, H.D., Cei, F., Lopez, D.M., Johnson, T.T., Koutsoukos, X.: Safety verification of cyber-physical systems with reinforcement learning control. In: ACM SIGBED International Conference on Embedded Software (EMSOFT 2019), ACM (October 2019)
35. Tran, H.D., et al.: Parallelizable reachability analysis algorithms for feed-forward neural networks. In: 7th International Conference on Formal Methods in Software Engineering (FormaliSE2019), Montreal, Canada (2019)
36. Tran, H.-D., et al.: Star-based reachability analysis of deep neural networks. In: ter Beek, M.H., McIver, A., Oliveira, J.N. (eds.) FM 2019. LNCS, vol. 11800, pp. 670–686. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30942-8_39
37. Tran, H.D., Xiang, W., Johnson, T.T.: Verification approaches for learning-enabled autonomous cyber-physical systems. IEEE Design & Test (2020)
38. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Efficient formal safety analysis of neural networks. In: Advances in Neural Information Processing Systems, pp. 6369–6379 (2018)
39. Weng, T.W., et al.: Towards fast computation of certified robustness for relu networks. arXiv preprint arXiv:1804.09699 (2018)
40. Wu, H., et al.: Parallelization techniques for verifying neural networks. In: 2020 Formal Methods in Computer Aided Design (FMCAD), pp. 128–137 (2020)
41. Wu, J., Li, X., Ao, X., Meng, Y., Wu, F., Li, J.: Improving robustness and generality of nlp models using disentangled representations (2020)
42. Wu, M., Wicker, M., Ruan, W., Huang, X., Kwiatkowska, M.: A game-based approximate verification of deep neural networks with provable guarantees. Theor. Comput. Sci. **807**, 298–329 (2020)
43. Xiang, W., et al.: Verification for machine learning, autonomy, and neural networks survey. arXiv preprint arXiv:1810.01989 (2018)
44. Xiang, W., Tran, H.D., Johnson, T.T.: Reachable set computation and safety verification for neural networks with relu activations. arXiv preprint arXiv:1712.08163 (2017)
45. Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial examples: attacks and defenses for deep learning. IEEE Trans. Neural Netw. Learn. Syst. **30**(9), 2805–2824 (2019)

46. Zhang, H., Weng, T.W., Chen, P.Y., Hsieh, C.J., Daniel, L.: Efficient neural network robustness certification with general activation functions. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31, pp. 4939–4948. Curran Associates, Inc. (2018)
47. Zhou, W., Berrio, J., Worrall, S., Nebot, E.M.: Automated evaluation of semantic segmentation robustness for autonomous driving. IEEE Trans. Intell. Transp. Syst. **21**, 1951–1963 (2020)