Bias Field Robustness Verification of Large Neural Image Classifiers

Patrick Henriksen¹ patrick.henriksen19@imperial.ac.uk Kerstin Hammernik¹² k.hammernik@imperial.ac.uk Daniel Rueckert¹² d.rueckert@imperial.ac.uk Alessio Lomuscio¹ a.lomuscio@imperial.ac.uk

- ¹ Imperial College London London, UK
- ² Technical University of Munich Munich, Germany

Abstract

We present a method for verifying the robustness of neural network-based image classifiers against a large class of intensity perturbations that frequently occur in computer vision. These perturbations, or intensity inhomogeneities, can be modelled by a spatially varying, multiplicative transformation of the intensities by a bias field. We illustrate an encoding of bias field transformations into neural network operations to exploit neural network formal verification toolkits. We extend the toolkit VeriNet with the above encoding, GPU support, input-domain splitting and a symbolic interval propagation pre-processing step. Finally, we show that the resulting implementation, VeriNetBF, can analyse models with up to 11M tuneable parameters and 6.5M ReLU nodes trained on the CIFAR-10 ImageNet and NYU fastMRI datasets.

1 Introduction

While image classifiers implemented via large and deep ReLU-based networks have achieved remarkable results in terms of accuracy [III], the resulting implementations remain fragile and susceptible to adversarial attacks [III]. This is highly problematic in safety-critical areas such as medical imaging or autonomous navigation where incorrect classifications may lead to misdiagnoses or accidents. While work on adversarial attacks focuses on the generation of counterexamples, it cannot provide formal robustness guarantees. In contrast, the area of verification of neural networks [III] is concerned with methods and tools for proving the *robustness* of neural networks, *i.e.*, giving formal guarantees that an input model is not susceptible to any possible attack within a given input range. As a byproduct of this analysis, counterexamples can also be generated and used for further training.

Robustness is typically studied in the context of verification with respect to particular specifications. In the area of image classification, particular attention has been given to local robustness with respect to adversarial noise attacks. These are very small modifications of the image, often not detectable by the human eye. In a nutshell, a model is locally robust





with respect to an input image if it produces the same classification class for all variations of the image within an ε ball, *i.e.*, all images in which each pixel value may vary within ε of the original value. The notion above is related to a particular input, e.g., an image. A different, albeit related notion, is that of global robustness, i.e., the robustness of the model with respect to all the points within the whole set of possible inputs. In a verification context, research into global robustness methods for neural networks is limited at present. While global robustness is a long-term objective, local robustness remains of great importance since local various robustness guarantees can provide valuable validation of the model and any counterexample found can be used in retraining to further robustify the model.

Considerable work has been devoted to the development of scalable methods for the verification of neural networks, summarized later in this section. However, two important challenges remain open in the literature. Firstly, the methods developed so far mostly concern verifying the robustness of models against "noise-based" perturbations, *i.e.*, regions defined by ε balls in the value of the input pixels. These capture realistic sensor errors; however, models should also be assessed against "semantical" variations, *e.g.*, linear transformations expressing variations in contrast, luminosity, scaling, rotation, etc. Secondly, scalability remains an issue in current research. Present methods generally do not scale to models of few hundred thousands ReLU nodes, even for white-noise perturbations. Solutions to semantical transformations often suffer from even more severe scalability problems.

In this paper we make a contribution towards developing methods that are both scalable and support complex semantical variations in the input space as depicted in Fig. 1. Specifically, we present the following results:

- We present a novel notion of semantical robustness for image classifiers against bias field transformations [59], to encode spatially-varying contrast and luminosity changes.
- We develop a highly scalable, parallel algorithm for the verification of robustness with respect to semantic changes expressed as bias field variations of the input image.
- We evaluate experimentally Verinet_{BF}, the resulting toolkit, which incorporates various optimisations to aid scalability, on large CIFAR-10, ImageNet and medical imaging models, with up to 11m tuneable parameters (6.5M ReLU nodes).

 applied to low dimensional datasets such as MNIST and CIFAR-10. In contrast, we here (i) introduce a novel method for verifying bias field perturbations, a class of perturbations that is natural in applications and has not been analysed before, and (ii) evaluate the method on an ImageNet classifier with 11M tunable parameters and 6.5M nodes.

We note that statistical verifiers (see e.g. $[\Box, \Box]$) often scale better than formal verifiers. However, this work differs significantly from formal verification in that it is not sound; the results only have statistical guarantees on correctness. Moreover, to the best of our knowledge, no work in statistical verification considers bias fields as we do here.

Research into methods supporting robustness against perturbations other than white noise is still quite limited. In [[1]] the authors consider affine transformations, [6, 21] support affine and geometric transformations, and [24, 25] consider colour-space and geometric transformations. Some of these are particularly relevant in our context as they encode the transformation into the neural network by inserting extra layers, encoding the transformations, in the early layers of the model. We follow a similar approach but differ from those by developing and inserting bias field transformations into the network instead, introducing spatially varying contrast and brightness changes as depicted in Fig. 1.

Most of the work mentioned above only considers networks with a few hundred thousand ReLU nodes at most, with some exceptions. The work in [22] analysed larger networks with 1M ReLU nodes via GPU accelerated computations; however, this network was trained on the low-dimensional CIFAR-10 dataset with techniques to ease verification and had a reported accuracy of 34.8% only. The method from [56] considers large ImageNet classifiers (VGG16 and VGG19); however, the perturbation radii under consideration was limited to 2×10^{-7} , significantly smaller than the precision of 8-bit images (1/255). In contrast, we here consider ImageNet classifier with 70% accuracy and perturbation radii larger than 1/255.

Bias field transformations are not only relevant to model spatially-varying lightning conditions in natural images, but also occur naturally in medical imaging. In Magnetic Resonance Imaging (MRI), the true image intensities are modulated by a multiplicative, smoothly varying bias field due to patient-induced electrodynamic interactions and imperfections in the MRI system. While a number of approaches have been proposed to remove intensity inhomogeneities from MR images [52], 53], these approaches can fail and often leave residual intensity inhomogeneities in the images. This further motivates the development of verification algorithms for image classifiers that are robust against bias field perturbations.

2 Preliminaries

We now introduce important notations for formal verification of neural networks. This includes definitions on robustness, symbolic interval propagation, and bias field perturbations.

In what follows we deal with ReLU-based Feed-forward Neural Networks (FFNN) [12]. We assume hidden layers to be governed by ReLU activation functions ReLU(z) = max(0,z) and the output of a layer $\Omega : \mathbb{R}^{h_i} \to \mathbb{R}^{h_{i+1}}$ to be $\Omega(\text{ReLU}(z))$, where the ReLU is applied element-wise and the layer's input is a linear combination of outputs from preceding layers. Overall the FFNN computes a function $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}^m$ where \mathcal{N} is obtained by successively applying the layer transformations for all layers. In what follows we only consider fully-connected, convolutional and average-pooling layers [12].

Most verification approaches are tailored towards checking models for robustness wrt noise changes in the input, *i.e.* that the model's classification for a given image does not change if small changes are made to the pixel values, installing local robustness specification.

Definition 1 (Local Robustness). A FFNN $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}^m$ is locally robust to some input constraints ψ_x and output constraints ψ_y iff $\mathcal{N}(\mathbf{x})$ satisfies ψ_y for all inputs \mathbf{x} satisfying ψ_x .

Local robustness against White-Noise (WN) perturbations can be formalised from Definition 1 by taking $\psi_{\mathbf{x}}$ to encode an ε ball of the input and $\psi_{\mathbf{y}}$ to express equality to $\mathcal{N}(\mathbf{x})$. More specifically, input constraints may be expressed via box constraints of the form $\psi_{\mathbf{x}} = \{\mathbf{x}_i^{low} \leq \mathbf{x}_i \leq \mathbf{x}_i^{up}\}_{i=0...n}$, where $\mathbf{x}_i^{low}, \mathbf{x}_i^{up} \in \mathbb{R}$, and the output bounds are limited to linear constraints on the network's output variables as formalised below.

Definition 2 (WN Robustness Decision Problem). Given a tuple $P = (\mathcal{N}, \psi_{\mathbf{x}}, \psi_{\mathbf{y}})$, where \mathcal{N} is a FFNN, $\psi_{\mathbf{x}}$ are constraints on \mathcal{N} 's input variables $\psi_{\mathbf{x}} = \{\mathbf{x}_{k}^{low} \leq \mathbf{x}_{k} \leq \mathbf{x}_{k}^{up} | \mathbf{x}_{k}^{low}, \mathbf{x}_{k}^{up} \in \mathbb{R}\}_{\forall k}$ and $\psi_{\mathbf{y}}$ are linear constraints on \mathcal{N} 's output variables, the White-Noise (WN) Robustness Problem concerns establishing whether or not $\mathcal{N}(\mathbf{x})$ satisfies $\psi_{\mathbf{y}}$ for all \mathbf{x} satisfying $\psi_{\mathbf{x}}$.

If the answer to the problem above is positive, we say that the network is robust for the image in question (wrt ψ_x, ψ_y); otherwise we say it is not robust.

The problem above is challenging due to the non-linearity introduced by the network's ReLUs, which potentially leads a branch-and-bound algorithm to consider an exponential number of branches. A common approach used to alleviate this problem consists in relaxing the network by calculating linear bounds with *Symbolic Interval Propagation (SIP)*.

Standard-SIP (SSIP). In [\square] lower and upper equations for the bounds of each input node *i* ($z_i^{low}(x) = z_i^{up}(x) = x_i$) are introduced and propagated layer-by-layer through the network. Since the bounding equations are linear in the network's input variables, and the input variables are constrained by box-constraints ψ_x , concrete bounds for the network output can be obtained by solving the corresponding linear program. If the bounds show that the output remains within the specifications ψ_y for all inputs in ψ_x , then we can conclude that the verification problem is satisfied. If this is not the case, the problem cannot immediately be solved and more advanced procedures may be attempted [\square].

Reversed-SIP (**RSIP**). In [**LD**] the bounding equations are introduced on the final layers and the resulting variables are substituted from the expressions of the previous layers. Formally, for each node *i* at the layer of interest *k*, the equations are initialised as $z_i^{low}(\mathbf{h}^k) = \mathbf{h}_i^k$ and $z_i^{up}(\mathbf{h}^k) = \mathbf{h}_i^k$ where \mathbf{h}_i^k is the input variable at node *i* in layer *k*. The output bounds can then be determined by considering the input bounds in the final equations obtained.

Although RSIP is computationally more expensive than SSIP, RSIP may lead to tighter bounds. Both approaches can be used independently or in a branch-and-bound approach [1].

Sound and Complete Verification. Verification algorithms are said to be sound if any property determined to hold by the algorithm does hold. If the algorithm is also theoretically guaranteed to provide an answer to any verification query, it is complete. Verifying local robustness properties for NNs is an NP-complete problem [IX]; thus, complete algorithms do have a worst-case exponential complexity in the number of ReLU nodes. Consequentially, incomplete approaches may outperform complete approaches in practice.

Bias Fields. Different from all verification approaches in the area, we here consider specifications given via bias field transformations. Such transformations occur frequently in medical imaging. For example, in Magnetic Resonance Imaging (MRI) the images are often corrupted by a smoothly varying intensity inhomogeneity or bias field. Here we model bias fields as both a multiplicative and additive modification of the image intensities.

Definition 3 (Bias Field). A bias field with k terms, $\boldsymbol{b}_a : \mathbb{R}^n$, is defined as $\boldsymbol{b}_a = \sum_{t=0}^k \boldsymbol{a}_t \boldsymbol{b}_t$, for some real-valued coefficients $\boldsymbol{a} \in \mathbb{R}^k$ and corresponding vectors $\boldsymbol{b} \in \mathbb{R}^n$.

Definition 4 (Bias Field transformation). A bias field transformation is defined as $T_{\boldsymbol{b}}(\boldsymbol{x};\boldsymbol{a}) = \boldsymbol{x}\boldsymbol{b}_{\boldsymbol{a}^{mul}}^{mul} + \boldsymbol{b}_{\boldsymbol{a}^{add}}^{add}$ where $\boldsymbol{x} \in \mathbb{R}^{n}$, \boldsymbol{b}^{mul} and \boldsymbol{b}^{add} are bias fields and $\boldsymbol{a} = \{\boldsymbol{a}^{mul}, \boldsymbol{a}^{add}\}$ are the coefficients of $\boldsymbol{b}_{\boldsymbol{a}^{mul}}^{mul}$ and $\boldsymbol{b}_{\boldsymbol{a}^{add}}^{add}$.

The definitions above are given for one-dimensional inputs; however, extending them to multi-dimensional inputs such as images is straight-forward. Intuitively, bias field transformations can be used to define the input range x, as we show in the next section.

3 Verifying Bias Field Robustness

In this section we present the novel specification of bias field robustness for classification networks, show how bias field robustness problems can be encoded in such a way that most verification toolkits can handle them, and propose a sound algorithm for verifying large models against this specification.

Definition 5 (BF Robustness Decision Problem). Given a tuple $P_b = (\mathcal{N}, \mathbf{x}, T_b, \psi_a, \psi_y)$, where \mathcal{N} is a FFNN, $\psi_a = \{\mathbf{a}_k^{low} \leq \mathbf{a}_k \leq \mathbf{a}_k^{up} | \mathbf{a}_k^{low}, \mathbf{a}_k^{up} \in \mathbb{R}\}_{\forall k}$ are constraints on the coefficients of the bias field transformation T_b and ψ_y are linear constraints on \mathcal{N} 's output variables, the Bias Field (BF) Robustness Decision Problem concerns establishing whether or not $\mathcal{N}(T_b(\mathbf{x}; \mathbf{a}))$ satisfies ψ_y for all \mathbf{a} satisfying ψ_a .

So, the BF robustness problem differs from the WN robustness problem in Definition 2 in that we consider perturbations of the coefficient a parameterising the bias field transformation T_b , rather than white noise on the network's input x. Note that the bias fields in the bias field transformation are arbitrary; thus, T_b can be highly non-linear in the spatial dimension. The only requirement of linearity is with respect to the coefficients a. Most importantly, BF local robustness encapsulates a wide range of verification problems, including the WN robustness problem, affine transformations, and higher-order polynomial transformations.

Even though the BF robustness problem is highly expressive, we now show that it can be reduced to a problem conforming to Definition 2. We use an approach similar to the work in [2], 2]. The work encodes some transformations by augmenting the network with extra layers; however, different from here, it does not consider bias field transformations.

Definition 6 (BF Network). Given an FFNN $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}^m$, an input $\mathbf{x} \in \mathbb{R}^n$ and a bias field transformation $T_{\mathbf{b}}$, the Bias Field (BF) Network $\mathcal{N}' : \mathbb{R}^k \to \mathbb{R}^m$ is defined as $\mathcal{N}' = \mathcal{N}(FC_{\mathbf{b}}(\mathbf{a}))$ where $\mathbf{a} \in \mathbb{R}^k$ and $FC_{\mathbf{b}}$ is a fully connected layer with weight matrix $W : \mathbb{R}^n \times \mathbb{R}^k$ defined as $W = [\mathbf{x}\mathbf{b}_0^{mul}, ..., \mathbf{x}\mathbf{b}_k^{mul}, \mathbf{b}_0^{add}, ..., \mathbf{b}_l^{add}]$ and \mathbf{b}_i^{mul} and \mathbf{b}_j^{add} are the additive and multiplicative terms of $T_{\mathbf{b}}$.

The BF network encodes the parameterised transformation $T_{\boldsymbol{b}}(\boldsymbol{x};\boldsymbol{a})$ for a given input \boldsymbol{x} in the sense that $\mathcal{N}'(\boldsymbol{a}) = \mathcal{N}(T_{\boldsymbol{b}}(\boldsymbol{x};\boldsymbol{a}))$. As we show below verifying the original network \mathcal{N} against bias field transformations can be solved via verifying WN robustness on \mathcal{N}' .

Theorem 1. Any BF Robustness problem $P_b = (\mathcal{N}, \mathbf{x}, T_b, \psi_a, \psi_y)$ can be solved by answering the WN robustness problem for $P = (\mathcal{N}', \psi_a, \psi_y)$, where \mathcal{N}' is the BF network of \mathcal{N}, \mathbf{x} and T_b as defined in Definition 6.

Proof sketch. This follows directly from the fact that the BF transformation $T_b(x; a)$ from the BF local robustness problem is encoded as the first layer of \mathcal{N}' .

From the above it follows that algorithms that can solve WN robustness can also solve BF robustness by applying BF networks. We show in the sequel that modifications can be made to this algorithm to make this computationally efficient.

BF Robustness Verification Algorithm. We now describe a SIP-based verification algorithm for solving the BF robustness problem as reported in Algorithm 1. We only consider classification networks; thus we assume that the output constraint is on the form $\bigwedge_{h \neq c} (\mathcal{N}(T_{\boldsymbol{b}}(\boldsymbol{x};\boldsymbol{a}))_c > \mathcal{N}(T_{\boldsymbol{b}}(\boldsymbol{x};\boldsymbol{a}))_h)$ where c is the decired classification class.

For clarity the algorithm is presented at high-level only. Some implementation details are crucial for computational efficiency reasons and are presented later in Section 4.

6

On line 2 the BF network as defined in Definition 6 is created and the branch queue is initialised with the first branch, here represented by "None" since the initial branch does not add additional constraints to the In the main loop, linear problem. 10 bounds $(l_k(\boldsymbol{a}), u_k(\boldsymbol{a}))$ are calculated via SIP for the network's output nodes, 11 such that $l_k(\boldsymbol{a}) < \mathcal{N}'(\boldsymbol{a})_k < u_k(\boldsymbol{a})$. 12 Note that several versions of SIP can 13 be used for this step (see Section 4). 14 15

The bounds from SIP are used in a satisfiability call to a linear program (LP) solver together with each negated output constraint in Line 8. The call is

Algorithm 1 BF Robustness Verification	
1:	procedure VERIFICATION($\mathcal{N}, \mathbf{x}, \psi_{\mathbf{a}}, T_{\mathbf{b}}, \psi_{\mathbf{y}}$)
2:	\mathcal{N}' , queue $\leftarrow \text{BFNet}(\mathcal{N}, \boldsymbol{x}, T_{\boldsymbol{b}})$, [None]
3:	while queue is not empty do
4:	$\psi_{branch} \leftarrow \text{queue.get}(0)$
5:	bounds \leftarrow SIP($\mathcal{N}', \psi_{\boldsymbol{a}}, \psi_{branch}$)
6:	shouldBranch = False
7:	for $\psi^h_{oldsymbol{y}}\in\psi_{oldsymbol{y}}$ do
8:	$\boldsymbol{a} \leftarrow \text{LP}(\text{bounds}, \boldsymbol{\psi}_{\boldsymbol{a}}, \boldsymbol{\psi}_{branch}, \neg \boldsymbol{\psi}_{\boldsymbol{y}}^{h})$
9:	if <i>a</i> is not None then
10:	shouldBranch = True
11:	$\boldsymbol{a}' \leftarrow \text{localSearch}(\boldsymbol{a}, \mathcal{N}', \boldsymbol{\psi}_{\boldsymbol{a}}, \neg \boldsymbol{\psi}_{\boldsymbol{y}}^h)$
12:	if $\mathcal{N}'(\boldsymbol{a}')$ satisfies $\neg \boldsymbol{\psi}_{\boldsymbol{y}}^h$ then
13:	return Not-Robust, a'
14:	if shouldBranch then
15:	$\psi_{b1}, \psi_{b2} \leftarrow \text{Branch}(\mathcal{N}, \psi_{a}, \psi_{y}, \psi_{branch})$
16:	queue.append(ψ_{b1}, ψ_{b2})
17:	return Robust

performed to determine a violation of the output constraints; if the output h is larger than c $(\boldsymbol{\psi}_{\mathbf{v}}^{h} = \mathcal{N}'(\boldsymbol{a})_{c} \leq \mathcal{N}'(\boldsymbol{a})_{h})$, then this call generates a counterexample to be returned (\boldsymbol{a}).

Lines 10-13 encode a gradient-descent search to identify valid counterexamples in the proximity of a spurious counterexample [**[]**]. For each clause $\psi_{\mathbf{v}}^{h} = \mathcal{N}'(\mathbf{a})_{c} \geq \mathcal{N}'(\mathbf{a})_{h}$, the local search is initialised at **a** with the loss $L(\mathbf{a}) = \mathcal{N}'(\mathbf{a})_c - \mathcal{N}'(\mathbf{a})_h$. After each step of gradient descent, the output a_s is clipped to the input constraints ψ_a and checked to determine whether it is a valid counterexample by evaluating whether $\mathcal{N}'(\boldsymbol{a}_s)$ violates $\psi_{\boldsymbol{v}}$. The local search terminates if a counterexample is found or after a pre-determined number of iterations.

If a valid counterexample is located, the value "Not-Robust" is returned and the procedure terminates. If the LP-calls are unsatisfiable for all clauses, no counterexample exists in the current branch and it is robust. Otherwise, the problem cannot be solved with the current constraints and a branch and bound phase is initiated. The resulting branch and bound (BaB) phase can be performed in many ways. We propose a BaB strategy in the next section.

We end this Section by remarking some properties of the procedure.

Theorem 2 (Algorithm 1 is sound). If Algorithm 1 returns "Robust" for $P_b = (\mathcal{N}, \mathbf{x}, T_b, \psi_a, \psi_v)$, then the network is robust for P_b , according to Definition 5; conversely, if Algorithm 1 returns "Not-Robust, a''' then the network is not robust and a' is a valid counter example.

Proof sketch. This follows from Algorithm 1 being sound for the tuple $P = (\mathcal{N}, \psi_a, \psi_v)$ [and solving the BF robustness problem P_b is equivalent solving P from Theorem 1. \square While the algorithm is sound, it may not be complete depending on the branching strategy; this, along with a efficient implementation, is discussed in detail in the next section.

4 The Verinet_{BF} Verification Toolkit

In this section we present Verinet_{BF}, a Python-based implementation of Algorithm 1. Verinet_{BF} utilises parts of the open-source SIP-based toolkit VERINET [**ID**] for the inner verification loop, extended with the bias field modification as described in the previous section and the novel contributions regarding efficient implementation introduced below. Verinet_{BF} is the first tool we are aware of that can verify models against bias field transformations.

SSIP pre-processing. The most computationally expensive part of Algorithm 1 concerns the generation of the bounds for the network, suitably transformed to account for the bias field transformation (*line* 5). To achieve scalability, Verinet_{BF} performs this step with a combination of SSIP and RSIP. Verinet_{BF} first perform a pass of the computationally efficient SSIP analysis to identify stable nodes, and only performs a more precise but computationally costly RSIP on the remaining nodes. The idea of a pre-processing step to filter out stable nodes is not new, *e.g.* [12] uses non-symbolic interval propagation for this purpose. However, SSIP produces more succinct bounds than non-symbolic interval propagation [11], and is particularly efficient for problems with low input dimensionality such as bias fields.

The computational efficiency of SSIP follows from the fact the the complexity is $O(nm^2l)$, where n, m, l are the number of input nodes, nodes in the largest layer and number of layers, respectively. In contrast, RSIP has the complexity $O(m^3l^2)$, which is reduced to $O(um^2l^2)$ with the pre-processing step where u is the number of unstable nodes in the largest layer. For networks with $n \ll m$, the complexity of SSIP is negligible compared to RSIP.

GPU enabled SIP. It is known that the computations in both RSIP and SSIP are dominated by matrix multiplications which can benefit from massive parallelisation via Graphics Processing Units (GPUs) [22]. However, for larger networks the memory usage of RSIP can easily exceed the memory limitations of most GPUs. To solve this issue, memory requirements for RSIP are first estimated, similarly to [22]. Then, RSIP is run repeatedly with smaller batches of nodes that do not violate the GPU memory requirements.

Input domain splitting. Most SoA verification methods rely on ReLU splitting for the branch and bound phase. This is because for white-noise robustness the input dimensions are usually larger than the network's layers and the consensus in the field is that splitting nodes in smaller layers is generally beneficial. However, for BF robustness, the input dimension, *i.e.* parameters of the bias field, is typically much smaller than the number of nodes in hidden layers. Thus, in contrast to most other tools, Verinet_{BF} performs input splitting.

In Verinet_{BF} splitting an input node a_i with lower and upper bounds a_i^{low}, a_i^{up} is implemented by adding the constraints $a_i < (a_i^{low} + a_i^{up})/2$ and $a_i > (a_i^{low} + a_i^{up})/2$ in the corresponding separate resolution branches. To identify the input node to be split, we use a novel heuristic based on the bounds calculated by RSIP for the network's output nodes, similar to the ReLU heuristic used in [13].

Output constraints on the form $\psi_{\mathbf{y}} = \{\mathcal{N}'(\mathbf{a})_i > \mathcal{N}'(\mathbf{a})_j\}_{\forall i \neq j}$ can be proven robust by (i) increasing the lower bound for output *i*, or (ii) decreasing the upper bound for outputs *j*. Since the lower and upper bounds for an output node *g* are linear on the form $l_g(\mathbf{a}) = \sum_k c_{g,k}^{low} \mathbf{a}_k + b_g^{low}$ and $u_g(\mathbf{a}) = \sum_k c_{g,k}^{up} \mathbf{a}_k + b_g^{up}$ we take the coefficients $c_{g,k}^{low}$ and $c_{g,k}^{up}$ as an indicator of the influence input \mathbf{a}_k has on the lower and upper output bounds of node *k*,

respectively. Thus, for each input node, we calculate the score

$$s(\boldsymbol{a}_k) = c_{k,i}^{low} \cdot (\boldsymbol{a}_k^{up} - \boldsymbol{a}_k^{low}) + \frac{1}{|C|} \sum_{j \neq i} c_{k,j}^{up} \cdot (\boldsymbol{a}_k^{up} - \boldsymbol{a}_k^{low}).$$

The divisor |C|, representing the number of classes, is intended to capture that increasing the lower bound of node *i* helps towards proving all other classes robust; improving the upper bounds of other nodes *j* only towards proving that one class robust. After calculating $s(a_k)$ for all input nodes, we split the node with the largest score.

Rounding errors. Some research has indicated that rounding errors may be a concern in verification of large networks [1]. To alleviate this, we used 64-bit precision in SIP; this is in contrast to VERINET and much of the field that uses 32-bit. We note that in experiments, 64-bit precision significantly increased the runtime, in many instances by almost 100%.

Completeness. Theorem 2 guarantees that the implementation presented here is sound. Note, however, that in principle completeness cannot be guaranteed because any inputsplitting heuristic cannot provide a guarantee that all ReLU nodes will become stable in a finite number of splits. However, the approach presented here still differs significantly from incomplete SIP algorithms without a branch-and-bound phase, such as [22, 28, 51] in that we would expect ReLU nodes to become stable as we split more and more input nodes. Thus, we would expect most problems to be solved given enough time.

5 Experimental Results

In view of Theorem 1, in this section we experimentally evaluate Verinet_{BF} in solving BF robustness problems. We begin by evaluating its performance on a CIFAR-10 network as is common in WN verification experiments. This is followed by ablation testing to assess the optimisations introduced in Section 4. Next we use Verinet_{BF} to verify BF robustness of an ImageNet model of 6.5M ReLU nodes which is, as discussed in the introduction, significantly larger than networks considered in WN verification. Finally, we evaluate the method on a medical imaging classification network trained on MRI images.

Bias field perturbations. To reason about bias field robustness, we consider multiplicative bias fields, parametrised by polynomials of degree d [53, 54]. Each element x, y of the bias field matrices \boldsymbol{b}^{d_1,d_2} are then defined as $\boldsymbol{b}_{x,y}^{d_1,d_2} = x^{d_1} \cdot y^{d_2}$, the bias field is $\boldsymbol{b}_{\boldsymbol{a}} = \sum_{d_1,d_2=0}^{d} \boldsymbol{a}^{d_1,d_2} \boldsymbol{b}^{d_1,d_2}$ and the full BF transformation is $T_{\boldsymbol{b}}(\boldsymbol{x};\boldsymbol{a}) = \boldsymbol{x}\boldsymbol{b}_{\boldsymbol{a}}$. Here, $\boldsymbol{a} \in \mathbb{R}^{d_1d_2}$ defines the perturbation coefficients, and $x, y \in \mathbb{N}$ encode the spatial position in the image. In our experiments, we use polynomials of order d = 3, resulting in 16 coefficients. In multiplicative bias fields, the entries of the position matrix are scaled with the image intensities.

In the following experiments we constrained the constant term of the bias field $(a^{0,0})$ to $[1 - \varepsilon, 1 + \varepsilon]$ and the remaining terms to $[-\varepsilon/15, \varepsilon/15]$. Thus, the BF perturbations are centred around the original image; the divisor accounts for the fact that we have 15 spatially dependent terms, each of which affects every pixel. For all experiments we used $\varepsilon \in \{0.01, 0.02, 0.03, 0.05, 0.1, 0.25\}$. We note that the maximal perturbation of a pixel under the bias field is a multiplication by $1 \pm 2\varepsilon$, *e.g.* with $\varepsilon = 0.05$ and a pixel with original value 0.5, the perturbation is limited to [0.45, 0.55]. Indeed, given the same pixel value and ε , this is also the reachable range under additive WN perturbation as defined in Definition 2.

CIFAR-10 experiments (Figure 2(a)). For the CIFAR-10 network, we used a 10-layer CNN with 8.6M tunable parameters, 100 thousand ReLU nodes and a test set accuracy of



Figure 2: Robust and non-robust cases for the CIFAR-10, ImageNet and MRI benchmarks (a-c) and mean runtimes for the CIFAR-10 and MRI queries excluding timeouts (d).

86% (see supplementary material). All CIFAR-10 experiments were run on a workstation with a Ryzen 3700X 3.6 GHz 8-core CPU, 64 GB RAM, GeForce RTX 2080 8 GB GPU and Ubuntu 18.04 with Linux kernel 5.8.0. We selected the first 100 images $(3 \times 32 \times 32 \text{ pixels})$ from the test set, analysed them with the network and used correctly classified images in our experiments (86 images in total). For verification, we used a 1800s timeout.

Verinet_{BF} found that most cases were robust for bias fields with $\varepsilon \leq 0.05$. This is in contrast to previous experiments on similar networks with white noise perturbations. In [13] a similar CIFAR-10 convolutional network was shown to have several non-robust cases for $\varepsilon = 2 \times 10^{-4}$, and no robust cases were found with $\varepsilon \geq 8 \times 10^{-4}$.

Ablation study (Table 1). To evaluate the effects of the implementation-specific choices we presented in Section 4, we ran a number of ablation studies on the CIFAR-10 network with $\varepsilon = 0.01$. The tests indicate that the SIP and Interval Propagation pre-processing steps introduced in Section 4 had speed-ups of around 3.7, and both in combination around 16.3.

ImageNet Experiments (Figure 2(b)). For the ImageNet experiments, we deployed a reduced version of NFNet-F0 [\square , \square] with 6.5M ReLU nodes and a a top-1% accuracy of 70% (see supplementary material). All ImageNet experiments were conducted on an high-performance computing cluster running CentOS Linux 8.2.2004; each verification query used one NVIDIA RTX 6000 24 GB GPU, with 4 CPU cores and 24 GB RAM. We selected 10 images from the validation set $(3 \times 256 \times 256 \text{ pixels})$ and ran verification queries on the correctly classified ones (10 images in total) with a 48h timeout. The mean runtime for cases that did not time out ranged from 16h ($\varepsilon = 0.01$) to 34h ($\varepsilon = 0.25$).

As shown in Figure 2(b), the model was proven to be robust to bias field perturbations with $\varepsilon \leq 0.02$ for most of the analysed images. Again, we note that the network is robust for the given images to relatively large perturbations, this is in contrast to WN robustness for not robustly-trained large networks that have been proven robust to tiny perturbations only, $e.g. \varepsilon = 10^{-7}$ [53]. These features of the model may to some extent be explained by the contrast and brightness data-augmentation used during training. However, it is known that data augmentation is not particularly effective for other transformations, e.g. rotation. So these findings warrant further research into whether neural networks may be more robust to BF perturbations than we would expect from comparable experiments on WN robustness.

MRI Experiments (Figure 2(c)). Having assessed the significant scalability of the approach, we now turn to a different domain to evaluate the suitability of the approach in the area of medical imaging. To do so, we trained a 10 layer CNN with 295 thousand tunable parameters, 52 thousand ReLU nodes that classifies MR images, obtaining an accuracy of 92.9%. The dataset contains 18000 brain MR images from 3 sequences (classes), extracted from the publicly available fastMRI DICOM data [21, 13]. NYU fastMRI investigators provided data but did not participate in analysis or writing of this paper. The network was robust for most inputs with $\varepsilon \leq 0.25$; so, differently from the other experiments, we also used $\varepsilon = 0.5$. The experiments were run on the same machine as the CIFAR-10 experiments.

Figure 2(c)) shows that the network is significantly more robust to bias field perturbations than the previously considered CIFAR-10 and ImageNet networks. This indicates that the network may have learned to remove bias field perturbations due to their potential presence in the training set. The result could also be influenced by the fact that the classification problem that the network is solving, is relatively simple.

The experiments demonstrate that the verification method here put forward can concretely help the validation of neural networks before deployment, particularly in the context of safety-critical application. A case in point here is the MRI experiments for which we were able to demonstrate considerable robustness up to a value of $\varepsilon = 0.25$. Adversarial examples, including those obtained via verification, can be used as part of a training procedure to further improve the robustness of the model. We also remark that, even when robustness cannot be shown, precisely understanding the limitations of the networks under study may prompt further training or changes to the architecture of the model.

6 Conclusions

10

Formally verifying the correctness of neural classifiers before deployment may contribute to their safety and trustworthiness. While progress has been made recently, present verification methods have limited scalability and are normally defined with respect to white noise perturbations. In this paper we considered the problem of verifying networks with respect to bias field perturbations. We proposed a method to reduce bias field verification problems to white noise verification problems for which we can utilise existing approaches verification approaches. We implemented and evaluated the approach on a verification toolkit optimised for bias field transformations. The results showed that the toolkit could verify robustness for an ImageNet classifier with 6.5M ReLU nodes against bias field transformations with large perturbation radii. The results also indicate that the tested networks are robust for significantly larger perturbations than what we would normally expect against white noise verification. In future work we would like to explore the encoding here presented for bias field robustness for other specifications of interest.

Acknowledgements

This work is partly funded by the UKRI Centre for Doctoral Training in Safe and Trusted Artificial Intelligence (grant number EP/S023356/1). Alessio Lomuscio is supported by a Royal Academy of Engineering Chair in Emerging Technologies.

References

- M. E. Akintunde, A. Lomuscio, L. Maganti, and E. Pirovano. Reachability analysis for neural agent-environment systems. In *Proceedings of the 15th International Conference on Principles* of Knowledge Representation and Reasoning (KR16), pages 184–193. AAAI Press, 2018.
- [2] M. E. Akintunde, A. Kevorchian, A. Lomuscio, and E. Pirovano. Verification of RNN-based neural agent-environment systems. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI19)*, pages 6006–6013. AAAI Press, 2019.
- [3] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J.P. Vielma. Strong mixed-integer programming formulations for trained neural networks. In *Integer Programming and Combinatorial Optimization*, volume 11480 of *LNCS*, pages 27–42. Springer, 2020.
- [4] S. Bak. Execution-guided overapproximation (ego) for improving scalability of neural network verification. In 3rd International Workshop on Verification of Neural Networks, 2020.
- [5] S. Bak, H.-D. Tran, K. Hobbs, and T.T. Johnson. Improved geometric path enumeration for verifying relu neural networks. In *Proceedings of the 32nd International Conference on Computer Aided Verification (CAV20)*, pages 66–96, 2020.
- [6] M. Balunovic, M. Baader, G. Singh, T. Gehr, and M. Vechev. Certifying geometric robustness of neural networks. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS2019)*, pages 15313–15323. Curran Associates, Inc., 2019.
- [7] T. Baluta, Z.L. Chua, K.S. Meel, and P. Saxena. Scalable quantitative verification for deep neural networks. arXiv preprint arXiv:2002.06864, 2021.
- [8] B. Batten, P. Kouvaros, A. Lomuscio, and Y. Zheng. Efficient neural network verification via layer-based semidefinite relaxations and linear cuts. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, (*IJCAI21*), pages 2184–2190. International Joint Conferences on Artificial Intelligence Organization, 2021.
- [9] E. Botoeva, P. Kouvaros, J. Kronqvist, A. Lomuscio, and R. Misener. Efficient verification of neural networks via dependency analysis. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI20)*. AAAI Press, 2020.
- [10] A. Brock, S. De, S.L. Smith, and K. Simonyan. High-performance large-scale image recognition without normalization. arXiv preprint arXiv:2102.06171, 2021.
- [11] Andrew Brock, Soham De, and Samuel L. Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. In *Proceedings of the 9th International Conference on Learning Representations (ICLR21)*. AAAI Press, 2021.
- [12] Andrew Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021.

12 HENRIKSEN ET AL.: VERIFICATION OF LARGE NEURAL IMAGE CLASSIFIERS

- [13] R. Bunel, J. Lu, I. Turkaslan, P.H.S. Torr, P. Kohli, and M.P. Kumar. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research (JMLR20)*, 21(42):1–39, 2020.
- [14] A. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press Cambridge, 2016.
- [15] P. Henriksen and A. Lomuscio. Efficient neural network verification via adaptive refinement and adversarial search. In *Proceedings of the 24th European Conference on Artificial Intelligence* (ECAI20), 2020.
- [16] P. Henriksen and A. Lomuscio. Deepsplit: An efficient splitting method for neural network verification via indirect effect analysis. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, (IJCAI21)*, pages 2549–2555. International Joint Conferences on Artificial Intelligence Organization, 2021.
- [17] K. Jia and M. Rinard. Exploiting verified neural networks via floating point numerical error. *arXiv preprint arXiv:2003.03021*, 2020.
- [18] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In 29th International Conference on Computer Aided Verification, volume 10426 of Lecture Notes in Computer Science, pages 97–117. Springer, 2017.
- [19] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljic, D. L. Dill, M. J. Kochenderfer, and C. W. Barrett. The marabou framework for verification and analysis of deep neural networks. In *Proceedings of the 31st International Conference* on Computer Aided Verification (CAV19), pages 443–452, 2019.
- [20] F. Knoll, J. Zbontar, A. Sriram, M.J. Muckley, M. Bruno, A. Defazio, M. Parente, K.J. Geras, J. Katsnelson, H. Chandarana, Z. Zhang, M. Drozdzalv, A. Romero, M. Rabbat, P. Vincent, J. Pinkerton, D. Wang, N. Yakubova, E. Owens, C.L. Zitnick, M.P. Recht, D.K. Sodickson, and Y.W. Lui. fastmri: A publicly available raw k-space and dicom dataset of knee images for accelerated mr image reconstruction using machine learning. *Radiology: Artificial Intelligence*, 2(1): e190007, 2020.
- [21] P. Kouvaros and A. Lomuscio. Formal verification of cnn-based perception systems. arXiv preprint arXiv:1811.11373, 2018.
- [22] P. Kouvaros and A. Lomuscio. Towards scalable complete verification of relu neural networks via dependency-based branching. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, (IJCAI21)*, pages 2643–2650. International Joint Conferences on Artificial Intelligence Organization, 2021.
- [23] C. Liu, T. Arnon, C. Lazaru, C. Barrett, and M.J. Kochenderfer. Algorithms for verifying deep neural networks. arXiv preprint arXiv:1903.06758, 2019.
- [24] J. Mohapatra, T.W. Weng, P.Y. Chen, S. Liu, and L. Daniel. Towards verifying robustness of neural networks against semantic perturbations. In *Proceedings of the 2020 IEEE/CVF Conference* on Computer Vision and Pattern Recognition (CVPR20), pages 241–249, 2020.
- [25] K. Pei, Y. Cao, J. Yang, and S. Jana. Towards practical verification of machine learning: The case of computer vision systems. arXiv preprint arXiv:1712.01785, 2017.
- [26] V. Rubies-Royo, R. Calandra, D. M. Stipanovic, and C. Tomlin. Fast neural network verification via shadow prices. *arXiv preprint arXiv:1902.07247*, 2019.

- [27] F. Serre, C. Müller, G. Singh, M. Püschel, and M. Vechev. Scaling polyhedral neural network verification on GPUs. In *Proceedings of the 4th Conference on Machine Learning and Systems (MLSys21)*, 2021.
- [28] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev. Fast and effective robustness certification. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems (NeurIPS18)*, pages 10802–10813. Curran Associates, Inc., 2018.
- [29] G. Singh, R. Ganvir, M. Püschel, and M. Vechev. Beyond the single neuron convex barrier for neural network certification. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS19)*, pages 15098–15109, 2019.
- [30] G. Singh, T. Gehr, M. Püschel, and M. Vechev. Boosting robustness certification of neural networks. In *Proceedings of the 7th International Conference on Learning Representations* (*ICLR19*), 2019.
- [31] G. Singh, T. Gehr, M. Püschel, and P. Vechev. An abstract domain for certifying neural networks. In ACM on Programming Languages, volume 3, pages 1–30. ACM Press, 2019.
- [32] J. G. Sled, A. P. Zijdenbos, and A. C. Evans. A nonparametric method for automatic correction of intensity nonuniformity in mri data. *IEEE Transactions on Medical Imaging*, 17(1):87–97, 1998.
- [33] M. Styner, C. Brechbuhler, G. Szckely, and G. Gerig. Parametric estimate of intensity inhomogeneities applied to mri. *IEEE Transactions on Medical Imaging*, 19(3):153–165, 2000.
- [34] M. Tincher, C. R. Meyer, R. Gupta, and D. M. Williams. Polynomial modeling and reduction of rf body coil spatial inhomogeneity in mri. *IEEE Transactions on Medical Imaging*, 12(2): 361–365, 1993.
- [35] V. Tjeng, K. Xiao, and R. Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *Proceedings of the 7th International Conference on Learning Representations* (ICLR19), 2019.
- [36] H.D. Tran, S. Bak, W. Xiang, and T.T. Johnson. Verification of deep convolutional neural networks using imagestars. In *Proceedings of the 32nd International Conference on Computer Aided Verification (CAV20)*, pages 18–42, 2020.
- [37] H.D. Tran, X. Yang, D.M. Lopez, P. Musau, L.V. Nguyen, W. Xiang, S. Bak, and T.T. Johnson. Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *Proceedings of the 32nd International Conference on Computer Aided Verification (CAV20)*, 2020.
- [38] N. J. Tustison, B. B. Avants, P. A. Cook, Y. Zheng, A. Egan, P. A. Yushkevich, and J. C. Gee. N4itk: Improved n3 bias correction. *IEEE Transactions on Medical Imaging*, 29(6):1310–1320, 2010.
- [39] U. Vovk, F. Pernus, and B. Likar. A review of methods for correction of intensity inhomogeneity in mri. *IEEE Transactions on Medical Imaging*, 26(3):405–421, 2007.
- [40] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Formal security analysis of neural networks using symbolic intervals. In *Proceedings of the 27th USENIX Security Symposium (USENIX18)*, 2018.
- [41] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Efficient formal safety analysis of neural networks. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems (NeurIPS18)*, pages 6367–6377. Curran Associates, Inc., 2018.

- [42] L. Weng, P.-Y. Chen, L. Nguyen, M. Squillante, A. Boopathy, I. Oseledets, and L. Daniel. PROVEN: Verifying robustness of neural networks with a probabilistic approach. In *Proceedings* of the 36th International Conference on Machine Learning (ICML19), pages 6727–6736. PMLR, 2019.
- [43] J. Zbontar, F. Knoll, A. Sriram, T. Murrell, Z. Huang, M.J. Muckley, A. Defazio, R. Stern, P. Johnson, M. Bruno, M. Parente, K.J. Geras, J. Katsnelson, H. Chandarana, Z. Zhang, M. Drozdzalv, A. Romero, M. Rabbat, P. Vincent, N. Yakubova, J. Pinkerton, D. Wang, E. Owens, C.L. Zitnick, M.P. Recht, D.K. Sodickson, and Y.W. Lui. fastmri: An open dataset and benchmarks for accelerated mri. arXiv preprint arXiv:1811.08839, 2018.