Learning Minimal Neural Specifications

Chuqin Geng CHUQIN.GENG@MAIL.MCGILL.CA
McGill University

Zhaoyue Wang ZHAOYUE.WANG@MAIL.MCGILL.CA
McGill University

Haolin Ye HAOLIN.YE@MAIL.MCGILL.CA
McGill University

Xujie Si SIX@CS.TORONTO.EDU
University of Toronto

Editors: G. Pappas, P. Ravikumar, S. A. Seshia

Abstract

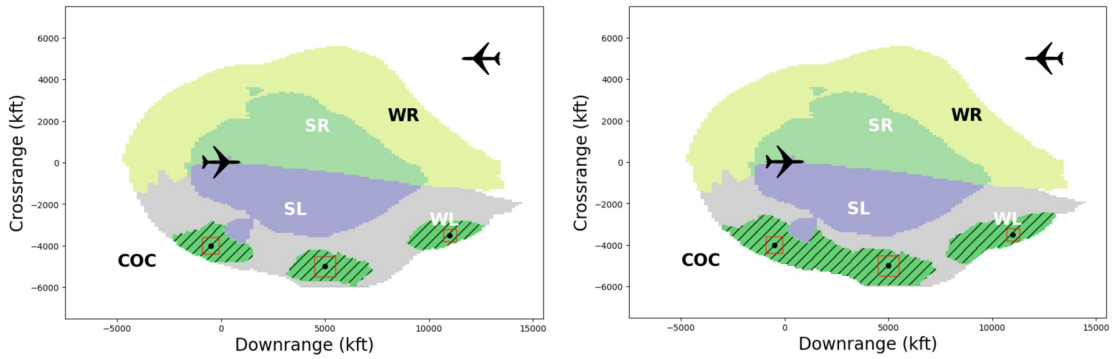
Formal verification is only as good as the specification of a system, which is also true for neural network verification. Existing specifications follow the paradigm of *data as specification*, where the local neighborhood around a reference data point is considered correct or robust. While these specifications provide a fair testbed for assessing model robustness, they are too *restrictive* for verifying any unseen test data points — a challenging task with significant real-world implications. Recent work shows great promise through a new paradigm, *neural representation as specification*, which uses neural activation patterns (NAPs) for this purpose. However, it computes the most refined NAPs, which include many redundant neurons. In this paper, we study the following problem: Given a neural network, find a minimal (general) NAP specification that is sufficient for formal verification of its robustness properties. Finding the minimal NAP specification not only expands verifiable bounds but also provides insights into which set of neurons contributes to the model's robustness. To address this problem, we propose three approaches: conservative, statistical, and optimistic. Each of these methods offers distinct strengths and trade-offs in terms of minimality and computational speed, making them suitable for scenarios with different priorities. Notably, the optimistic approach can probe potential causal links between neurons and the robustness of large vision neural networks without relying on verification tools — a task existing methods struggle to scale. Our experiments show that minimal NAP specifications use far fewer neurons than those from previous work while expanding verifiable boundaries by several orders of magnitude.

Keywords: Specifications, formal verification, and neural network robustness.

1. Introduction

The rise of deep learning in decision-critical applications has elevated safety concerns regarding AI systems, particularly their vulnerability to adversarial attacks [17, 10]. Therefore, the verification of AI systems has become increasingly important and attracted much attention from the research community [32]. The field of neural network verification largely follows the paradigm of software verification – using formal methods to verify desirable properties of systems through rigorous mathematical specifications and proofs [42]. A notable trend in existing works [23, 24, 19, 21, 39] focuses on scaling verification to larger and more complex neural networks. While scalability is undeniably important and requires the collective effort of the research community, this paper explores an orthogonal angle that has been largely overlooked: *defining meaningful specifications*.

To illustrate, nearly all existing works follow a *"data as specification"* paradigm, where the specification is defined by the consistency of local neighborhoods—often L_∞ balls—around reference data points. While local neighborhood specification provides a fair and effective testbed for evaluating neural network robustness, it primarily covers a tiny convex region of input data that can mathematically be described by adding noise to the reference point, as illustrated by the red bounding box in Figure 1(*a*). This small region is constrained by it's adversarial examples. It is too restrictive to adequately address a broader, unseen test set, which are real data sampled from the underlying distribution.



(*a*) The NAP specifications cover larger and more flexible regions than the local neighborhoods specifications.

(*b*) A coarser (more general) NAP specification extends much larger verifiable boundaries, from 21.37% to 54.43% of the total area.

Figure 1: Verifiable regions of the WL class in ACAS_Xu [22] for a head-on encounter with $a_{\text{prev}} =$ COC, $\tau = 0$s. Red boxes represent local neighborhood specifications around reference points, while green hatched areas indicate NAP specifications in the WL region.

Ideally, the specification should produce a verifiable region that includes all data points from the same class, where the data points may be distributed in a non-linear and non-convex manner within the input space. For instance, consider the safety-critical Airborne Collision Avoidance System for Unmanned Aircraft (ACAS Xu), the neural network processes an input representing the state of the aircraft and outputs one of five advisories: Clear of Conflict (COC), Weak Left (WL), Strong Left (SL), Weak Right (WR), or Strong Right (SR). As shown in the gray irregular-shaped region in Figure 1, which represents the WL class, this complexity highlights the need for flexible and robust specifications capable of accurately capturing such intricate distributions. Unlike local neighborhood specifications, manually defining such a specification is impractical. This poses a tricky chicken-egg problem: machine learning is necessary because it's challenging to formally write down a precise definition (aka specification); but to be able to verify machine learning models, a formal specification would be needed. We argue that, in order to tackle this challenge, a separate learning algorithm for specifications is necessary. In this view, the *"data as specification"* paradigm is a simple but extremely overfitted algorithm for specification learning, which simply picks a small neighborhood of a reference data point in the input space. To this end, Geng et al. [16] proposes using a new and more promising paradigm - *"neural representation as specification"*, which learns a specification in the representation space of the trained machine learning model in the form of neural activation patterns (NAPs). NAPs are value abstractions of hidden neurons which have been shown useful for understanding the inference of a model [18].

Most importantly, a well-trained neural network would exhibit similar activation patterns for input data from the same class, regardless of their actual distance in the input space [5, 36, 16]. This key observation suggests that if we learn a NAP—a common activation pattern shared by a certain class of data—it can serve as a candidate specification for verifying data from that class. Once the NAP specification is verified, any data exhibiting this NAP pattern can be provably classified within the corresponding class. Geometrically, NAPs offer greater flexibility than L_∞ ball specifications, enabling coverage of larger and more irregular regions, as shown by the learnt NAP in Figure 1(*a*).

However, the current approach to computing NAPs relies on a simple statistical method that assumes every neuron contributes to certifying robustness. As a result, the computed NAPs are often overly refined. This is a restrictive assumption, as many studies [15, 28, 26, 41] suggest that a significant portion of neurons may be redundant. In the spirit of Occam's Razor, we aim to systematically remove (coarsen) these neurons while preserving robustness. This leads to the key question: given a neural network, how can we identify a minimal NAP (i.e., the coarsest abstraction) sufficient for verification? This problem is important because: i) Minimal NAP specifications can cover much larger input regions than refined ones, improving generalization to unseen data. For example, as shown in Figure 1(*b*), coarsening just a few neurons results in a NAP that verifies much broader input regions than those refined ones in Figure 1(*a*); ii) Learning minimal NAP specifications reveals which neurons contribute to the robust predictions of models, shedding light on neural networks' internal behavior and ultimately helping uncover their black-box nature.

To tackle this, we propose three approaches—conservative, statistical, and optimistic—each balancing efficiency and performance differently. The first two methods iteratively improve the solution through interaction with verification tools in either deterministic or statistical manners, whereas the optimistic method estimates minimal NAPs using adversarial examples. We show that the optimistic approach can estimate essential neurons - the building blocks of minimal NAP specifications without invoking verification tools. This allows us to explore potential causal links between neurons and the robustness of large neural networks, such as VGG-19 [34] — a task at which existing methods struggle to scale. Our contributions can be summarized as follows:

- We introduce the problem of learning minimal NAP specifications, emphasizing its importance in neural network verification. We present three approaches, each offering distinct trade-offs between efficiency and performance.
- Our experiments show that minimal NAP specifications use far fewer neurons than those from previous work while expanding verifiable boundaries by several orders of magnitude.
- We estimate essential neurons in the large vision network VGG-19. Using a modified Grad-CAM map, we show that these essential neurons contribute to visual interpretability, providing strong evidence that they may also explain the model's robustness.

2. Background and Problem Formulation

2.1. NAP Specifications for Robustness Verification

Definition 1 (Neural Activation Pattern (NAP)) *Given a neural network N and an abstraction function $\mathcal{A} : N \rightarrow \mathcal{S}$ mapping neurons to an abstraction space $\mathcal{S}$, a Neural Activation Pattern (NAP) P is a tuple representing the abstracted activation states of all neurons in N. Formally,*

$$P = \langle \mathcal{A}(N_i) \mid N_i \in N \rangle,$$

where $\mathcal{A}(N_i)$ (or P_i) is the abstracted state of neuron N_i. Equivalently, $P = \mathcal{A}(N)$.

In this work, we focus on feed-forward neural networks with ReLU activations. Using a binary abstraction $\ddot{\mathcal{A}}$ for ReLU neurons, we define two states: $\mathbf{0} := 0$ (deactivation) and $\mathbf{1} := (0, \infty)$ (activation). The binary activation pattern of N triggered by an input x is denoted $\ddot{\mathcal{A}}_x(N)$. Additionally, $\mathbf{0}$ and $\mathbf{1}$ can be abstracted into a unary state $* = [0, \infty)$, yielding $\mathcal{S} := \{\mathbf{0}, \mathbf{1}, *\}$. The power set of NAPs in N, denoted $\mathcal{P}$, forms a partially ordered set.

Definition 2 (Partially ordered NAP) *For any given two NAPs $P, P' \in \mathcal{P}$, we say P' subsumes P if, for each neuron, its state in P is an abstraction of that in P'. Formally, this can be defined as:*

$$P' \preccurlyeq P \iff P'_i \preceq P_i \ \ \forall N_i \in N$$

Moreover, two NAPs P, P' are equivalent if $P \preccurlyeq P'$ and $P' \preccurlyeq P$.

For example, consider the NAP family of a simple neural network with 4 neurons, we have: $\langle *, *, *, * \rangle \preccurlyeq \langle \mathbf{1}, \mathbf{0}, *, * \rangle \preccurlyeq \langle \mathbf{1}, \mathbf{0}, \mathbf{1}, * \rangle$. Similar to local neighbor specifications, a NAP P can *specify a region in the input space X*, denoted as R_P. R_P is the set of inputs whose binary activation pattern subsumes P. Formally, we have $R_P = \{x \mid \ddot{\mathcal{A}}_x(N) \preccurlyeq P, x \in X\}$. A coarser NAP will specify larger regions than those that subsume it. Clearly, the most abstracted NAP, such as $\langle *, *, *, * \rangle$, will cover the entire input space. Recall that robustness verification involves proving that no adversarial examples exist in the specification. Geng et al. [16] argues that to qualify as NAP specifications, the following property must be satisfied:

Property 1 (NAP Robustness Property) *For any x located in the region specified by a NAP P, x must be predicted as the certain class $c \in C$ by the neural network N. Formally, we have:*

$$\forall x \in R_P \quad \forall k \in C \text{ s.t. } k \neq c \quad F_c^N(x) - F_k^N(x) > 0$$

where $F^N(\cdot)$ denotes the output logit of N. P is also denoted as P^c to represent class c.

2.2. Problem Formulation

Let $(\mathcal{P}^c, \preccurlyeq)$ be a partially ordered set corresponding to a family of class NAPs regarding some class $c \in C$. For simplicity, we omit the superscript c when the context is clear. We assume access to a verification tool, $\mathcal{V} : \mathcal{P} \to \{0, 1\}$, which maps a class NAP P to a binary set. Here, $\mathcal{V}(P) = 1$ denotes a successful verification of the underlying robustness query, while 0 indicates the presence of an adversarial example. From an alternative perspective, $\mathcal{V}(P) = 1$ also signifies that P is a NAP specification, i.e., it satisfies NAP robustness properties; whereas $\mathcal{V}(P) = 0$ implies the opposite.

It is not hard to see that $\mathcal{V}$ is monotone with respect to the NAP family $(\mathcal{P}, \preccurlyeq)$. Given $P' \preccurlyeq P$ and $\mathcal{V}(P') = 1$, it follows that $\mathcal{V}(P) = 1$. However, given $P' \preccurlyeq P$ and $\mathcal{V}(P) = 1$, we cannot determine $\mathcal{V}(P')$. In other words, refining a NAP (by increasing the number of neurons abstracted to $\mathbf{0}$ or $\mathbf{1}$) can only enhance the likelihood of successful verification of the robustness query.

Definition 3 (The minimal NAP specification problem) *Given a family of NAPs $\mathcal{P}$ and a verification tool $\mathcal{V}$, the minimal NAP specification problem is to find a NAP P such that:*

$$\forall P' \preccurlyeq P,\ P' \neq P \implies \mathcal{V}(P') = 0$$

That is, when P is minimal, any NAP P' that is strictly coarser than P will result in $\mathcal{V}(P') = 0$. The size of P, denoted by $|P|$ or s, represents the number of neurons abstracted to $\mathbf{0}$ or $\mathbf{1}$.

It is important to note that "minimal" refers to the level of abstraction, not the size of the NAP. Since $(\mathcal{P}^c, \preccurlyeq)$ is a partially ordered set, multiple minimal NAP specifications may exist, rather than a single global minimum. In such cases, selecting any one of the minimal NAPs is sufficient.

3. Learning Minimal Neural Specifications

3.1. Conservative Bottom Up Approach

We introduce COARSEN, a simple method that begins with the most refined NAP and progressively coarsens it to derive minimal NAP specifications. The most refined NAP, denoted as $\widetilde{P}$, is computed based on the activation values of training data using a statistical approach [16]. For further details, we refer interested readers to the original paper.

The COARSEN approach starts by verifying if the most refined NAP $\widetilde{P}$ successfully passes verification. For each neuron, we attempt to coarsen it, i.e., change state from $\mathbf{0}$ or $\mathbf{1}$ to $*$. If the resulting NAP no longer verifies the query, we revert it to its previous refined state; otherwise, we retain the coarsened NAP. The procedure may require up to $|N|$ calls to $\mathcal{V}$ in the worst case, as shown in Theorem 4. More details on the proof and algorithm are provided in Appendix A.

Theorem 4 *The algorithm* COARSEN *returns a minimal NAP specification with* $\mathcal{O}(|N|)$ *calls to* $\mathcal{V}$.

Even the most refined NAPs may fail to verify the robustness query, leaving no minimal NAP specification. Given the high cost of verification, we aim to efficiently find a minimal NAP specification while minimizing the number of calls to $\mathcal{V}$.

3.2. Statistical Coarsen Approach

The COARSEN algorithm initiates with the most refined NAP and progressively coarsens each neuron until verification fails. Enhancing the algorithm's performance is possible by coarsening multiple neurons during each iteration. However, a fundamental question emerges: How do we determine which *set* of neurons to coarsen in each round?

We present STOCHCOARSEN to answer this question. In this approach, we assume that each neuron is independent of the others and select neurons to coarsen statistically. Specifically, in each iteration, we randomly coarsen a subset of refined neurons in the current NAP simultaneously to see if the new NAP can pass verification. We repeat this process until the NAP size reaches s.

It's important to note that the stochastic manner of this algorithm faces the challenge of sample efficiency. For instance, if a minimal NAP specification P of size s is identified after one iteration, the probability of selecting the exact s essential neurons in P is θ^s. Consequently, if we set θ as a constant, the expected number of samples required to find the NAP becomes $(\frac{1}{\theta})^s$.

STOCHCOARSEN allows us to narrow down the estimated solution by an expected factor of θ once a valid NAP is learned. Thus, the expected number of samples and iterations are inversely related, with their product equating to the total calls to $\mathcal{V}$. By setting $\theta = e^{-\frac{1}{s}}$, we can achieve polynomial expected samples in s while minimizing the total number of calls to $\mathcal{V}$, as proven in Theorem 5. The proof and algorithm are detailed in Appendix B.

Theorem 5 *With probability* θ=$e^{-\frac{1}{s}}$, STOCHCOARSEN *learns a minimal NAP specification with* $\mathcal{O}(slog|N|)$ *calls to* $\mathcal{V}$.

Algorithm 1: OPTADVPRUNE

Input: Neural network N, input dataset X
Output: Collection of essential neurons
Function *OptAdvPrune(N, X)*

```
Essent ← ∅; P ← P̃
for x_j ∈ X do
    x'_j ← Adversarial_Attack(x_j)
    for N_i ∈ N do
        if P_i ∈ {0, 1} and Ä_{x'_j}(N_i) ⊕ P_i then
            Essent ← Essent ∪ {N_i}        /* Add essential neurons for x'_j */
    end
end
return Essent
```

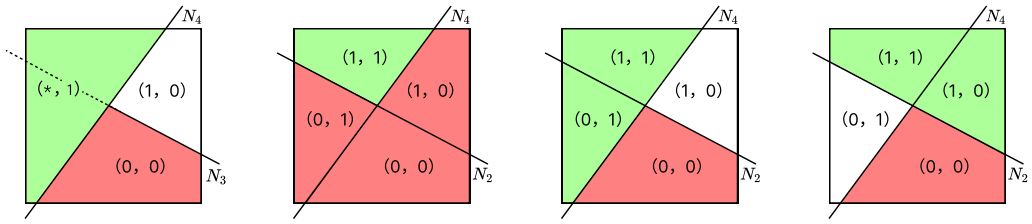
3.3. Optimistic Approach

While COARSEN and STOCHCOARSEN guarantee minimal NAP specifications, their practicality is hindered by numerous costly verification calls, making them unsuitable for time-sensitive or large-scale applications. To address this, we propose an optimistic approach that efficiently estimates NAP specifications without verification calls and provides a reliable upper bound on minimal NAP size. Central to this approach is identifying essential neurons.

Definition 6 (Essential neuron) *A neuron $n \in N$ is considered essential if it cannot be coarsened to $*$ in any minimal NAP specification. We denote the set of all essential neurons as E, defined by:*

$$E = \{N_i \mid P_i \in \{\mathbf{0}, \mathbf{1}\}, P \text{ is minimal}\}$$

Note that E is the union of the set of essential neurons from all minimal NAP specifications. It follows that $|E| \geqslant s$, where s denotes the size of the largest minimal NAP specification.



(*a*) N_4 is essential. (*b*) N_4, N_2 are essential. (*c*) N_4 is essential. (*d*) N_2 is essentibal.

Figure 2: Geometric interpretation of NAPs on essential neurons.

We introduce OPTADVPRUNE to identify essential neurons. Intuitively, it attempts to show a neuron is essential by actively falsifying NAP candidates with adversarial examples. When an adversarial example x is found, it implies that any NAP subsumed by the activation pattern of x fails verification. For instance, suppose an adversarial example x is found for a simple one-layer four-neuron neural network and its activation pattern is $\langle \mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{0} \rangle$. We can infer that NAPs like

$\langle \mathbf{1}, \mathbf{0}, 1, * \rangle$, $\langle \mathbf{1}, \mathbf{0}, *, * \rangle$, $\langle \mathbf{1}, *, *, \mathbf{0} \rangle$, and $\langle \mathbf{1}, \mathbf{0}, *, \mathbf{0} \rangle$ fail the verification. The fourth neuron N_4 is essential because coarsening it would expand P to $\langle \mathbf{1}, \mathbf{0}, *, * \rangle$, including the adversarial example x and causing verification to fail, as shown in Figure 2(*a*).

Now, consider a more general case. Suppose the NAP specification P is $\langle \mathbf{1}, \mathbf{1}, *, \mathbf{1} \rangle$, i.e., $\mathcal{V}(\langle \mathbf{1}, \mathbf{1}, *, \mathbf{1} \rangle) = 1$. We know $\mathcal{V}(\langle \mathbf{1}, \mathbf{0}, *, \mathbf{0} \rangle) = 0$ by the adversarial example x. If we coarsen the second and fourth neurons, N_2 and N_4, the specification will expand to $\langle \mathbf{1}, *, *, * \rangle$, which will cover the $\langle \mathbf{1}, \mathbf{0}, *, \mathbf{0} \rangle$, thus failing the verification. In this case, N_2 and N_4 could both be essential neurons or either one of them is essential, as illustrated in Figures 2(*b*), 2(*c*), 2(*d*). So, we simply let $\{N_2, N_4\}$ be the upper bound of essential neurons (learned from x). Formally, given a NAP P, we say a neuron N_i is in the upper bound of essential neurons E if satisfies the following condition:

1. N_i must be in the binary states, i.e., $P_i \in \{\mathbf{0}, \mathbf{1}\}$
2. There exists x such that $\ddot{\mathcal{A}}_x(N_i)$ XORs with P_i, i.e., $\exists x$ such that $\ddot{\mathcal{A}}_x(N_i) \oplus P_i = 1$

In this work, we leverage adversarial attack methods, including Projected Gradient Descent (PGD) [30] and Carlini-Wagner (CW) [8], to generate adversarial examples. By computing the upper bound of essential neurons for each example and taking their union, we estimate the overall upper bound without verification calls, as outlined in Algorithm 1. OPTADVPRUNE can also interact with verification tools, improving the learned specification by iteratively sampling additional adversarial examples until verification is achieved. While it may provide a loose upper bound, it usually requires significantly fewer verification calls than other approaches.

4. Evaluation

Setup All experiments were conducted on an Ubuntu 20.04 LTS machine with 172 GB RAM and an Intel Xeon Silver Processor. We use Marabou [24] as the verifier, with a 5-minute timeout per query. If the timeout is exceeded, the neuron is retained in the minimal NAP specification. To compare region sizes across NAPs, we propose an efficient volume estimation method for NAP-specified regions (see Appendix C for details). The most refined NAP [16] serves as the baseline.

Table 1: Overview of the learned minimal NAP specifications on the WBC benchmark. Columns indicate labels, and rows represent different approaches. $|P|$ is the NAP size, and $\#\mathcal{V}$ is the number of calls to $\mathcal{V}$. The *train* and *test* columns show the percentage of data covered by P. The $vol.$ column shows the estimated volume change (order of magnitude) relative to a baseline, which is normalized to 1.

	0					1				
	$\lvert P\rvert$	$\#\mathcal{V}$	*train*	*test*	$vol.$	$\lvert P\rvert$	$\#\mathcal{V}$	*train*	*test*	$vol.$
Baseline [16]	102	1	78.11	81.40	1	93	1	83.06	80.28	1
COARSEN	31	102	98.22	95.35	10^5	32	93	99.65	94.37	10^5
STOCHCOARSEN	42	47	94.69	92.34	10^3	45	41	94.15	91.52	10^2
OPTADVPRUNE	61	1	91.06	89.37	10^2	54	3	87.06	87.28	10^2

4.1. The Wisconsin Breast Cancer Dataset with Binary Classifier

We evaluate a four-layer neural network with 32 neurons per layer, trained on the Wisconsin Breast Cancer (WBC) dataset [43]. The most refined NAPs, $\widetilde{P}^0$ and $\widetilde{P}^1$, contain 102 and 93 neurons,

respectively. In contrast, COARSEN reduces these to 31 and 32. Additionally, STOCHCOARSEN learns NAPs of size 42 and 45 for labels 0 and 1 using only 47 and 41 verification calls, offering a less precise but more efficient solution.

Our optimistic approach, OPTADVPRUNE, efficiently estimates essential neurons while providing a reliable upper bound. For label 0, it identifies 61 essential neurons, capturing 25 of the 31 in COARSEN's minimal NAP. For label 1, it finds 54, covering 25 of the 32 essential neurons. Notably, OPTADVPRUNE requires only 1 and 3 verification calls, demonstrating its efficiency. However, this comes at the cost of a looser upper bound on the minimal NAP specification.

We see minimal NAP specifications enable verification over significantly larger input regions. The baseline NAPs, $\widetilde{P}^0$ and $\widetilde{P}^1$, verify 81.40% and 80.28% of test data for labels 0 and 1, respectively. In contrast, minimal NAPs learned by our approaches achieve substantially higher coverage. Notably, COARSEN learns minimal NAP specifications that extend test data verification coverage to 95.35% and 94.37% for labels 0 and 1, respectively. The estimated verifiable region R_P expands dramatically—by a factor of 10^5 for both labels.

Table 2: Overview of learned minimal NAP specifications on the MNIST benchmark.

	0					1					4				
	$\lvert P\rvert$	$\#\mathcal{V}$	*train*%	*test*%	*vol.*	$\lvert P\rvert$	$\#\mathcal{V}$	train%	test%	*vol.*	$\lvert P\rvert$	$\#\mathcal{V}$	train%	test%	*vol.*
Baseline [16]	751	1	79.50	71.51	1	745	1	75.01	70.11	1	712	1	77.54	75.24	1
COARSEN	480	751	98.68	98.78	10^8	491	745	98.90	98.59	10^6	506	712	98.51	97.45	10^9
STOCHCOARSEN	532	33	93.19	93.01	10^5	559	27	94.12	93.68	10^3	562	25	93.89	93.64	10^6
OPTADVPRUNE	618	15	83.13	81.32	10^2	630	18	86.02	85.51	10^2	699	21	82.66	84.12	10^2

4.2. MNIST with Fully Connected Network

To demonstrate the applicability of our methods to more complex datasets and networks, we conduct experiments using the `mnistfc_256x4` model [38], a 4-layer fully connected network with 256 neurons per layer, trained on the MNIST dataset. Due to space constraints, we present results for selected classes 0, 1, and 4; details for other classes are in Appendix D.

We compute the baseline NAPs, $\widetilde{P}^0$, $\widetilde{P}^1$, and $\widetilde{P}^4$, for labels 0, 1, and 4, with sizes 751, 745, and 712, respectively. In contrast, COARSEN reduces these to 480, 491, and 506, covering over 98% of the training and test data, with volumetric changes on the order of 10^9. STOCHCOARSEN achieves comparable results with only about 30 calls, at the cost of a 5% reduction in coverage. This highlights the generalization ability of minimal NAPs to unseen test data. In contrast, local neighborhood specifications cover *zero* test images in the MNIST dataset with $\epsilon = 0.2$, which is the maximum L_∞ verifiable bound used in VNN-COMP 2024 [7]. For label 0, OPTADVPRUNE detects 618 essential neurons and identifies 445 and 160 of the 480 neurons in COARSEN's minimal NAP. It requires fewer than 20 calls on average but incurs a 5% drop in coverage.

In summary, there is no one-size-fits-all solution for minimal specification problems. Our experiment demonstrates the distinct strengths and trade-offs of the three algorithms. While all provide correctness guarantees, COARSEN finds the minimal NAP, making it the most reliable for applications requiring strict minimality. STOCHCOARSEN offers a speed advantage over COARSEN, using 1.1x more neurons but only 5% of the verification calls, making it ideal for efficiency-focused scenarios. Lastly, OPTADVPRUNE is the fastest. When the initial set of neurons can be successfully verified, it provides a tight upper bound for the global minimum. When the initial set cannot be verified, it can provide an effective starting point for STOCHCOARSEN and COARSEN.

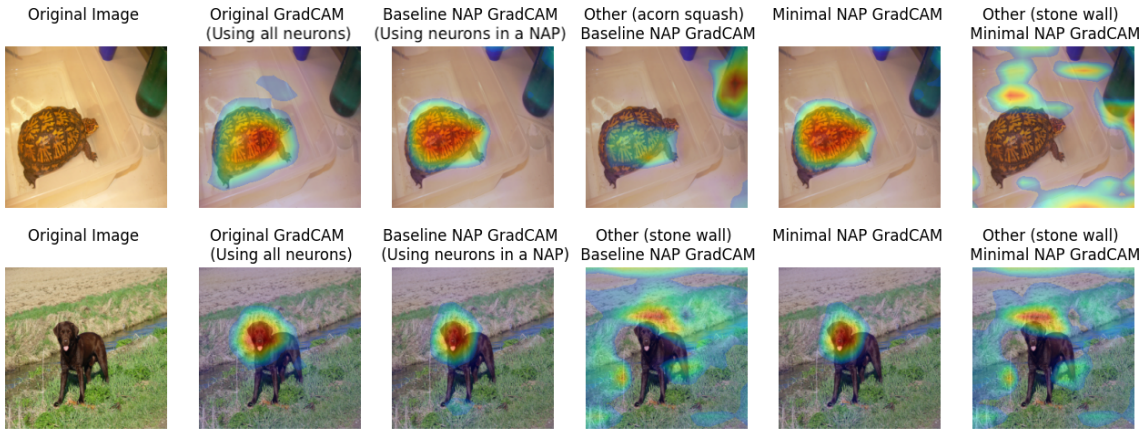


Figure 3: Visualization of hidden representations retained by NAPs. Columns show: original images, Grad-CAMs, modified Grad-CAMs using baseline NAPs (same class and another class), and modified Grad-CAMs using minimal NAPs (same class and another class).

4.3. ImageNet with Deep Convolutional Neural Network

In this experiment, we analyze the fully connected layers of the VGG-19 network pretrained on ImageNet, containing 8192 neurons. Due to the challenge of verifying NAP specifications at this scale, we focus on studying minimal NAPs estimated by OPTADVPRUNE, limiting our analysis to the top five largest classes (about 1000 training and 350 test images each). Our results show that these estimated NAPs cover a significant portion of unseen test data, highlighting their robustness as certificates. More details are in Appendix E.

NAP Captures Visual Interpretability and Inherent Robustness Neural networks learn both low- and high-level features to make classification decisions based on hidden neuron representations [5]. The robustness and consistency of a model's predictions depend on the quality of these features, with "good" hidden representations contributing to accuracy and interpretability [46]. Studies highlight the link between visual interpretability and robustness in learned features [2, 6, 11]. Although we cannot yet formally verify these estimated NAP specifications, we show through visual interpretability that they are meaningful and enhance model robustness.

To this end, we modify Grad-CAM [31] by masking neurons not included in the NAPs from the fully connected layers and recalculating the gradient flow to generate the updated map. We conduct the following experiments on image samples: 1) Compute modified Grad-CAM maps using the baseline NAPs; 2) Compute modified Grad-CAM maps using minimal NAPs; 3) Compute modified Grad-CAM maps using NAPs from different classes. All maps are validated with sanity checks [1].

Figure 3 presents the experimental results. The original Grad-CAMs highlight key image regions crucial for classification. Both refined and minimal NAPs produce nearly identical highlights, despite using a small fraction of neurons neurons, indicating that minimal NAPs preserve essential visual features. This suggests minimal NAPs capture critical aspects of VGG-19's decision-making, supporting "the NAP robustness property". Furthermore, Grad-CAMs for different classes pinpoint distinct regions, indicating that our estimated NAPs do not specify overlapping areas - a critical property for their role as specifications. Notably, these NAPs provide valuable interpretability insights. A small subset of NAP neurons can capture critical neural network dynamics, helping to

demystify their black-box nature. From a machine-checkable perspective, concise NAPs are easier to decode into human-understandable programs than refined ones, emphasizing the value of learning minimal NAPs. We leave translating NAPs into human-readable formats as future work.

NAP as a Defense Against Adversarial Attacks From a practical point of view, we believe that even before formal verification scales or NAPs are fully interpretable, NAPs can already serve as empirical certificates or defense mechanisms, as shown in prior work [29]. We demonstrate that our estimated essential neurons can defend against adversarial attacks.

We select images that are: 1) correctly predicted by the model and 2) covered by the respective NAP. On average, each NAP covers 40% of the training data for the corresponding class. For each selected image, we generate 100 adversarial examples using the Projected Gradient Descent [30] and Carlini Wagner [8] attacks. We check whether the NAPs reject the adversarial activation patterns. Both the baseline and minimal NAPs reject all adversarial examples, confirming their effectiveness as safe region specifications and potential certificates.

5. Related Work

Neural Network Verification Neural network verification has gained significant attention due to the growing use of neural networks in safety-critical systems. Its NP-hard nature resulting from the non-convexity introduced by activation functions [23] makes it a challenging task. Most existing work focuses on scalable verification algorithms. Early solver-based approaches [20, 13, 9, 37] were limited to small networks ($<$100 neurons), but state-of-the-art methods [44, 40, 27] can now verify more complex networks. It is worth mentioning that most existing work adopts local neighbourhood specifications to verify the robustness properties of neural networks [33]. Despite being a reliable measure, local neighbourhood specifications around reference data points may fail to cover test data or generalize to unseen test sets. Geng et al. [16] propose a new paradigm of NAP specifications to address this challenge. Our work advances the understanding of NAP specifications.

Neural Activation Patterns Neural activation patterns are widely used to interpret neural network decision-making. Research in feature visualization [45, 4] examines neuron activations for different inputs, while activation maximization [35] identifies inputs that maximally activate specific neurons. These works focus on learning statistical correlations between NAPs and inputs [3, 14] or prediction outcomes [29]. However, these correlations raise questions about their trustworthiness and verifiability. We introduce essential neurons, emphasizing their role in model robustness, and establish verified causal links between neurons and predictions. We believe this "identify then verify" paradigm can be extended to existing research on NAPs to certify our understanding of neural networks. We leave the exploration of this direction for our future work.

6. Conclusion

We introduce a novel challenge: learning minimal NAP specifications and demonstrating their significance in neural network verification. To address this, we propose three approaches—conservative, statistical, and optimistic, and show each offers unique trade-offs between efficiency and performance. Additionally, we use the optimistic approach to explore potential causal connections between neurons and the robustness properties of large networks like VGG-19, a task existing methods cannot scale to. Our experimental results show that minimal NAPs require significantly fewer neurons compared to prior work while expanding verifiable regions by several orders of magnitude.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[2] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in neural information processing systems*, 31, 2018.

[3] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3319–3327. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.354. URL https://doi.org/10.1109/CVPR.2017.354.

[4] Alex Bäuerle, Daniel Jönsson, and Timo Ropinski. Neural activation patterns (naps): Visual explainability of learned concepts. *CoRR*, abs/2206.10611, 2022. doi: 10.48550/arXiv.2206.10611. URL https://doi.org/10.48550/arXiv.2206.10611.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8): 1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

[6] Akhilan Boopathy, Sijia Liu, Gaoyuan Zhang, Cynthia Liu, Pin-Yu Chen, Shiyu Chang, and Luca Daniel. Proper network interpretability helps adversarial robustness in classification. In *International Conference on Machine Learning*, pages 1014–1023. PMLR, 2020.

[7] Christopher Brix, Stanley Bak, Taylor T. Johnson, and Haoze Wu. The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results, 2024. URL https://arxiv.org/abs/2412.19985.

[8] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, Los Alamitos, CA, USA, may 2017. IEEE Computer Society. doi: 10.1109/SP.2017.49. URL https://doi.ieeecomputersociety.org/10.1109/SP.2017.49.

[9] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis - 15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings*, volume 10482 of *Lecture Notes in Computer Science*, pages 251–268. Springer, 2017. doi: 10.1007/978-3-319-68167-2_18. URL https://doi.org/10.1007/978-3-319-68167-2_18.

[10] Thomas G. Dietterich and Eric Horvitz. Rise of concerns about AI: reflections and directions. *Commun. ACM*, 58(10):38–40, 2015. doi: 10.1145/2770869. URL https://doi.org/10.1145/2770869.

[11] Yinpeng Dong, Hang Su, Jun Zhu, and Fan Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *arXiv preprint arXiv:1708.05493*, 2017.

[12] Martin E. Dyer and Alan M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.*, 17(5):967–974, 1988. doi: 10.1137/0217060. URL https://doi.org/10.1137/0217060.

[13] Rüdiger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. *CoRR*, abs/1705.01320, 2017. URL http://arxiv.org/abs/1705.01320.

[14] D. Erhan, Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. 2009.

[15] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

[16] Chuqin Geng, Nham Le, Xiaojie Xu, Zhaoyue Wang, Arie Gurfinkel, and Xujie Si. Towards reliable neural specifications. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 11196–11212. PMLR, 2023. URL https://proceedings.mlr.press/v202/geng23a.html.

[17] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6572.

[18] Divya Gopinath, Kaiyuan Wang, Mengshi Zhang, Corina S. Pasareanu, and Sarfraz Khurshid. Symbolic execution for deep neural networks. *CoRR*, abs/1807.10439, 2018. URL http://arxiv.org/abs/1807.10439.

[19] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 3–29. Springer, 2017. doi: 10.1007/978-3-319-63387-9_1. URL https://doi.org/10.1007/978-3-319-63387-9_1.

[20] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 3–29. Springer, 2017. doi: 10.1007/978-3-319-63387-9_1. URL https://doi.org/10.1007/978-3-319-63387-9_1.

[21] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Comput. Sci. Rev.*, 37:

100270, 2020. doi: 10.1016/j.cosrev.2020.100270. URL https://doi.org/10.1016/j.cosrev.2020.100270.

[22] Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks, 2017. URL https://arxiv.org/abs/1702.01135.

[23] 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 Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 97–117. Springer, 2017. doi: 10.1007/978-3-319-63387-9_5. URL https://doi.org/10.1007/978-3-319-63387-9_5.

[24] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljic, David L. Dill, Mykel J. Kochenderfer, and Clark W. Barrett. The marabou framework for verification and analysis of deep neural networks. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*, volume 11561 of *Lecture Notes in Computer Science*, pages 443–452. Springer, 2019. doi: 10.1007/978-3-030-25540-4_26. URL https://doi.org/10.1007/978-3-030-25540-4_26.

[25] Percy Liang, Omer Tripp, and Mayur Naik. Learning minimal abstractions. In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 31–42. ACM, 2011. doi: 10.1145/1926385.1926391. URL https://doi.org/10.1145/1926385.1926391.

[26] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021. doi: 10.1016/J.NEUCOM.2021.07.045. URL https://doi.org/10.1016/j.neucom.2021.07.045.

[27] Jingyue Lu and M. Pawan Kumar. Neural network branching for neural network verification. *CoRR*, abs/1912.01329, 2019. URL http://arxiv.org/abs/1912.01329.

[28] Lu Lu, Yeonjong Shin, Yanhui Su, and George E. Karniadakis. Dying relu and initialization: Theory and numerical examples. *CoRR*, abs/1903.06733, 2019. URL http://arxiv.org/abs/1903.06733.

[29] Anna Lukina, Christian Schilling, and Thomas A. Henzinger. Into the unknown: Active monitoring of neural networks. In Lu Feng and Dana Fisman, editors, *Runtime Verification - 21st International Conference, RV 2021, Virtual Event, October 11-14, 2021, Proceedings*, volume 12974 of *Lecture Notes in Computer Science*, pages 42–61. Springer, 2021. doi: 10.1007/978-3-030-88494-9_3. URL https://doi.org/10.1007/978-3-030-88494-9_3.

[30] 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, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

[31] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017. doi: 10.1109/ICCV.2017.74.

[32] Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Toward verified artificial intelligence. *Commun. ACM*, 65(7):46–55, June 2022. ISSN 0001-0782. doi: 10.1145/3503914. URL https://doi.org/10.1145/3503914.

[33] Yuval Shapira, Eran Avneri, and Dana Drachsler-Cohen. Deep learning robustness verification for few-pixel attacks. *Proc. ACM Program. Lang.*, 7(OOPSLA1), apr 2023. doi: 10.1145/3586042. URL https://doi.org/10.1145/3586042.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL https://api.semanticscholar.org/CorpusID:14124313.

[35] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6034.

[36] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

[37] Vincent Tjeng, Kai Yuanqing Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HyGIdiRqtm.

[38] VNNCOMP. Vnncomp, 2021. URL https://sites.google.com/view/vnn2021.

[39] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29909–29921, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/fac7fead96dafceaf80c1daffeae82a4-Abstract.html.

[40] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29909–29921, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/fac7fead96dafceaf80c1daffeae82a4-Abstract.html.

[41] Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 14913–14922. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01467. URL https://openaccess.thecvf.com/content/CVPR2021/html/Wang_Convolutional_Neural_Network_Pruning_With_Structural_Redundancy_Reduction_CVPR_2021_paper.html.

[42] Jeannette M. Wing. A specifier's introduction to formal methods. *Computer*, 23(9):8–24, 1990. doi: 10.1109/2.58215. URL https://doi.org/10.1109/2.58215.

[43] William Wolberg, Olvi Mangasarian, Nick Street, and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C5DW2B.

[44] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=nVZtXBI6LNn.

[45] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *CoRR*, abs/1506.06579, 2015. URL http://arxiv.org/abs/1506.06579.

[46] Yu Zhang, Peter Tiňo, Ales Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Trans. Emerg. Top. Comput. Intell.*, 5(5):726–742, 2021. doi: 10.1109/TETCI.2021.3100641. URL https://doi.org/10.1109/TETCI.2021.3100641.

Appendix A. Algorithm and Proof for COARSEN

Algorithm 2: COARSEN

Input: The neural network N
Output: A minimal NAP specification P
Function *Coarsen(N)*
  $P \leftarrow \widetilde{P}$
  if $\mathcal{V}(P) == 0$ **then**
    return $None$; /* Return None if the most refined NAP fails verification */
  else
    for N_i ***in*** N **do**
      $s \leftarrow P_i$;
      $* \leftarrow P_i$; /* Try to coarsen this neuron */
      if $\mathcal{V}(P) == 0$ **then**
        $P_i \leftarrow s$; /* Refine the neuron back if the verification fails */
    end
    return P;
  end
end

Theorem 7 (Property of COARSEN) *The algorithm* COARSEN *returns a minimal NAP specification with* $\mathcal{O}(|N|)$ *calls to* $\mathcal{V}$.

Proof Let P be the NAP returned by COARSEN. Our goal is to show that any P' smaller than P results in $\mathcal{V}(P') = 0$. To construct such a smaller P', we need to apply the coarsen action through a collection of neurons. According to the algorithm, $\mathcal{V}(P') = 0$. In the worst case, the algorithm needs to iterate through each neuron in N, resulting in a runtime complexity of $\mathcal{O}(|N|)$. ■

Appendix B. Algorithm and Proof for STOCHCOARSEN

Setting the sample probability θ Setting s poses a challenge in practice, as we assume that s is always provided in STOCHCOARSEN. However, this can be addressed by dynamically updating θ based on the result of $\mathcal{V}(P)$ [25]. With θ from theorem 5, STOCHCOARSEN finds a NAP specification with probability $\left(e^{-1/s}\right)^s = e^{-1}$. Thus, we aim to set θ such that the $Pr(\mathcal{V}(P) = 1) = e^{-1}$. Intuitively, if a sampled NAP P is a specification, we decrease θ so more neurons will be coarsened. Similarly, if P is not a specification, θ needs to be increased.

Given that $\theta \in [0, 1]$, we can parameterize it using the Sigmoid function $\sigma(\lambda) = \left(1 + e^{-\lambda}\right)^{-1}$, where $\lambda \in (-\infty, \infty)$. Since $Pr(\mathcal{V}(P) = 1)$ depends on θ as well, we express it as a function of λ, $g(\lambda) = Pr(\mathcal{V}(P) = 1)$. Then, setting $Pr(\mathcal{V}(P) = 1) = e^{-1}$ can be achieved through the following minimization problem:

$$L(\lambda) = \frac{1}{2}(g(\lambda) - e^{-1/s})^2 \tag{1}$$

The loss function $L(\lambda)$ can be minimized by statistical learning using stochastic gradient descent. With a step size η, update λ using $\lambda \leftarrow \lambda - \eta \frac{dL}{d\lambda}$. Note that $\frac{dL}{d\lambda}$ can be expressed as:

$$\frac{dL}{d\lambda} = (g(\lambda) - e^{-1/s})\frac{dg(\lambda)}{d\lambda} \tag{2}$$

Given $g(\lambda) = Pr(\mathcal{V}(P) = 1)$, we can replace $g(\lambda)$ with $\mathcal{V}(P)$ for stochastic gradient update. Additionally, since $\frac{dg(\lambda)}{d\lambda} > 0$, we simply ignore it as its multiplication effect can be represented by η. Therefore, the final update rule is given by:

$$\lambda \leftarrow \lambda - \eta(\mathcal{V}(P) - e^{-1}) \tag{3}$$

Algorithm 3: STOCHCOARSEN

Input: The neural network N, the probability θ, and the size s
Output: A minimal NAP specification P
Function *StochCoarsen(cur_neurons, θ, s)*

```
P ← P̃; cur_neurons ← N
if V(P) == 0 then
    return None            /* Return None if the most refined NAP fails
     verification */
else
    while |P| > s do
        P ← Sample_NAPs(cur_neurons, θ)
        if V(P) == 1 then
            found_neurons ← ∅
            for N_i in N do
                if P_i == P̃_i then
                    found_neurons ← found_neurons ∪ {N_i}
            end
            cur_neurons ← found_neurons          /* Reduce search space */
        else
            P ← Sample_Naps(cur_neurons, θ)      /* Sample a new NAP */
        end
    end
    return P                        /* Return the minimal NAP of size s */
end
end
```

Theorem 8 (Property of STOCHASTIC_COARSEN) *With probability θ=$e^{-\frac{1}{s}}$,* SAMPLE_COARSEN *learns a minimal NAP specification with $\mathcal{O}(slog|N|)$ calls to $\mathcal{V}$.*

Proof Let's first estimate the number of calls that SAMPLE_COARSEN makes to $\mathcal{V}$. We denote the number of calls as $\mathcal{C}(P^L)$, where P^L is the most refined NAP. Then $\mathcal{C}(P^L)$ can be computed recursively using the following rule:

$$\mathcal{C}(P^L) = \begin{cases} |P| & \text{if } |P^L| \leqslant s+1 \\ 1 + \mathbb{E}[(1-\mathcal{V}(P))\mathcal{C}(P) + \mathcal{V}(P)\mathcal{C}(P^L)] & \text{otherwise} \end{cases} \tag{4}$$

where P is the sampled NAP. By assumption, there exists a NAP $P^S \preccurlyeq P$ of size s that passes the verification. Define $G(P) = \neg(P^S \preccurlyeq P)$, which is 0 when P^S subsumes P, i.e., all essential

neurons in P^S shows up in the sampled P. This follows that $Pr(G(P) = 0) = Pr(P^S \preccurlyeq P) = \theta^s$. Note that $G(P) \geqslant \mathcal{V}(P)$, as the NAP P^S suffices to prove the robustness query. We can estimate the upper bound of $\mathcal{C}(P^L)$ by replacing $\mathcal{V}$ with G:

$$\mathcal{C}(P^L) \leqslant 1 + \mathbb{E}[(1 - G(P))\mathcal{C}(P) + G(P)\mathcal{C}(P^L)] \tag{5}$$

$$\leqslant 1 + \theta^s \mathbb{E}[\mathcal{C}(P)|P^S \preccurlyeq P] + (1 - \theta^s)\mathcal{C}(P^L) \tag{6}$$

$$\leqslant \mathbb{E}[\mathcal{C}(P)|P^S \preccurlyeq P] + \theta^{-s} \tag{7}$$

We now denote $\mathcal{C}(n) = max_{|P|=n}\mathcal{C}(P)$ as the maximum over NAP of size n. Note that given $P^S \preccurlyeq P$, $|P| = s + N$ where N is a binomial random variable with $\mathbb{E}(N) = \theta(n - s)$.

Using the bound $\mathcal{C}(n) \leqslant (1 - \theta^n)\mathcal{C}(n-1) + \theta^n\mathcal{C}(n)\theta^{-s}$, we can observe that $\mathcal{C}(n) \leqslant \frac{\theta^{-s}}{1-\theta^n} \cdot n$. In addition, when n is large enough, $\mathcal{C}(n)$ is concave, then by use Jensen's inequality:

$$\mathcal{C}(n) \leqslant \mathcal{C}(\mathbb{E}[s + N]) + \theta^{-s} = \mathcal{C}(s + \theta(n - s)) + \theta^{-s} \tag{8}$$

To solve the recurrence, this gives us:

$$\mathcal{C}(n) \leqslant \frac{\theta^{-s} log n}{log \theta^{-1}} + s + 1 \tag{9}$$

The equation above illustrates a tradeoff between reducing the number of iterations (by increasing $\log \theta^{-1}$) and reducing the number of samples (by decreasing θ^{-s}). To minimize $\mathcal{C}(n)$, we need to set θ so that the gradient of $\mathcal{C}(n)$ w.r.t θ^{-1} is 0. This gives us: $\frac{sx^{s-1}}{\log x} - \frac{x^{s-1}}{\log^2 x} = 0$, solving this gives us $\theta = e^{-\frac{1}{s}}$. Consequently, the upper bound becomes $\mathcal{C}(n) = es \log n + s + 1 = O(s \log n)$. ■

Appendix C. Volume Estimation of R_P

Conceptually, NAP specifications typically correspond to significantly larger input regions compared to local neighbourhood specifications. This serves as the primary motivation for utilizing NAPs as specifications. However, previous work lacks sufficient justification or evidence to support this claim. In this section, we propose a simple method for approximating the volume of R_P, i.e., the region corresponding to a NAP P. This allows us to: 1) quantify the size difference between R_P and L_∞ ball specifications; 2) gain insights into the volumetric change from the most refined NAP specification to the minimal NAP specification.

Computing the exact volume of R_P is at least NP-hard, as determining the exact volume of a polygon is known to be NP-hard [12]. Moreover, computing the exact volume of R_P can be even more challenging due to its potential concavity. To this end, our method estimates the volume of R_P by efficient computation of an orthotope that closely aligns with R_P, as illustrated in Figure 4. We briefly describe it as follows:

Finding an anchor point The first step is to find an anchor point to serve as the center of the orthotope. Ideally, this anchor point should be positioned close to the center of R_P to ensure a significant overlap between the orthotope and R_P. However, computing the actual center of R_P is

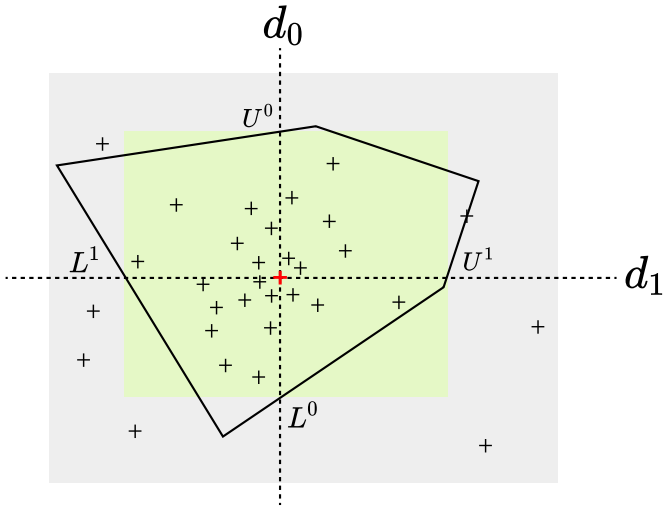


Figure 4: Volume Estimation of R_P using an orthotope in a 2-dimensional case. The gray rectangle represents the input space, with the training set depicted by a collection of data points $+$. The polygon corresponds to some NAP P. Initially, we identify an anchor point, denoted by $+$. Then, we construct the orthotope, represented by the green rectangle, by extending upper and lower bounds starting from the anchor point until it extends beyond P.

costly. Thus, we look for a pseudo-center from the training set X that resides in R_P. This pseudo-center can be computed by finding the point that uses the smallest L_∞ ball to cover other data points, solved as the following optimization problem:

$$c_{\text{pseudo}} = \arg\min_{x \in R_P} \max_{x' \in R_P} \|x - x'\|_\infty$$

where $R_P = \{x \mid \ddot{\mathcal{A}}_x(N) \preccurlyeq P, x \in X\}$. When $|X|$ is small, c_{pseudo} can be computed directly; for larger $|X|$, a statistical computation strategy is required.

Constructing the orthotope Once the pseudo-center c_{pseudo} is determined, we want to create an orthotope around c_{pseudo} to closely align with R_P. The orthotope is constructed by determining pairs of upper and lower bounds $U^{(i)}$ and $L^{(i)}$ for each dimension i. Specifically, $U^{(i)}$ and $L^{(i)}$ are computed through expansion in two opposite directions from c_{pseudo} along dimension i until they extend beyond R_P. This expansion can be expressed as:

$$\max_{U^{(i)}} \{x' \in R_P \,|\, x' := c_{\text{pseudo}} + U^{(i)}\} \; ; \; \max_{L^{(i)}} \{x' \in R_P \,|\, x' := c_{\text{pseudo}} - L^{(i)}\}$$

Here, $U^{(i)}$ and $L^{(i)}$ represent the upper and lower bounds in dimension i respectively, originating from c_{pseudo}. These bounds can be efficiently calculated with binary search.

The choice of the archer point is crucial in our approach. If it is located at a corner of R_p, the volume calculation will be highly biased. This can pose a problem when we seek to understand the volumetric change from the most refined NAP specification to the minimal NAP specification. Ad-

ditionally, using the orthotope as an estimator provides convenience in understanding the volumetric change simply by examining differences in each input dimension.

Appendix D. Learned Minimal NAP Specifications on the MNIST Benchmark (Complement)

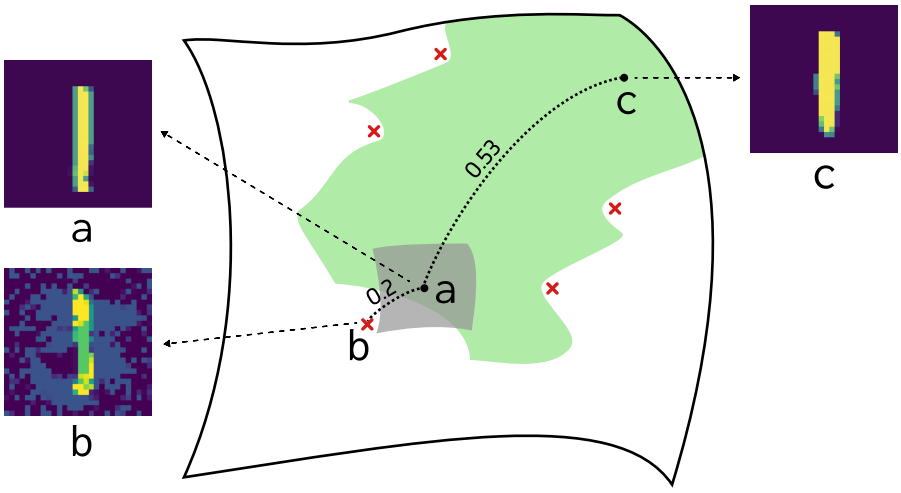


Figure 5: The comparison between NAP specifications (green region) and L_∞ ball specifications (gray region) on the MNIST dataset. Image a is the reference image, and image c is the closest among all 6000 training images of digit 1, with an L_∞ distance of 0.5294. However, c cannot be verified using the L_∞ ball specification, as an adversarial example b exists at an L_∞ distance of 0.2. Note that this is not a limitation of the underlying verification engines but rather an intrinsic limitation of the specifications. In contrast, NAP specifications allow verification of unseen test set data like c.

Table 3: Overview of the size of learned minimal NAP specifications on the MNIST benchmark.

	2				3				5				6			
	$\|P\|$	$\#\mathcal{V}$	train	test	$\|P\|$	$\#\mathcal{V}$	train	test	$\|P\|$	$\#\mathcal{V}$	train	test	$\|P\|$	$\#\mathcal{V}$	train	test
Baseline [16]	751	1	79.50	80.51	745	1	86.01	85.11	712	1	77.54	80.24 1	712	1	77.54	80.24
COARSEN	480	751	98.68	98.78	491	745	98.90	98.59	506	712	98.51	97.45	503	708	98.81	98.39
STOCHCOARSEN	532	33	93.19	93.01	559	27	94.12	93.68	562	25	93.89	93.64	542	29	98.81	98.39
OPTADVPRUNE	618	15	83.13	71.32	630	18	79.02	76.51	699	21	82.66	84.12	681	20	85.98	87.63

Table 4: Overview of the size of learned minimal NAP specifications on the MNIST benchmark.

	7				8				9			
	$\|P\|$	$\#\mathcal{V}$	train	test	$\|P\|$	$\#\mathcal{V}$	train	test	$\|P\|$	$\#\mathcal{V}$	train	test
Baseline [16]	751	1	79.50	80.51	745	1	86.01	85.11	712	1	77.54	80.24
COARSEN	480	751	98.68	98.78	491	745	98.90	98.59	506	712	98.51	97.45
STOCHCOARSEN	532	33	93.19	93.01	559	27	94.12	93.68	562	25	93.89	93.64
OPTADVPRUNE	618	15	83.13	71.32	630	18	79.02	76.51	699	21	82.66	84.12

Appendix E. Learned Minimal NAP Specifications on ImageNet with Deep Convolutional Neural Network (Complement)

Table 5: Overview of the size of learned minimal NAP specifications on the ImageNet benchmark.

	box_turtle		labrador_retriever		acorn_squash		confectionery		stone_wall	
	$\|P\|$	*test*	$\|P\|$	*test*	$\|P\|$	*test*	$\|P\|$	*test*	$\|P\|$	*test*
Baseline [16]	1978	39.42	691	39.61	1003	29.23	878	33.84	971	31.54
OPTADVPRUNE	1863	39.42	661	41.28	823	25.68	845	23.07	865	39.61
GRADIENT_SEARCH	611	53.30	572	87.01	301	52.40	256	44.23	260	54.90

Appendix F. More Visual Examples for NAP Captures Visual Interpretability and Inherent Robustness

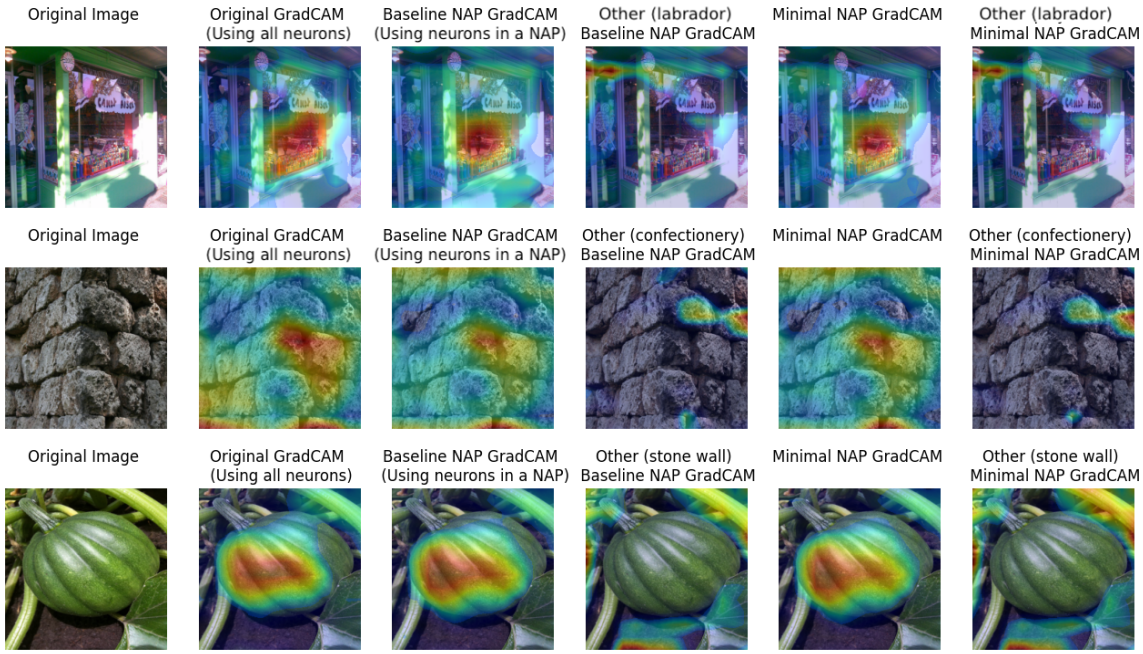


Figure 6: Visualization of hidden representations retained by NAPs. Columns show: original images, Grad-CAMs, modified Grad-CAMs using baseline NAPs (same class and another class), and modified Grad-CAMs using minimal NAPs (same class and another class).