Recently, there is growing concern that machine-learning models, which currently assist or even automate decision making, reproduce, and in the worst case reinforce, bias of the training data. The development of tools and techniques for certifying fairness of these models or describing their biased behavior is, therefore, critical. In this paper, we propose a perfectly parallel static analysis for certifying causal fairness of feed-forward neural networks used for classification of tabular data. When certification succeeds, our approach provides definite guarantees, otherwise, it describes and quantifies the biased behavior. We design the analysis to be sound, in practice also exact, and configurable in terms of scalability and precision, thereby enabling pay-as-you-go certification. We implement our approach in an open-source tool and demonstrate its effectiveness on models trained with popular datasets.

1 INTRODUCTION

Due to the tremendous advances in machine learning and the vast amounts of available data, software systems, and neural networks in particular, are of ever-increasing importance in our everyday decisions, whether by assisting them or by autonomously making them. We are already witnessing the wide adoption and societal impact of such software in criminal justice, health care, and social welfare, to name a few examples. It is, therefore, not far-fetched to imagine a future where most of the decision making is automated.

However, several studies have recently raised concerns about the fairness of such systems. For instance, consider a commercial recidivism-risk assessment algorithm that was found racially biased [Larson et al. 2016]. Similarly, a commercial algorithm that is widely used in the U.S. health care system falsely determined that Black patients were healthier than other equally sick patients by using health costs to represent health needs [Obermeyer et al. 2019]. There is also empirical evidence of gender bias in image searches, for instance, there are fewer results depicting women when searching for certain occupations, such as CEO [Kay et al. 2015]. Commercial facial recognition algorithms, which are increasingly used in law enforcement, are less effective for women and darker skin types [Buolamwini and Gebru 2018].

In other words, machine-learning software may reproduce, or even reinforce, bias that is directly or indirectly present in the training data. This awareness will certainly lead to regulations and strict audits in the future. It is, therefore, critical to develop tools and techniques for certifying fairness of neural networks and understanding the circumstances of their potentially biased behavior.

Causal Fairness. We make a step forward in meeting these needs by designing a static analysis framework for certifying causal fairness [Galhotra et al. 2017] of feed-forward neural networks used for classification tasks. Specifically, given a choice (e.g., driven by a causal model) of input features that are considered (directly or indirectly) sensitive to bias, a neural network is causally fair if the output classification is not affected by different values of the chosen features. Note that, unlike...
local robustness of neural networks, causal fairness is a global property, which is evaluated with respect to all inputs, instead of only those within a particular distance metric.

Of course, the most obvious approach to avoid such bias is to remove any sensitive feature from the training data, called fairness through unawareness [Grgić-Hlača et al. 2016]. However, this does not work for three main reasons. First, neural networks learn from latent variables (e.g., [Lum and Isaac 2016; Udeshi et al. 2018]). For instance, a credit-screening algorithm might not use race (or gender) as an explicit input but still be biased with respect to it, say, by using the ZIP code of applicants as proxy for race (or their first name as proxy for gender). Therefore, simply removing a sensitive feature does not necessarily free the training data or the corresponding neural network from bias. Second, the training data is only a relatively small sample of the entire input space, on portions of which the neural network might end up being inaccurate. For example, if women are underrepresented in the training data, a credit-screening algorithm is less likely to be accurate for them. Third, the information provided by a sensitive feature might be necessary, for instance, to introduce intended bias in a certain input region. Assume a credit-screening algorithm that should not discriminate with respect to age unless it is above a particular threshold. Above this age threshold, the higher the requested credit amount, the lower the chances of receiving it. In such cases, removing the sensitive feature is not even possible.

**Our Approach.** Verification of global neural-network properties, such as causal fairness, is still a long way from being practical (see Section 12). In this paper, we propose an approach that brings us closer to this aspiration. Our approach certifies causal fairness of neural networks used for classification of tabular data by employing a combination of a forward and a backward static analysis. On a high level, the forward pass aims to reduce the overall analysis effort. At its core, it divides the input space of the network into independent partitions. The backward analysis then attempts to certify fairness of the classification within each partition (in a perfectly parallel fashion) with respect to a chosen (set of) feature(s), which may be directly or indirectly sensitive, for instance, race or ZIP code. In the end, our approach reports for which regions of the input space the neural network is proved fair and for which there is bias. Note that we do not necessarily need to analyze the entire input space; our technique is also able to answer specific bias queries about a fraction of the input space, e.g., are Hispanics over 45 years old discriminated against with respect to gender?

The scalability-vs-precision tradeoff of our approach is configurable. Partitions that do not satisfy the given configuration are excluded from the analysis and may be resumed later, with a more flexible configuration. This enables usage scenarios in which our approach adapts to the available resources, e.g., time or CPUs, and is run incrementally. In other words, we designed a pay-as-you-go certification approach that the more resources it is given, the larger the region of the input space it is able to analyze.

**Related Work.** In the literature, related work on determining fairness of machine-learning models has focused on providing probabilistic guarantees [Bastani et al. 2019]. In contrast, our approach gives definite guarantees for those input partitions that satisfy the analysis configuration. Similarly to our approach, there is work that also aims to provide definite guarantees [Albarghouthi et al. 2017b] (although for different fairness criteria). However, it has been shown to scale only up to neural networks with two hidden neurons. Our approach is significantly more scalable since its design enables perfectly parallel fairness certification of each input partition.

**Contributions.** We make the following contributions:

1. We propose a perfectly parallel static analysis approach for certifying causal fairness of feed-forward neural networks used for classification of tabular data. If certification fails, our approach can describe and quantify the biased input space region(s).
(2) We show that our approach is sound and, in practice, exact for the analyzed regions of the input space.
(3) We discuss the configurable scalability-vs-precision tradeoff of our approach that enables pay-as-you-go certification.
(4) We implement our approach in an open-source tool called libra and evaluate it on neural networks trained with popular datasets. We show the effectiveness of our approach in detecting injected bias and answering bias queries. We also experiment with the precision and scalability of the analysis and discuss the tradeoffs.

2 OVERVIEW

In this section, we give an overview of our approach using a small constructed example, which is shown in Figure 1.

Example. The figure depicts a feed-forward neural network for credit approval. There are two inputs $x_{0,1}$ and $x_{0,2}$ (shown in purple). Input $x_{0,1}$ denotes the requested credit amount and $x_{0,2}$ denotes age. Both inputs have continuous values in the range $[0, 1]$. Output $x_{3,2}$ (shown in green) denotes that the credit request is approved, whereas $x_{3,1}$ (in red) denotes that it is denied. The neural network also consists of two hidden layers with two nodes each (in gray).

Now, let us assume that this neural network is trained to deny requests for large credit amounts from older people. Otherwise, the network does not discriminate with respect to age for small credit amounts. There is also no bias for younger people with respect to the requested credit. When choosing age as the sensitive input, our approach can certify fairness with respect to different age groups for small credit amounts. Our approach is also able to find (as well as quantify) bias with respect to age for large credit amounts. Note that this bias may be intended or accidental — our analysis does not aim to address this question.

Our approach does not require age to be an explicit input of the neural network. For example, $x_{0,2}$ could denote the ZIP code of credit applicants, and the network could still use it as proxy for age. That is, requests for large credit amounts are denied for a certain range of ZIP codes (where older people tend to live), yet there is no discrimination between ZIP codes for small credit amounts. When choosing the ZIP code as the sensitive input, our approach would again be able to detect bias with respect to it for large credit amounts.

Below, we present on a high level how our approach achieves these results.

Naïve Approach. In theory, the simplest way to certify causal fairness is to first analyze the neural network backwards starting from each output node, in our case $x_{3,1}$ and $x_{3,2}$. This allows us to determine the regions of the input space (i.e., age and requested credit amount) for which credit is approved and denied. For example, assume that we find that requests are denied for credit
amounts larger than 10 000 (i.e., $10 000 < x_{0,1}$) and age greater than 60 (i.e., $60 < x_{0,2}$), while they are approved for $x_{0,1} \leq 10 000$ and $60 < x_{0,2}$ or for $x_{0,2} \leq 60$.

The second step is to forget the value of the sensitive input (i.e., age) or, in other words, to project these regions over the credit amount. In our example, after projection we have that credit requests are denied for $10 000 < x_{0,1}$ and approved for any value of $x_{0,1}$. A non-empty intersection between the projected input regions indicates bias with respect to the sensitive input. In our example, the intersection is non-empty for $10 000 < x_{0,1}$: there exist people that differ in age but request the same credit amount (greater than 10 000), some of whom receive the credit while others do not.

This approach, however, is not practical. Specifically, neural networks with ReLU activation functions (see Section 3 for more details, other activation functions are discussed in Section 9), each hidden node effectively represents a disjunction between two activation statuses (active and inactive). In our example, there are $2^4$ possible activation patterns for the 4 hidden nodes. To retain maximum precision, the analysis would have to explore all of them, which does not scale in practice.

Our Approach. Our analysis is based on the observation that there might exist many activation patterns that do not correspond to a region of the input space [Hanin and Rolnick 2019]. Such patterns can, therefore, be ignored during the analysis. We push this idea further by defining abstract activation patterns, which fix the activation status of only certain nodes and thus represent sets of (concrete) activation patterns. Typically, a relatively small number of abstract activation patterns is sufficient for covering the entire input space, without necessarily representing and exploring all possible concrete patterns.

Identifying those patterns that definitely correspond to a region of the input space is only possible with a forward analysis. Hence, we combine a forward pre-analysis with a backward analysis. The pre-analysis partitions the input space into independent partitions corresponding to abstract activation patterns. Then, the backward analysis tries to prove fairness of the neural network for each such partition.

More specifically, we set an upper bound $U$ on the number of tolerated disjunctions (i.e., on the number of nodes with an unknown activation status) per abstract activation pattern. Our forward pre-analysis uses a cheap abstract domain (e.g., the boxes domain [Cousot and Cousot 1976]) to iteratively partition the input space along the non-sensitive input dimensions to obtain fair input partitions (i.e., boxes). Each partition satisfies one of the following conditions: (a) its classification is already fair because only one network output is reachable for all inputs in the region, (b) it has an abstract activation pattern with at most $U$ unknown nodes, or (c) it needs to be partitioned further. We call partitions that satisfy condition (b) feasible.

In our example, let $U = 2$. At first, the analysis considers the entire input space, that is, $x_{0,1} : [0, 1]$ (credit amount) and $x_{0,2} : [0, 1]$ (age). (Note that we could also specify a part of the input space for analysis.) The abstract activation pattern corresponding to this initial partition $I$ is $\epsilon$ (i.e., no hidden nodes have fixed activation status) and, thus, the number of disjunctions would be 4, which is greater than $U$. Therefore, I needs to be divided into $I_1 (x_{0,1} : [0, 0.5], x_{0,2} : [0, 1])$ and $I_2 (x_{0,1} : [0.5, 1], x_{0,2} : [0, 1])$. Observe that the input space is not split with respect to $x_{0,2}$, which is the sensitive input. Now, $I_1$ is feasible since its abstract activation pattern is $x_{1,2}\neg x_{2,1}$ (i.e., 3 nodes are always active), while $I_2$ must be divided further since its abstract activation pattern is $\epsilon$.

To control the number of partitions, we impose a lower bound $L$ on the size of each of their dimensions. Partitions that require a dimension of a smaller size are excluded. In other words, they are not considered until more analysis budget becomes available, that is, a larger $U$ or a smaller $L$.

In our example, let $L = 0.25$. The forward pre-analysis further divides $I_2$ into $I_{2,1} (x_{0,1} : [0.5, 0.75], x_{0,2} : [0, 1])$ and $I_{2,2} (x_{0,1} : [0.75, 1], x_{0,2} : [0, 1])$. Now, $I_{2,1}$ is feasible, with abstract pattern $\neg x_{1,2} x_{2,1}$, while $I_{2,2}$ is not. However, $I_{2,2}$ may not be split further because the size of the only
non-sensitive dimension \( x_{0,1} \) has already reached the lower bound \( L \). As a result, \( I_{2,2} \) is excluded, and only the remaining 75% of the input space is considered for analysis.

Next, feasible input partitions (within bounds \( L \) and \( U \)) are grouped by abstract activation patterns. In our example, the pattern corresponding to \( I_1 \), namely \( x_{1,2}x_{2,1} \), is subsumed by the (more abstract) pattern of \( I_2 \), namely \( x_{1,2}x_{2,1} \). Consequently, we group \( I_1 \) and \( I_2 \) under pattern \( x_{1,2}x_{2,1} \).

The backward analysis is then run in parallel for each representative abstract activation pattern, in our example \( x_{1,2}x_{2,1} \). This analysis determines the region of the input space (within a given partition group) for which each output of the neural network is returned, e.g., credit is approved for \( c_1 \leq x_{0,1} \leq c_2 \) and \( a_1 \leq x_{0,2} \leq a_2 \). To achieve this, the analysis uses an expensive abstract domain, for instance, disjunctive or powerset polyhedra \([Cousot and Cousot 1979; Cousot and Halbwachs 1978]\), and leverages abstract activation patterns to avoid disjunctions. For instance, pattern \( x_{1,2}x_{2,1} \) only requires reasoning about two disjunctions from the remaining hidden nodes \( x_{1,1} \) and \( x_{2,2} \).

Finally, fairness is checked for each partition in the same way that it is done by the naive approach for the entire input space. In our example, we prove that the classification within \( I_1 \) is fair and determine that within \( I_2 \), the classification is biased. Concretely, our approach determines that bias occurs for \( 0.54 \leq x_{0,1} \leq 0.75 \), which corresponds to 21% of the entire input space (assuming a uniform probability distribution). In other words, the network returns different outputs for people that request the same credit in the above range but differ in age. Recall that partition \( I_{2,2} \), where \( 0.75 \leq x_{0,1} \leq 1 \), was excluded from analysis, and therefore, we cannot draw any conclusions about whether there is any bias for people requesting credit in this range.

Note that bias may also be quantified according to a probability distribution of the input space. In particular, it might be that credit requests in the range \( 0.54 \leq x_{0,1} \leq 0.75 \) are more (resp. less) common in practice. Given their probability distribution, our analysis computes a tailored percentage of bias, which in this case would be greater (resp. less) than 21%.

3 FEED-FORWARD DEEP NEURAL NETWORKS

Formally, a feed-forward deep neural network consists of an input layer (\( l_0 \)), an output layer (\( l_N \)), and a number of hidden layers (\( l_1, \ldots, l_{N−1} \)) in between. Each layer \( l_i \) contains \(|l_i| \) nodes and, with the exception of the input layer, is associated to a \( |l_i| \times |l_{i−1}| \)-matrix \( W_i \) of weight coefficients and a vector \( b_i \) of \(|l_i| \) bias coefficients. In the following, we use \( X \) to denote the set of all nodes, \( X_i \) to denote the set of nodes of the \( i \)th layer, and \( x_{i,j} \) to denote the \( j \)th node of the \( i \)th layer of a neural network. We focus here on neural networks used for classification tasks. Thus, \( |l_N| \) is the number of target classes (e.g., 2 classes in Figure \( 1 \)).

The value of the input nodes is given by the input data: continuous data is represented by one input node (e.g., \( x_{0,1} \) or \( x_{0,2} \) in Figure \( 1 \)), while categorical data is represented by multiple input nodes via one-hot encoding. In the following, we use \( K \) to denote the subset of input nodes considered (directly or indirectly) sensitive to bias (e.g., \( x_{0,2} \) in Figure \( 1 \)) and \( K_e \) to denote the input nodes not deemed sensitive to bias.

The value of each hidden and output node \( x_{i,j} \) is computed by an activation function \( f \) applied to a linear combination of the values of all nodes in the preceding layer \([Goodfellow et al. 2016]\), i.e., \( x_{i,j} = f \left( \sum_{k}^{\left|l_{i−1}\right|} w^i_{j,k} \cdot x_{i−1,k} + b_{i,j} \right) \), where \( w^i_{j,k} \) and \( b_{i,j} \) are weight and bias coefficients in \( W_i \) and \( B_i \), respectively. In a fully-connected neural network, all \( w^i_{j,k} \) are non-zero. Weights and biases are adjusted during the training phase of the neural network. In what follows, we focus on already trained neural networks, which we call neural-network models.

Nowadays, the most commonly used activation for hidden nodes is the Rectified Linear Unit (ReLU) \([Nair and Hinton 2010]\): \( \text{ReLU}(x) = \max(x, 0) \). In this case, the activation used for output
nodes is the identity function. The output values are then normalized into a probability distribution on the target classes [Goodfellow et al. 2016]. We discuss other activation functions in Section 9.

4 TRACE SEMANTICS

Our approach expresses neural-network models as programs. These programs consist of assignments for computing the activation value of each node (e.g., \( x_{1,1} = -0.31 \times x_{0,1} + 0.99 \times x_{0,2} - 0.63 \) in Figure 1) and implementations of activation functions (e.g., \( \text{if-statement for ReLUs} \)). As is standard practice in static program analysis, we define a semantics for these programs and use it to prove soundness of our approach.

The semantics of a neural-network model is a mathematical characterization of its behavior when executed for all possible input data. We model the operational semantics of a feed-forward neural-network model \( M \) as a transition system \( \langle \Sigma, \tau \rangle \), where \( \Sigma \) is a (potentially infinite) set of states and the acyclic transition relation \( \tau \subseteq \Sigma \times \Sigma \) describes the possible transitions between states [Cousot and Cousot 1977, 1979]. Properties of neural-network models are properties of their semantics tailored to reasoning about this property.

In the rest of the paper, we write \( \langle 1 \rangle \) to denote the trace semantics of a neural-network model \( M \).

The trace semantics fully describes the behavior of \( M \). However, reasoning about a particular property of \( M \) does not need all this information and, in fact, is facilitated by the design of a semantics that abstracts away from irrelevant details about \( M \)’s behavior. In the following sections, we formally define our property of interest, causal fairness, and systematically derive, using abstract interpretation [Cousot and Cousot 1977], a semantics tailored to reasoning about this property.

5 CAUSAL FAIRNESS

A property is specified by its extension, that is, by the set of elements having such a property [Cousot and Cousot 1977, 1979]. Properties of neural-network models are properties of their semantics. Thus, properties of network models with trace semantics in \( \mathcal{P}(\Sigma^+) \) are sets of sets of
traces in $\mathcal{P}(\mathcal{P}(\Sigma^+))$. In particular, the set of neural-network properties forms a complete boolean lattice $\langle \mathcal{P}(\mathcal{P}(\Sigma^+)), \subseteq, \cup, \cap, \emptyset, \mathcal{P}(\Sigma^+) \rangle$ for subset inclusion, that is, logical implication. The strongest property is the standard collecting semantics $\Lambda \triangleq \{ Y \}$.

Let $\langle M \rangle$ denote the collecting semantics of a particular neural-network model $M$. Then, model $M$ satisfies a given property $\mathcal{H}$ if and only if its collecting semantics is a subset of $\mathcal{H}$:

$$M \models \mathcal{H} \iff \langle M \rangle \subseteq \mathcal{H} \quad (3)$$

Here, we consider the property of causal fairness, which expresses that the classification determined by a network model does not depend on sensitive input data. In particular, the property might interest the classification of all or just a fraction of the input space.

More formally, let $\mathbb{V}$ be the set of all possible value choices for all sensitive input nodes in $K$, e.g., for $K = \{ x_{0,i}, x_{0,j} \}$ one-hot encoding, say, gender information, $\mathbb{V} = \{ \{1, 0\}, \{0, 1\} \}$; for $K = \{ x_{0,k} \}$ encoding continuous data, say, in the range $[0, 1]$, a possibility is $\mathbb{V} = \{[0, 0.25], [0.25, 0.75], [0.75, 1]\}$. In the following, given a trace $\sigma \in \mathcal{P}(\Sigma^+)$, we write $\sigma_0$ and $\sigma_\omega$ to denote its initial and final state, respectively. We also write $\sigma_0 =_{K} \sigma'_0$ to indicate that the states $\sigma_0$ and $\sigma'_0$ agree on all values of all non-sensitive input nodes, and $\sigma_\omega =_{K} \sigma'_\omega$ to indicate that $\sigma$ and $\sigma'$ have the same outcome $O \in \mathbb{O}$. We can now formally define when the sensitive input nodes in $K$ are unused with respect to a set of traces $T \in \mathcal{P}(\Sigma^+)$ [Urban and Müller 2018]. For one-hot encoded sensitive inputs\(^1\) we have

$$\text{UNUSED}_K(T) \triangleq \forall \sigma \in T, \forall V \subseteq \mathbb{V} : \sigma_0(V) \neq V \implies \exists \sigma' \in T : \sigma_0 =_{K} \sigma'_0 \land \sigma_\omega =_{K} \sigma'_\omega,$$  

(4)

where $\sigma_0(V) \triangleq \{ \sigma_0(x) \mid x \in K \}$ is the image of $K$ under $\sigma_0$. Intuitively, the sensitive input nodes in $K$ are unused if any possible outcome in $T$ (i.e., any outcome $\sigma_\omega$ of any trace $\sigma$ in $T$) is possible from all possible value choices for $K$ (i.e., there exists a trace $\sigma'$ in $T$ for each value choice for $K$ with the same outcome as $\sigma$). That is, each outcome is independent of the value choice for $K$.

**Example 5.1.** Let us consider again our example in Figure 1. We write $\langle c, a \rangle \sim o$ for a trace starting in a state with $x_{0,1} = c$ and $x_{0,2} = a$ and ending in a state where $o$ is the node with the highest value (i.e., the output class). The sensitive input $x_{0,2}$ (age) is unused in $T = \{ \langle 0.5, a \rangle \sim o \mid 0 \leq a \leq 1 \}$. It is instead used in $T' = \{ \langle 0.75, a \rangle \sim o \mid 0 \leq a < 0.5 \} \cup \{ \langle 0.75, a \rangle \sim o \mid 0.5 \leq a \leq 1 \}$.

The causal-fairness property $\mathcal{F}_K$ can now be defined as $\mathcal{F}_K \triangleq \{ \langle [M] \rangle \mid \text{UNUSED}_K([M]) \}$, that is, as the set of all neural-network models (or rather, their semantics) that do not use the values of the sensitive input nodes for classification. In practice, the property might interest just a fraction of the input space, i.e., we define

$$\mathcal{F}_K[Y] \triangleq \{ \langle [M]^Y \rangle \mid \text{UNUSED}_K([M]^Y) \}, \quad (5)$$

where $Y \in \mathcal{P}(\Sigma)$ is a set of initial states of interest and the restriction $T^Y \triangleq \{ \sigma \in T \mid \sigma_0 \in Y \}$ only contains traces of $T \in \mathcal{P}(\Sigma^+)$ that start with a state in $Y$. Similarly, in the rest of the paper, we write $S^Y \triangleq \{ T^Y \mid T \in S \}$ for the set of sets of traces restricted to initial states in $Y$. Thus, from Equation 3, we have the following:

**Theorem 5.2.** $M \models \mathcal{F}_K[Y] \iff \langle M \rangle^Y \subseteq \mathcal{F}_K[Y]$.

**Proof.** The proof follows trivially from Equation 3 and the definition of $\mathcal{F}_K[Y]$ (cf. Equation 5) and $\langle M \rangle^Y$.

\(^1\)For continuous sensitive inputs, we can replace $\sigma_0(K) \neq V$ (resp. $\sigma_0(K) = V$) with $\sigma_0(K) \notin V$ (resp. $\sigma_0(K) \subseteq V$).
6 DEPENDENCY SEMANTICS

We now use abstract interpretation to systematically derive, by successive abstractions of the collecting semantics \( \Lambda \), a sound and complete semantics \( \Lambda_\sim \) that contains only and exactly the information needed to reason about \( \mathcal{F}_K[Y] \).

6.1 Outcome Semantics

Let \( T_Z \overset{\text{def}}{=} \{ \sigma \in T \mid \sigma_o \in Z \} \) be the set of traces of \( T \in \mathcal{P}(\Sigma^+) \) that end with a state in \( Z \in \mathcal{P}(\Sigma) \). As before, we write \( S_Z \overset{\text{def}}{=} \{ T_Z \mid T \in S \} \) for the set of sets of traces restricted to final states in \( Z \). From the definition of \( \mathcal{F}_K[Y] \) (and in particular, from the definition of \( \text{UNUSED}_K \), cf. Equation 4), we have:

**Lemma 6.1.** \( \langle M \rangle^Y \subseteq \mathcal{F}_K[Y] \Leftrightarrow \forall O \in \mathcal{O} : \langle M \rangle^Y_O \subseteq \mathcal{F}_K[Y] \)

**Proof.** Let \( \langle M \rangle^Y \subseteq \mathcal{F}_K[Y] \). From the definition of \( \langle M \rangle^Y \) (cf. Equation 2), we have that \( [M]^Y \subseteq \mathcal{F}_K[Y] \). Thus, from the definition of \( \mathcal{F}_K[Y] \) (cf. Equation 5), we have \( \text{UNUSED}_K([M]^Y) \). Now, from the definition of \( \text{UNUSED}_K \) (cf. Equation 4), we equivalently have \( \forall O \in \mathcal{O} : \text{UNUSED}_K([M]^Y_O) \). Thus, we can conclude that \( \forall O \in \mathcal{O} : \langle M \rangle^Y_O \subseteq \mathcal{F}_K[Y] \). \qed

In particular, this means that in order to determine whether a neural-network model \( M \) satisfies causal fairness, we can independently verify, for each of its possible target classes \( O \in \mathcal{O} \), that the values of its sensitive input nodes are unused.

We use this insight to abstract the collecting semantics \( \Lambda \) by *partitioning*. More specifically, let \( \bullet \overset{\text{def}}{=} \{ \Sigma^+_O \mid O \in \mathcal{O} \} \) be a trace partition with respect to outcome. We have the following Galois connection

\[
\langle \mathcal{P}(\mathcal{P}(\Sigma^+)) , \subseteq \rangle \xleftarrow{\alpha_\sim} \langle \mathcal{P}(\mathcal{P}(\Sigma^+)) , \subseteq \rangle
\]

where \( \alpha_\sim(S) \overset{\text{def}}{=} \{ T_0 \mid T \in S \land O \in \mathcal{O} \} \). The order \( \subseteq \) is the pointwise ordering between sets of traces with the same outcome, i.e., \( A \subseteq B \overset{\text{def}}{=} \bigwedge_{O \in \mathcal{O}} A_O \subseteq B_O \), where \( S_Z \) denotes the only non-empty set of traces in \( S_Z \). We can now define the *outcome semantics* \( \Lambda_\sim \in \mathcal{P}(\mathcal{P}(\Sigma^+)) \) by abstraction of \( \Lambda \):

\[
\Lambda_\sim \overset{\text{def}}{=} \alpha_\sim(\Lambda) = \{ Y_O \mid O \in \mathcal{O} \}
\]

(7)

In the rest of the paper, we write \( \langle M \rangle_\sim \) to denote the outcome semantics of a particular neural-network model \( M \).

6.2 Dependency Semantics

We observe that, to reason about causal fairness, we do not need to consider all intermediate computations between the initial and final states of a trace. Thus, we can further abstract the outcome semantics into a set of dependencies between initial states and outcomes of traces.

To this end, we define the following Galois connection\(^2\)

\[
\langle \mathcal{P}(\mathcal{P}(\Sigma^+)) , \subseteq \rangle \xleftarrow{\alpha_\sim} \langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma)) , \subseteq \rangle
\]

(8)

where \( \alpha_\sim(S) \overset{\text{def}}{=} \{ \{ \sigma_0, \sigma_o \} \mid \sigma \in T \mid T \in S \} \) [Urban and Müller 2018] abstracts away all intermediate states of any trace. We finally derive the *dependency semantics* \( \Lambda_\sim \in \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma)) \):

\[
\Lambda_\sim \overset{\text{def}}{=} \alpha_\sim(\Lambda) = \{ \{ \sigma_0, \sigma_o \} \mid \sigma \in Y_O \mid O \in \mathcal{O} \}
\]

(9)

In the following, let \( \langle M \rangle_\sim \) denote the dependency semantics of a particular network model \( M \).

\(^2\)Note that here and in the following, for convenience, we abuse notation and reuse the order symbol \( \subseteq \) defined over sets of sets of traces, instead of its abstraction, defined over sets of sets of pairs of states.
Let \( R^Y \overset{\text{def}}{=} \{ \langle s, \_ \rangle \in R \mid s \in Y \} \) restrict a set of pairs of states to pairs whose first element is in \( Y \) and, similarly, let \( S^Y \overset{\text{def}}{=} \{ R^Y \mid R \in S \} \) restrict a set of sets of pairs of states to first elements in \( Y \). The next result shows that \( \Lambda_\rightarrow \) is sound and complete for proving causal fairness:

**Theorem 6.2.** \( M \models \mathcal{F}_K[Y] \iff \langle \mathcal{M} \rangle^Y_{\rightarrow} \subseteq \alpha_\rightarrow(\alpha_\rightarrow(\mathcal{F}_K[Y])) \)

**Proof.** Let \( M \models \mathcal{F}_K[Y] \). From Theorem 5.2, we have that \( \langle \mathcal{M} \rangle^Y \subseteq \mathcal{F}_K[Y] \).Thus, from the Galois connections in Equation 6 and 8, we have \( \alpha_\rightarrow(\alpha_\rightarrow(\mathcal{F}_K[Y])) \subseteq \alpha_\rightarrow(\alpha_\rightarrow(\mathcal{F}_K[Y])) \). From the definition of \( \langle \mathcal{M} \rangle^Y_{\rightarrow} \) (cf. Equation 9), we can then conclude that \( \langle \mathcal{M} \rangle^Y_{\rightarrow} \subseteq \alpha_\rightarrow(\alpha_\rightarrow(\mathcal{F}_K[Y])) \).

**Corollary 6.3.** \( M \models \mathcal{F}_K[Y] \iff \langle \mathcal{M} \rangle^Y_{\rightarrow} \subseteq \alpha_\rightarrow(\mathcal{F}_K[Y]) \)

**Proof.** The proofs follows trivially from the definition of \( \subseteq \) (cf. Equation 6 and 8) and Lemma 6.1.

Furthermore, we observe that partitioning with respect to outcome induces a partition of the space of values of the input nodes used for classification. For instance, partitioning \( T^i \) in Example 5.1 induces a partition on the values of (the indeed used node) \( x_{0,2} \). Thus, we can equivalently verify whether \( \langle \mathcal{M} \rangle^Y_{\rightarrow} \subseteq \alpha_\rightarrow(\mathcal{F}_K[Y]) \) by checking if the dependency semantics \( \langle \mathcal{M} \rangle^Y \) induces a partition of \( Y_{\mathcal{K}} \). Let \( R_0 \overset{\text{def}}{=} \{ s \mid \langle s, \_ \rangle \in R \} \) (resp. \( R_\rightarrow \overset{\text{def}}{=} \{ s \mid \langle \_ , s \rangle \in R \} \) be the selection of the first (resp. last) element from each pair in a set of pairs of states. We formalize this observation below.

**Lemma 6.4.** \( M \models \mathcal{F}_K[Y] \iff \forall A, B \in \langle \mathcal{M} \rangle^Y_{\rightarrow} : (A_{\omega_0} \neq B_{\omega_0} \Rightarrow A_{0|\mathcal{K}} \cap B_{0|\mathcal{K}} = \emptyset) \)

**Proof.** Let \( M \models \mathcal{F}_K[Y] \). From Corollary 6.3, we have that \( \langle \mathcal{M} \rangle^Y_{\rightarrow} \subseteq \alpha_\rightarrow(\mathcal{F}_K[Y]) \). Thus, from the definition of \( \langle \mathcal{M} \rangle^Y_{\rightarrow} \) (cf. Equation 9), we have \( \forall O \in \mathcal{O} : \alpha_\rightarrow(\langle \mathcal{M} \rangle^Y_0) \in \alpha_\rightarrow(\mathcal{F}_K[Y]) \). In particular, from the definition of \( \alpha_\rightarrow \) and \( \mathcal{F}_K[Y] \) (cf. Equation 5), we have that \( \text{UNUSED}_K(\langle \mathcal{M} \rangle^Y_0) \) for each \( O \in \mathcal{O} \). From the definition of \( \text{UNUSED}_K \) (cf. Equation 4), for each pair of non-empty \( \langle \mathcal{M} \rangle^Y_0 \) and \( \langle \mathcal{M} \rangle^Y_2 \) for different \( O_1, O_2 \in \mathcal{O} \) (the case in which one or both are empty is trivial), it must necessarily be the value of the non-sensitive input nodes in \( \mathcal{K} \) that causes the different outcome \( O_1 \) or \( O_2 \). We can thus conclude that \( \forall A, B \in \langle \mathcal{M} \rangle^Y_{\rightarrow} : (A_{\omega_0} \neq B_{\omega_0} \Rightarrow A_{0|\mathcal{K}} \cap B_{0|\mathcal{K}} = \emptyset) \).

## 7 NAÏVE CAUSAL-FAIRNESS ANALYSIS

In this section, we present a first static analysis for causal fairness that computes a sound over-approximation \( \Lambda_\rightarrow \) of the dependency semantics \( \Lambda_\rightarrow \), i.e., \( \Lambda_\rightarrow \subseteq \Lambda_\rightarrow \). This analysis corresponds to the naïve approach we discussed in Section 2. While it is too naïve to be practical, it is still useful for building upon later in the paper.

For simplicity, we consider ReLU activation functions. (We discuss extensions to other activation functions in Section 9.) The naïve static analysis is described in Algorithm 1. It takes as input (cf. Line 14) a neural-network model \( M \), a set of sensitive input nodes \( K \) of \( M \), a (representation of a) set of initial states of interest \( Y \), and an abstract domain \( A \) to be used for the analysis. The analysis proceeds backwards for each outcome (i.e., each target class \( x_{n, j} \)) of \( M \) (cf. Line 17) in order to determine an over-approximation of the initial states that satisfy \( Y \) and lead to \( x_{n, j} \) (cf. Line 18).

More specifically, the transfer function \( \text{OUTCOME}_A[x] \) (cf. Line 2) modifies a given abstract-domain element to assume the given outcome \( x \), that is, to assume that max \( X_n = x \). The transfer functions \( \text{RELU}_A \left[ x_{i,j} \right] \) and \( \text{ASSIGN}_A \left[ x_{i,j} \right] \) (cf. Line 5) respectively consider a ReLU operation and replace \( x_{i,j} \) with the corresponding linear combination of nodes in the preceding layer (see Section 3).

Finally, the analysis checks whether the computed over-approximations satisfy causal fairness with respect to \( K \) (cf. Line 19). In particular, it checks whether they induce a partition of \( Y_{\mathcal{K}} \) as observed for Lemma 6.4 (cf. Lines 7-13). If so, we have proved that \( M \) satisfies causal fairness. If
Algorithm 1: A Naïve Backward Analysis

1: function BACKWARD(M, A, x)
2: \( a \leftarrow \text{OUTCOME}_A[x] \) (NEW_A)
3: for \( i \leftarrow N - 1 \) down to 0 do
4: \( j \leftarrow |L_i| \) down to 0 do
5: \( a \leftarrow \text{ASSIGN}_A[x_{i,j}] (\tilde{\text{RELU}}_A[x_{i,j}] a) \)
6: return a
7: function CHECK(O)
8: \( B \leftarrow \emptyset \)
9: for all \( o_1, a_1 \in O \) do
10: \( \text{for all } o_2 \neq o_1, a_2 \in O \) do
11: if \( a_1 \sqcap a_2 \neq \bot \) then
12: \( B \leftarrow B \cup \{ a_1 \sqcap a_2 \} \)
13: return B
14: function ANALYZE(M, K, Y, A)
15: \( O \leftarrow \emptyset \)
16: for \( j \leftarrow 0 \) up to \(|L_N|\) do \( \triangleright \) perfectly parallelizable
17: \( a \leftarrow \text{BACKWARD}(M, A, x_{N,j}) \)
18: \( O \leftarrow O \cup \{ x_{N,j} \mapsto (\text{ASSUME}_A[Y] a)_{\tilde{K}} \} \)
19: return B = 0, B \( \triangleright \) fair: \( B = \emptyset \), maybe biased: \( B \neq \emptyset \)

not, the analysis returns a set \( B \) of abstract-domain elements over-approximating the input regions in which bias might occur.

**Theorem 7.1.** If \( \text{ANALYZE}(M, K, Y, A) \) of Algorithm 1 returns \( \text{true} \), \( \emptyset \) then \( M \) satisfies \( \mathcal{F}_K[Y] \).

**Proof (Sketch).** \( \text{ANALYZE}(M, K, Y, A) \) in Algorithm 1 computes an over-approximation \( a \) of the regions of the input space that yield each target class \( x_{N,j} \) (cf. Line 17). Thus, it actually computes an over-approximation \( \langle M \rangle^a \) of the dependency semantics \( \langle M \rangle^Y \), i.e., \( \langle M \rangle^Y \subseteq \langle M \rangle^a \). Thus, if \( \langle M \rangle^a \) satisfies \( \mathcal{F}_K[Y] \), i.e., \( \forall A, B \in \langle M \rangle^a: (A_0 \neq B_0 \Rightarrow A_0_{\tilde{K}} \cap B_0_{\tilde{K}} = \emptyset) \) (according to Lemma 6.4, cf. Line 19), then by transitivity we can conclude that also \( \langle M \rangle^Y \) necessarily satisfies \( \mathcal{F}_K[Y] \). \( \square \)

In the analysis implementation, there is a tradeoff between performance and precision, which is reflected in the choice of abstract domain \( A \) and its transfer functions. Unfortunately, existing numerical abstract domains that are less expressive than polyhedra [Cousot and Halbwachs 1978] would make for a rather fast but too imprecise analysis. This is because they are not able to precisely handle constraints like \( \max_{X_N} = x \), which are introduced by \( \text{OUTCOME}_A[x] \) to partition with respect to outcome.

Furthermore, even polyhedra would not be precise enough in general. Indeed, each \( \tilde{\text{RELU}}_A[x_{i,j}] \) would over-approximate what effectively is a conditional branch. Let \( |M| \overset{\text{def}}{=} |L_1| + \cdots + |L_{n-1}| \) denote the number of hidden nodes (i.e., the number of RELUs) in a model \( M \). On the other side of the spectrum, one could use a disjunctive completion [Cousot and Cousot 1979] of polyhedra, thus keeping a separate polyhedron for each branch of a ReLU. This would yield a precise (in fact, exact) but extremely slow analysis: even with parallelization (cf. Lines 16), each of the \( |L_N| \) processes would have to effectively explore \( 2^{|M|} \) paths!
In the rest of the paper, we improve on this naïve analysis and show how far we can go all the while remaining exact by using disjunctive polyhedra.

8 PARALLEL SEMANTICS

We first have to take a step back and return to reasoning at the concrete-semantics level. At the end of Section 6, we observed that the dependency semantics of a neural-network model \( M \) satisfying \( \mathcal{F}_{K}[Y] \) effectively induces a partition of \( Y \). We call this input partition fair.

More formally, given a set \( Y \) of initial states of interest, we say that an input partition \( I \) of \( Y \) is fair if all value choices \( \forall \) for the sensitive input nodes \( K \) of \( M \) are possible in all elements of the partitions: \( \forall I \in \mathbb{I}, V \in \mathbb{V} : \exists s \in I : s(K) = V \). For instance, \( I = \{ T_0, T'_0 \} \), with \( T \) and \( T' \) in Example 5.1 is a fair input partition of \( Y = \{ s \mid s(x_{0,1}) = 0.5 \lor s(x_{0,1}) = 0.75 \} \).

Given a fair input partition \( I \) of \( Y \), the following result shows that we can verify whether a model \( M \) satisfies \( \mathcal{F}_{K}[Y] \) for each element \( I \) of \( \mathbb{I} \), independently.

**Lemma 8.1.** \( M \models \mathcal{F}_{K}[Y] \iff \forall I \in \mathbb{I} : \forall A, B \in \langle M \rangle \downarrow \alpha_I : (A_{\alpha_I} \neq B_{\alpha_I} \Rightarrow A_{\alpha_I}[K] \cap B_{\alpha_I}[K] = \emptyset) \)

**Proof.** The proof follows trivially from Lemma 6.4 and the fact that \( \mathbb{I} \) is a fair partition. \( \Box \)

We use this new insight to further abstract the dependency semantics \( \Lambda_{\alpha_I} \). We have the following Galois connection

\[
\langle \mathcal{P} (\mathcal{P} (\Sigma \times \Sigma)), \mathcal{G} \rangle \xrightarrow{\alpha_{\alpha_I}} \langle \mathcal{P} (\mathcal{P} (\Sigma \times \Sigma)), \mathcal{G} \rangle, \tag{10}
\]

where \( \alpha_{\alpha_I}(S) \overset{\text{def}}{=} \{ R^I \mid R \in S \land I \in \mathbb{I} \} \). Here the order \( \mathcal{G} \) is the pointwise ordering between sets of pairs of states restricted to first elements in the same \( I \in \mathbb{I} \), i.e., \( A \mathcal{G} B \overset{\text{def}}{=} \bigwedge_{I \in \mathbb{I}} A^I \subseteq B^I \), where \( S^I \) denotes the only non-empty set of pairs in \( S^I \). We can now derive the parallel semantics \( \Pi_{\alpha_I} \in \mathcal{P} (\mathcal{P} (\Sigma \times \Sigma)) \):

\[
\Pi_{\alpha_I} \overset{\text{def}}{=} \alpha_I(\Lambda_{\alpha_I}) = \{ \{ \sigma_0, \sigma_0 \} \mid I \in \mathbb{I} \land O \in \mathcal{O} \} \tag{11}
\]

In fact, we derive a hierarchy of semantics, as depicted in Figure 2. We write \( \langle M \rangle \downarrow \alpha_I \) to denote the parallel semantics of a particular neural-network model \( M \). It remains to show soundness and completeness for \( \Pi_{\alpha_I} \).

**Theorem 8.2.** \( M \models \mathcal{F}_{K}[Y] \iff \langle M \rangle \downarrow \alpha_I \subseteq \alpha_I(\alpha_I(\mathcal{F}_{K}[Y])) \)

**Proof.** Let \( M \models \mathcal{F}_{K}[Y] \). From Theorem 6.2, we have that \( \langle M \rangle \downarrow \alpha_I \subseteq \alpha_I(\alpha_I(\mathcal{F}_{K}[Y])) \). Thus, from the Galois connections in Equation 10, we have \( \alpha_I(\langle M \rangle \downarrow \alpha_I) \subseteq \alpha_I(\alpha_I(\alpha_I(\mathcal{F}_{K}[Y]))) \). From the definition of \( \langle M \rangle \downarrow \alpha_I \) (cf. Equation 11), we can then conclude that \( \langle M \rangle \downarrow \alpha_I \subseteq \alpha_I(\alpha_I(\mathcal{F}_{K}[Y])) \). \( \Box \)

**Corollary 8.3.** \( M \models \mathcal{F}_{K}[Y] \iff \langle M \rangle \downarrow \alpha_I \subseteq \alpha_I(\mathcal{F}_{K}[Y]) \)
The transfer function $\rightarrow (\text{Lines 28-38}).$ The forward pre-analysis uses an abstract domain $A$ for a value of hidden node $1:1$ Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang.

ReLU fixes fewer groups partitions whose abstract activation patterns fix more abstract activation patterns that are subsumed by other (more) abstract patterns. In other words, it abstract activation patterns to feasible partitions (cf. Line 22). The insertion takes care of merging $I$ the backward analysis for $p$ if abstract activation pattern $I$ is no need for a backward analysis; $I$ activation pattern $I$ are to be explored by the backward analysis in each $I$ of a neural-network model $I$. The design of our analysis builds on this key observation.

Relative small number of abstract activation patterns is sufficient for covering the entire input space an abstract activation pattern have an unknown (i.e., not fixed) activation status. Typically, $I$ thus, represents a set of activation patterns.

ReLU activation status of every $I$ from its input space.

2 analysis has to explore. Indeed, not all of the $Y$ of initial states $M$ neural-network model $A$ backward analysis uses an abstract domain $U$ (cf. Line 13) on the number of tolerated $\omega$ $R$ $A$ and builds partition $\mathbb{I}$, while the backward analysis uses an abstract domain $A_2$ and performs the actual causal-fairness analysis of a neural-network model $M$ with respect to its sensitive input nodes $K$ and a (representation of a) set of initial states $Y$ (cf. Line 13).

More specifically, the forward pre-analysis bounds the number of paths that the backward analysis has to explore. Indeed, not all of the $2^{|M|}$ paths of a model $M$ are necessarily viable starting from its input space.

In the rest of this section, we represent each path by an activation pattern, which determines the activation status of every $\text{ReLU}$ operation in $M$. More precisely, an activation pattern is a sequence of flags. Each flag $p_{i,j}$ represents the activation status of the $\text{ReLU}$ operation used to compute the value of hidden node $x_{i,j}$. If $p_{i,j}$ is $x_{i,j}$, the $\text{ReLU}$ is always active, otherwise the $\text{ReLU}$ is always inactive and $p_{i,j}$ is $\overline{x}_{i,j}$.

An abstract activation pattern gives the activation status of only a subset of the $\text{ReLU}$s of $M$, and thus, represents a set of activation patterns. $\text{ReLU}$s whose corresponding flag does not appear in an abstract activation pattern have an unknown (i.e., not fixed) activation status. Typically, only a relatively small number of abstract activation patterns is sufficient for covering the entire input space of a neural-network model. The design of our analysis builds on this key observation.

We set an analysis budget by providing an upper bound $U$ (cf. Line 13) on the number of tolerated $\text{ReLU}$s with an unknown activation status for each element $I$ of $\mathbb{I}$, i.e., on the number of paths that are to be explored by the backward analysis in each $I$. The forward pre-analysis starts with the trivial partition $\mathbb{I} = \{Y\}$ (cf. Line 15). It proceeds forward for each element $I$ in $\mathbb{I}$ (cf. Lines 17-18). The transfer function $\text{ReLU}^p_A[x_{i,j}]$ considers a $\text{ReLU}$ operation and additionally builds an abstract activation pattern $p$ for $I$ (cf. Line 5) starting from the empty pattern $\epsilon$ (cf. Line 2).

If $I$ leads to a unique outcome (cf. Line 19), then causal fairness is already proved for $I$, and there is no need for a backward analysis; $I$ is added to the set of completed partitions (cf. Line 20). Instead, if abstract activation pattern $p$ fixes the activation status of enough $\text{ReLU}$s (cf. Line 21), we say that the backward analysis for $I$ is feasible. In this case, the pair of $p$ and $I$ is inserted into a map $F$ from abstract activation patterns to feasible partitions (cf. Line 22). The insertion takes care of merging abstract activation patterns that are subsumed by other (more) abstract patterns. In other words, it groups partitions whose abstract activation patterns fix more $\text{ReLU}$s with partitions whose patterns fix fewer $\text{ReLU}$s, and therefore, represent a superset of (concrete) patterns.

**Proof.** The proofs follows trivially from the definition of $\mathbb{G}_3$ (cf. Equation 6 and 8 and 10) and Lemma 6.1 and 8.1.

Finally, from Lemma 8.1, we have that we can equivalently verify whether $\{M\}_{i}^{\mathbb{I}} \subseteq \alpha_3(\mathbb{A}_{\mathbb{I}}(F_{\mathbb{K}}[Y]))$ by checking if the parallel semantics $\{M\}_{0}^{\mathbb{I}}$ induces a partition of each $I_{K}$.

**Lemma 8.4.** $M \models F_{\mathbb{K}}[Y] \iff \forall I \in \mathbb{I}: \forall A, B \in \{M\}_{i}^{\mathbb{I}} : (A_{i,0}^{\omega} \neq B_{i,0}^{\omega} \Rightarrow A_{i,0}^{\omega} \cap B_{i,0}^{\omega} = \emptyset)$

**Proof.** The proof follows trivially from Lemma 8.1. □

9 PARALLEL CAUSAL-FAIRNESS ANALYSIS

In this section, we build on the parallel semantics to design our novel perfectly parallel static analysis for causal fairness, which automatically finds a fair partition $\mathbb{I}$ and computes a sound over-approximation $\Pi_{\mathbb{I}}^{\mathbb{I}}$ of $\Pi_{\mathbb{I}}^{\mathbb{I}}$, i.e., $\Pi_{\mathbb{I}}^{\mathbb{I}} \subseteq \mathbb{G}_3(\Pi_{\mathbb{I}}^{\mathbb{I}})$.

**ReLU Activation Functions.** We again only consider $\text{ReLU}$ activation functions for now and postpone the discussion of other activation functions to the end of the section. The analysis is described in Algorithm 2. It combines a forward pre-analysis (Lines 15-24) with a backward analysis (Lines 28-38). The forward pre-analysis uses an abstract domain $A_1$ and builds partition $\mathbb{I}$, while the backward analysis uses an abstract domain $A_2$ and performs the actual causal-fairness analysis of a neural-network model $M$ with respect to its sensitive input nodes $K$ and a (representation of a) set of initial states $Y$ (cf. Line 13).

More specifically, the forward pre-analysis bounds the number of paths that the backward analysis has to explore. Indeed, not all of the $2^{|M|}$ paths of a model $M$ are necessarily viable starting from its input space.

In the rest of this section, we represent each path by an activation pattern, which determines the activation status of every $\text{ReLU}$ operation in $M$. More precisely, an activation pattern is a sequence of flags. Each flag $p_{i,j}$ represents the activation status of the $\text{ReLU}$ operation used to compute the value of hidden node $x_{i,j}$. If $p_{i,j}$ is $x_{i,j}$, the $\text{ReLU}$ is always active, otherwise the $\text{ReLU}$ is always inactive and $p_{i,j}$ is $\overline{x}_{i,j}$.

An abstract activation pattern gives the activation status of only a subset of the $\text{ReLU}$s of $M$, and thus, represents a set of activation patterns. $\text{ReLU}$s whose corresponding flag does not appear in an abstract activation pattern have an unknown (i.e., not fixed) activation status. Typically, only a relatively small number of abstract activation patterns is sufficient for covering the entire input space of a neural-network model. The design of our analysis builds on this key observation.

We set an analysis budget by providing an upper bound $U$ (cf. Line 13) on the number of tolerated $\text{ReLU}$s with an unknown activation status for each element $I$ of $\mathbb{I}$, i.e., on the number of paths that are to be explored by the backward analysis in each $I$. The forward pre-analysis starts with the trivial partition $\mathbb{I} = \{Y\}$ (cf. Line 15). It proceeds forward for each element $I$ in $\mathbb{I}$ (cf. Lines 17-18). The transfer function $\text{ReLU}^p_A[x_{i,j}]$ considers a $\text{ReLU}$ operation and additionally builds an abstract activation pattern $p$ for $I$ (cf. Line 5) starting from the empty pattern $\epsilon$ (cf. Line 2).

If $I$ leads to a unique outcome (cf. Line 19), then causal fairness is already proved for $I$, and there is no need for a backward analysis; $I$ is added to the set of completed partitions (cf. Line 20). Instead, if abstract activation pattern $p$ fixes the activation status of enough $\text{ReLU}$s (cf. Line 21), we say that the backward analysis for $I$ is feasible. In this case, the pair of $p$ and $I$ is inserted into a map $F$ from abstract activation patterns to feasible partitions (cf. Line 22). The insertion takes care of merging abstract activation patterns that are subsumed by other (more) abstract patterns. In other words, it groups partitions whose abstract activation patterns fix more $\text{ReLU}$s with partitions whose patterns fix fewer $\text{ReLU}$s, and therefore, represent a superset of (concrete) patterns.
Algorithm 2: Our Analysis Based on Activation Patterns

1: function \textsc{forward}(M, A, I)
2: \quad a, p \leftarrow \textsc{assume}_A[I](\textsc{new}_A), \varepsilon
3: \quad \textbf{for} i \leftarrow 1 \textbf{ up to } n \textbf{ do}
4: \quad \quad \textbf{for} j \leftarrow 0 \textbf{ up to } |L_i| \textbf{ do}
5: \quad \quad \quad a, p \leftarrow \textsc{relu}_A(x_{i,j}, p) (\textsc{assign}_A[x_{i,j}] a)
6: \quad \textbf{return} a, p

7: function \textsc{backward}(M, A, O, p)
8: \quad a \leftarrow \textsc{outcome}_A[O](\textsc{new}_A)
9: \quad \textbf{for} i \leftarrow n - 1 \textbf{ down to } 0 \textbf{ do}
10: \quad \quad \textbf{for} j \leftarrow |l_i| \textbf{ down to } 0 \textbf{ do}
11: \quad \quad \quad a \leftarrow \textsc{assign}_A[x_{i,j}](\textsc{relu}_A p(x_{i,j}, a))
12: \quad \textbf{return} a

13: function \textsc{analyze}(M, K, Y, A_1, A_2, L, U)
14: \quad F, E, C \leftarrow \emptyset, \emptyset, \emptyset \quad \triangleright F: \text{feasible}, E: \text{excluded}, C: \text{completed}
15: \quad I \leftarrow \{Y\}
16: \textbf{while} I \neq \emptyset \textbf{ do} \quad \triangleright \text{perfectly parallelizable}
17: \quad I \leftarrow I.\text{get}()
18: \quad a, p \leftarrow \textsc{forward}(M, A_1, I) \quad \triangleright I \text{ is already fair}
19: \quad \textbf{if} \textsc{uniquely-classified}(a) \textbf{ then}
20: \quad \quad C \leftarrow C \cup \{I\}
21: \quad \textbf{else if} |M| - |p| \leq U \textbf{ then}
22: \quad \quad F \leftarrow F \cup \{p \mapsto I\} \quad \triangleright I \text{ is feasible}
23: \quad \textbf{else if} |I| \leq L \textbf{ then}
24: \quad \quad E \leftarrow E \cup \{p \mapsto I\} \quad \triangleright I \text{ is excluded}
25: \quad \textbf{else}
26: \quad \quad I \leftarrow I \cup \textsc{partition}_K(I) \quad \triangleright I \text{ must be partitioned further}
27: \quad B \leftarrow \emptyset \quad \triangleright B: \text{biased}
28: \quad \textbf{for all} p, I \in F \textbf{ do} \quad \triangleright \text{perfectly parallelizable}
29: \quad \quad O \leftarrow \emptyset
30: \quad \quad \textbf{for} j \leftarrow 0 \textbf{ up to } |L_N| \textbf{ do}
31: \quad \quad \quad a \leftarrow \textsc{backward}(M, A_2, x_{N,j}, p)
32: \quad \quad \quad O \leftarrow O \cup \{x_{N,j} \mapsto a\}
33: \quad \textbf{for all} I \in I \textbf{ do}
34: \quad \quad O' \leftarrow \emptyset
35: \quad \quad \textbf{for all} o, a \in O \textbf{ do}
36: \quad \quad \quad O' \leftarrow O' \cup \{o \mapsto (\textsc{assume}_{A_2}[I] a)_{\text{K}}\}
37: \quad \quad B \leftarrow B \cup \textsc{check}(O')
38: \quad \quad C \leftarrow C \cup \{I\}
39: \quad \textbf{return} C, B = \emptyset, B, E \quad \triangleright \text{fair} \quad B = \emptyset, \text{maybe biased} \quad B \neq \emptyset

Otherwise, I needs to be partitioned further, with respect to K (cf. Line 25). Partitioning may continue until the size of I is smaller than the given lower bound L (cf. Lines 13 and 23). At this
point, it is set aside and excluded from the analysis until more resources (a larger upper bound $U$ or a smaller lower bound $L$) become available (cf. Line 24).

Note that the forward pre-analysis lends itself to choosing a relatively cheap abstract domain $A_1$ since it does not need to precisely handle polyhedral constraints (like $\max X_N = x$, needed to partition with respect to outcome, cf. Section 7).

The analysis then proceeds backwards, independently for each abstract activation path $p$ and associated group of partitions $I$ (cf. Lines 28 and 31). The transfer function $\overrightarrow{\text{ReLU}}^p_A[I]_{x_i, j}$ uses $p$ to choose which path(s) to explore at each ReLU operation, i.e., only the active (resp. inactive) path if $x_{i,j}$ (resp. $\overrightarrow{x_{i,j}}$) appears in $p$, or both if the activation status of the ReLU corresponding to hidden node $x_{i,j}$ is unknown. The (as we have seen, necessarily) expensive backward analysis only needs to run for each abstract activation pattern in the feasible map $F$. This is also why it is advantageous to merge subsumed abstract activation paths as described above.

Finally, the analysis checks causal fairness of each element $I$ associated to $p$ (cf. Line 37). The analysis returns the set of input-space regions $C$ that have been completed and a set $B$ of abstract-domain elements over-approximating the regions in which bias might occur (cf. Line 39). If $B$ is empty, then the given model $M$ satisfies causal fairness with respect to $K$ and $Y$ over $C$.

**Theorem 9.1.** If function $\text{analyze}(M, K, Y, A_1, A_2, L, U)$ in Algorithm 2 returns $C$, $\text{true}$, $\emptyset$, then $M$ satisfies $\mathcal{F}_K[Y]$ over the input-space fraction $C$.

**Proof (Sketch).** $\text{analyze}(M, K, Y, A_1, A_2, L, U)$ in Algorithm 2 first computes the abstract activation patterns that cover a fraction $C$ of the input space in which the analysis is feasible (Lines 15-24). Then, it computes an over-approximation $a$ of the regions of $C$ that yield each target class $x_{n,j}$ (cf. Line 31). Thus, it actually computes an over-approximation $\{M\}^{1}_{\omega}$ of the parallel semantics $\{M\}_{\omega}$, i.e., $\{M\}^{1}_{\omega} \subseteq \{M\}_{\omega}$. Thus, if $\{M\}^{1}_{\omega}$ satisfies $\mathcal{F}_K[Y]$, i.e., $\forall I \in \mathbb{I}: \forall A, B \in \{M\}^{1}_{\omega}: (A_1^{\omega} \neq B_1^{\omega} \Rightarrow A_0^{\omega} \cap B_0^{\omega} = \emptyset)$ (according to Lemma 8.4, cf. Lines 33-37), then by transitivity we can conclude that also $\{M\}_{\omega}$ necessarily satisfies $\mathcal{F}_K[Y]$. 

**Remark.** Recall that we assumed neural-network nodes to have real values (cf. Section 4). Thus, Theorem 9.1 is true for all choices of classical numerical abstract domains [Cousot and Cousot 1976; Cousot and Halbwachs 1978; Ghorbal et al. 2009; Miné 2006b, etc.] for $A_1$ and $A_2$. If we were to consider floating-point values instead, the only sound choices would be floating-point abstract domains [Chen et al. 2008; Miné 2004; Singh et al. 2019].

**Other Activation Functions.** Let us discuss how activation functions other than ReLUs would be handled. The only difference in Algorithm 2 would be the transfer functions $\overrightarrow{\text{ReLU}}^p_A[I]_{x_i, j}$ (cf. Line 5) and $\overrightarrow{\text{ReLU}}^p_A[I]_{\overrightarrow{x_i, j}}$ (cf. Line 11), which would have to be replaced with the transfer functions corresponding to the considered activation function.

Piecewise-linear activation functions, like Leaky ReLU$(x) = \max(x, k \cdot x)$ or Hard TanH$(x) = \max(-1, \min(x, 1))$, can be treated analogously to ReLUs. The case of Leaky ReLUs is trivial. For Hard TanHs, the patterns $p$ used in Algorithm 2 will consist of flags $p_{i,j}$ with three possible values, depending on whether the corresponding hidden node $x_{i,j}$ has value less than or equal to $-1$, greater than or equal to $1$, or between $-1$ and $1$. For these activation functions, our approach remains sound and, in practice, exact when using disjunctive polyhedra for the backward analysis.

Other activation functions, e.g., $\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$, can be soundly over-approximated [Singh et al. 2019] and similarly treated in a piecewise manner. In this case, however, we necessarily lose the exactness of the analysis, even when using disjunctive polyhedra.
10 IMPLEMENTATION

We implemented our causal-fairness analysis described in the previous section in a tool called libra. The implementation is written in python and is open source\(^3\).

**Tool Inputs.** libra takes as input a neural-network model \(M\) expressed as a python program (cf. Section 3), a specification of the input layer \(l_0\) of \(M\), an abstract domain for the forward pre-analysis, and budget constraints \(L\) and \(U\). The specification for \(l_0\) determines which input nodes correspond to continuous and (one-hot encoded) categorical data and, among them, which should be considered bias sensitive. We assume that continuous data is in the range \([0, 1]\). A set \(Y\) of initial states of interest is specified using an assumption at the beginning of the program representation of \(M\).

**Abstract Domains.** For the forward pre-analysis, choices of the abstract domain are either boxes [Cousot and Cousot 1976] (i.e., boxes in the following), or a combination of boxes and symbolic constant propagation [Li et al. 2019; Miné 2006a] (i.e., symbolic in the following), or the DEEPPOLY domain [Singh et al. 2019], which is designed for proving local robustness of neural networks. As previously mentioned, we use disjunctive polyhedra for the backward analysis. All abstract domains are built on top of the APRON abstract-domain library [Jeannet and Miné 2009].

**Parallelization.** Both the forward and backward analyses are parallelized to run on multiple CPU cores. The pre-analysis uses a queue from which each process draws a fraction \(I\) of \(Y\) (cf. Line 17). Fractions that need to be partitioned further are split in half along one of the non-sensitive dimensions (in a round-robin fashion), and the resulting (sub)fractions are put back into the queue (cf. Line 26). Feasible \(I\)s (with their corresponding abstract activation pattern \(p\)) are put into another queue (cf. Line 22) for the backward analysis.

**Tool Outputs.** The analysis returns the fractions of \(Y\) that were analyzed and any (sub)regions of these where bias was found. It also reports the percentage of the input space that was analyzed and (an estimate of) the percentage that was found biased according to a given probability distribution of the input space (uniform by default). To obtain the latter, we simply use the size of a box wrapped around each biased region. More precise but also costlier solutions exist [Barvinok 1994].

11 EXPERIMENTAL EVALUATION

In this section, we evaluate our approach by focusing on the following research questions:

- **RQ1:** Can our analysis detect seeded (i.e., injected) bias?
- **RQ2:** Is our analysis able to answer specific bias queries?
- **RQ3:** How does the model structure affect the scalability of the analysis?
- **RQ4:** How does the analyzed input-space size affect the scalability of the analysis?
- **RQ5:** How does the analysis budget affect the scalability-vs-precision tradeoff?
- **RQ6:** Can our analysis effectively leverage multiple CPUs?

11.1 Data

For our evaluation, we used public datasets from the UCI Machine Learning Repository and ProPublica (see below for more details) to train several neural-network models. We primarily focused on datasets discussed in the literature [Mehrabi et al. 2019] or used by related techniques (e.g., [Albarghouthi et al. 2017a,b; Albarghouthi and Vinitsky 2019; Bastani et al. 2019; Datta et al. 2017; Galhotra et al. 2017; Tramèr et al. 2017; Udeshi et al. 2018]).

We pre-processed these datasets both to make them fair with respect to a certain sensitive input feature as well as to seed bias. We describe how we seeded bias in each particular dataset later on.

\(^3\)https://github.com/caterinaurban/Libra
Our methodology for making the data fair was common across datasets. In particular, given an original dataset and a sensitive feature (say, race), we selected the largest population with a particular value for this feature (say, Caucasian) from the dataset (and discarded all others). We removed any duplicate or inconsistent entries from this population. We then duplicated the population for every other value of the sensitive feature (say, Asian and Hispanic). For example, assuming the largest population was 500 Caucasians, we created 500 Asians and 500 Hispanics, and any two of these populations differ only in the value of race. Consequently, the new dataset is causally fair because there do not exist two inputs \( k \) and \( k' \) that differ only in the value of the sensitive feature for which the classification outcomes are different.

We define the causal-unfairness score of a dataset as the percentage of inputs \( k \) in the dataset for which there exists another input \( k' \) that differs from \( k \) only in the value of the sensitive feature and the classification outcome. Our fair datasets have an unfairness score of 0%.

All datasets used in our experiments are open source as part of LIBRA.

### 11.2 Setup

Since neural-network training is non-deterministic, we typically train eight neural networks on each dataset, unless stated otherwise. The model sizes range from 2 hidden layers with 5 nodes each to 32 hidden layers with 40 nodes each. All models used in our experiments are open source as part of LIBRA. For each model, we assume a uniform distribution of the input space.

We performed all experiments on a 12-core Intel ® Xeon ® X5650 CPU @ 2.67GHz machine with 48GB of memory, running Debian GNU/Linux 9.6 (stretch).

### 11.3 Results

In the following, we present our experimental results for each of the above research questions.

**RQ1: Detecting Seeded Bias.** This research question focuses on detecting seeded bias by comparing the analysis results for models trained with fair versus biased data.

For this experiment, we used the German Credit dataset\(^4\). This dataset classifies creditworthiness into two categories, “good” and “bad”. An input feature is age, which we consider sensitive to bias. (Recall that this could also be an input feature that the user considers indirectly sensitive to bias.) We seeded bias in the fair dataset by randomly assigning a bad credit score to people of age 60 and above who request a credit amount of more than EUR 1 000 until we reached a 20% causal-unfairness score of the dataset. The median classification accuracy of the models (17 inputs and 4 hidden layers with 5 nodes each) trained on fair and biased data was 71% and 65%, respectively. Note that accuracy does not improve by adding more layers or nodes per layer — we tried up to 100 hidden layers with 100 nodes each.

To analyze these models, we set \( L = 0 \) to be sure to complete the analysis on 100% of the input space. The drawback with this is that the pre-analysis might end up splitting input partitions

---

\( ^4 \)https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)
endlessly. To counteract, for each model, we chose the smallest upper bound $U$ that did not cause this issue. Table 1 shows the analysis results for the different choices of domain used for the forward pre-analysis. In particular, it shows whether the models are biased with respect to age for credit requests of 1,000 or less as well as for credit requests of over 1,000. Columns $\text{BIAS}$ and $\text{TIME}$ show the detected bias (in percentage of the entire input space) and the analysis running time. We show minimum, median, and maximum bias percentage and running time for each credit request group. For each line in Table 1, we highlighted the choice of the abstract domain that entailed the shortest analysis time. The analysis results for all models are shown in the appendix (cf. Tables 7–9).

For all models, the analysis finds little bias for small credit amounts, as intended. Instead, for large credit amounts, the analysis finds significantly more bias (i.e., about three times as much median bias) for the models trained on biased data in comparison to models trained on fair data. This demonstrates that our approach is able to effectively detect seeded bias.

For the models trained on fair data, we observe a maybe unexpected difference in the bias found for small credit amounts compared to larger amounts. This is in part due to the fact that bias is given in percentage of the entire input space and not scaled with respect to the analyzed input space. When considering the analyzed input space (small credit amounts correspond to a mere 4% of the input space), the difference is less marked: the median bias is 0.19% / 4% = 4.75% for small credit amounts and 6.72% / 96% = 7% (or 6.63% / 96% = 6.9% for the DEEPPOLY domain) for large credit amounts. The remaining difference indicates that the models contain bias that does not necessarily depend on the credit amount. The bias is introduced by the training process itself (as explained in the Introduction) and is not due to imprecision of our analysis. Recall that our approach is exact, and imprecision is only introduced when estimating the bias percentage (cf. Section 10).

**RQ2: Answering Bias Queries.** To further evaluate the precision of our approach, we created queries concerning bias within specific groups of people, each corresponding to a subset of the entire input space. We used the COMPAS dataset5 from ProPublica for this experiment. The data assigns a three-valued recidivism-risk score (high, medium, and low) indicating how likely criminals are to re-offend. The data includes both personal attributes (e.g., age and race) as well as criminal history (e.g., number of priors and violent crimes). As for RQ1, we trained models both on fair and biased data. Here, we considered race as the sensitive feature. We seeded bias in the fair data by randomly assigning high recidivism risk to African Americans until we reached a 20% causal-unfairness score of the dataset. The median classification accuracy of the 3-class models (19 inputs and 4 hidden layers with 5 nodes each) trained on fair and biased data was 55% and 56%, respectively. Accuracy does not improve with larger networks — we tried up to 100 hidden layers with 100 nodes each.

To analyze these models, we used a lower bound $L$ of 0, and an upper bound $U$ between 7 and 19. Table 2 shows the results of our analysis (i.e., columns shown as in Table 1) for three queries:

---


---

<table>
<thead>
<tr>
<th>QUERY</th>
<th>FAIR DATA</th>
<th>BIASED DATA</th>
<th>SYMBOLIC</th>
<th>DEEPPOLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGES</td>
<td>BIAS TIME</td>
<td>BIAS TIME</td>
<td>BIAS TIME</td>
<td>BIAS TIME</td>
</tr>
<tr>
<td>0.22%</td>
<td>24m 32s</td>
<td>0.12%</td>
<td>14m 54s</td>
<td>0.12%</td>
</tr>
<tr>
<td>0.20%</td>
<td>2h 17m 3s</td>
<td>0.23%</td>
<td>36m 6s</td>
<td>0.22%</td>
</tr>
<tr>
<td>2.46%</td>
<td>3h 29m 19s</td>
<td>8.50%</td>
<td>3h 34m 30s</td>
<td>1h 11m 43s</td>
</tr>
<tr>
<td>RACE BIAS</td>
<td>MIN</td>
<td>MEDIAN</td>
<td>MAX</td>
<td>MIN</td>
</tr>
<tr>
<td>0.22%</td>
<td>2h 36m 1s</td>
<td>4.21%</td>
<td>1h 34m 7s</td>
<td>6.95%</td>
</tr>
<tr>
<td>2.46%</td>
<td>8m 46s</td>
<td>5m 18s</td>
<td>2h 5m 5s</td>
<td>2h 17m 3s</td>
</tr>
<tr>
<td>5.36%</td>
<td>45m 2m 12s</td>
<td>6.98%</td>
<td>70h 50m 16s</td>
<td>20%</td>
</tr>
</tbody>
</table>

---

Table 2. Queries on Models Trained on Fair and Race-Biased Data (ProPublica’s COMPAS Data)
\( Q_A \): Is there bias with respect to race for people younger than 25?
\( Q_B \): Is there bias with respect to age for males?
\( Q_C \): Is there bias with respect to the number of priors for Caucasians?

For \( Q_A \), the analysis detects only a small percentage of race bias in the fair models, but as intended, the race bias is found to be significantly higher (about three times as much median bias) for the biased models. In contrast, for \( Q_B \), the analysis finds a comparable amount of age bias across both sets of models. This becomes more evident when scaling the median bias with respect to the queried input space (males correspond to 50% of the input space): the smallest median bias for the models trained on fair data is 12.16% (for the boxes domain) and the largest median bias for the models trained on biased data is 14.04% (for the symbolic domain). This bias is not intended and was either present in the original data or introduced by the training process (or both). Finally, for \( Q_C \), the analysis detects significant bias across both sets of models with respect to the number of priors. When considering the queried input space (Caucasians represent 1/6 of the entire input space), this translates to 17.7% median bias for the models trained on fair data and 25.26% for the models trained on biased data. This bias is intended and present in the original data: as one would expect, recidivism risk differs for different numbers of priors. Overall, these results demonstrate the effectiveness of our analysis in answering specific bias queries.

For each line in Table 2, we highlighted the choice of abstract domain that entailed the shortest analysis time. We observe that deeppoly seems generally the better choice. The difference in performance becomes more striking as the analyzed input space becomes smaller, i.e., for \( Q_C \). This is because deeppoly is specifically designed for proving local robustness of neural networks. Thus, our input partitioning, in addition to allowing for parallelism, is also enabling analyses designed for local properties to prove global properties, like causal fairness.

The analysis results for all models are shown in the appendix (see Tables 10, 11, and 12).

**RQ3: Effect of Model Structure on Scalability.** To evaluate the effect of the model structure on the scalability of our analysis, we trained models on the Adult Census dataset\(^6\) by varying the number of layers and nodes per layer. The dataset assigns a yearly income (\(> \) or \(\leq\) USD 50K) based on personal attributes such as gender, race, and occupation. We trained all models (with 23 inputs) on a fair dataset with respect to gender and ensured that each model reached a minimum classification accuracy of 78%. Accuracy does not increase by adding more layers or nodes per layer, in fact, it may significantly decrease — we tried up to 100 hidden layers with 100 nodes each.

Table 3 shows the results. The first column (|\(M|\)) shows the total number of hidden nodes and introduces the marker symbols used in the scatter plot of Figure 3 (to identify the domain used for the forward pre-analysis: left, center, and right symbols respectively refer to the boxes, symbolic, and deeppoly domains). The models have the following number of hidden layers and nodes per layer (from top to bottom): 2 and 5; 4 and 3; 4 and 5; 4 and 10; 9 and 5.

Column \(U\) shows the chosen upper bound for the analysis. For each model, we tried four different choices of \(U\). Column input shows the input-space coverage, i.e., the percentage of the input space that was completed by the analysis. Column |\(C|\) shows the total number of analyzed (i.e., completed) input space partitions. Column |\(F|\) shows the total number of abstract activation patterns (left) and feasible input partitions (right) that the backward analysis had to explore. The difference between |\(C|\) and the number of partitions shown in |\(F|\) are the input partitions that the pre-analysis found to be already fair (i.e., uniquely classified). Finally, column time shows the analysis running time.

---

\(6\)https://archive.ics.uci.edu/ml/datasets/adult
Table 3. Comparison of Different Model Structures (Adult Census Data)

<table>
<thead>
<tr>
<th>[M]</th>
<th>U</th>
<th>Analysis Time</th>
<th>Analyzed Input Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>0.74%</td>
<td>99</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>0.5%</td>
<td>99</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>0.5%</td>
<td>99</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
<td>0.5%</td>
<td>99</td>
</tr>
</tbody>
</table>

(a) Fig. 3. Comparison of Different Model Structures (Adult Census Data)

(b) Zoom on Best U-Configurations

The scatter plot of Figure 3a visualizes the input coverage and analysis running time. We zoom in on the best U-configurations for each pre-analysis domain (i.e., the chosen U) in Figure 3b.

Overall, we observe that coverage decreases for larger model structures, and the more precise symbolic and deeppoly domains result in a significant coverage boost, especially for larger structures. We also note that, as in this case we are analyzing the entire input space, deeppoly generally performs worse than the symbolic domain. In particular, for larger structures, the symbolic domain often yields a higher input coverage in a shorter analysis running time. Finally, we observe that increasing the upper bound U tends to increase coverage independently of the specific model structure. However, interestingly, this does not always come at the expense of an increased running time. In fact, such a change often results in decreasing the number of partitions that the expensive backward analysis needs to analyze (cf. columns [F]) and, in turn, this reduces the overall running time.
### Table 4. Comparison of Different Input Space Sizes and Model Structures (Adult Census Data)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>F</td>
<td>100.00%</td>
<td>9</td>
<td>2</td>
<td>3m 3s</td>
<td>3m 5s</td>
<td>100.00%</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3m 5s</td>
<td>100.00%</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>99.96%</td>
<td>83</td>
<td>9</td>
<td>3m 13s</td>
<td>3m 8s</td>
<td>100.00%</td>
<td>26</td>
<td>3</td>
<td>9</td>
<td>3m 8s</td>
<td>100.00%</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>99.97%</td>
<td>1.04%</td>
<td>1.04%</td>
<td>1.04%</td>
<td>1.04%</td>
<td>100.00%</td>
<td>292</td>
<td>9</td>
<td>63</td>
<td>4m 50s</td>
<td>100.00%</td>
<td>287</td>
</tr>
<tr>
<td>80</td>
<td>C</td>
<td>99.69%</td>
<td>3173</td>
<td>20</td>
<td>1212</td>
<td>36m 12s</td>
<td>100.00%</td>
<td>2668</td>
<td>13</td>
<td>417</td>
<td>17m 40s</td>
<td>100.00%</td>
<td>2897</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>97.18%</td>
<td>50%</td>
<td>15415</td>
<td>51</td>
<td>5646</td>
<td>1.99%</td>
<td>99.99%</td>
<td>99.99%</td>
<td>15445</td>
<td>34</td>
<td>2112</td>
<td>1.99%</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>94.03%</td>
<td>100%</td>
<td>18642</td>
<td>70</td>
<td>8700</td>
<td>2h 30m 46s</td>
<td>0.96%</td>
<td>99.91%</td>
<td>99.91%</td>
<td>15445</td>
<td>40</td>
<td>3481</td>
</tr>
<tr>
<td>320</td>
<td>F</td>
<td>99.91%</td>
<td>0.09%</td>
<td>6</td>
<td>0</td>
<td>3m 15s</td>
<td>99.44%</td>
<td>0</td>
<td>0</td>
<td>3m 35s</td>
<td>99.91%</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>99.58%</td>
<td>0.14%</td>
<td>120</td>
<td>1</td>
<td>3m 9s</td>
<td>99.627%</td>
<td>0.104%</td>
<td>120</td>
<td>1</td>
<td>6m 34s</td>
<td>99.58%</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>97.91%</td>
<td>1.04%</td>
<td>151</td>
<td>0</td>
<td>2m 56s</td>
<td>98.247%</td>
<td>1.04%</td>
<td>297</td>
<td>0</td>
<td>3m 41s</td>
<td>97.91%</td>
<td>301</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>83.53%</td>
<td>0.33%</td>
<td>506</td>
<td>2</td>
<td>3h 1m</td>
<td>95.642%</td>
<td>7.996%</td>
<td>885</td>
<td>25</td>
<td>34</td>
<td>13h</td>
<td>83.53%</td>
</tr>
<tr>
<td>1280</td>
<td>B</td>
<td>25.00%</td>
<td>0.50%</td>
<td>5744</td>
<td>0</td>
<td>2h 24m 36s</td>
<td>46.063%</td>
<td>4766</td>
<td>0</td>
<td>7h 25m 57s</td>
<td>25.074%</td>
<td>5762</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>0.00%</td>
<td>100%</td>
<td>24</td>
<td>0</td>
<td>2h 54m 25s</td>
<td>24.258%</td>
<td>2436</td>
<td>0</td>
<td>9h 41m 36s</td>
<td>0.017%</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

### RQ4: Effect of Analyzed Input Space on Scalability.

As said above, the analysis of the models considered in Table 3 is conducted on the entire input space. In practice, as already mentioned, one might be interested in just a portion of the input space, e.g., depending on the probability distribution. More generally, we argue that the size of the analyzed input space (rather than the size of the analyzed neural network) is the most important factor that affects the performance of the analysis. To support this claim, we trained even larger models and analyzed them with respect to queries exercising different input space sizes. Table 4 shows the results. The first column again shows the total number of hidden nodes for each trained model. In particular, the models we analyzed have the following number of hidden layers and nodes per layer (from top to bottom): 4 and 5; 8 and 10; 16 and 20; 32 and 40. Column QUERY shows the query used for the analysis and the corresponding queried input space size. Specifically, the queries identify people with the following characteristics:

- **A**: true

- **queried input space: 100.00%**
For the analysis budget, we used $L = 0.25$, $U = 0.1 \times |M|$, and a time limit of 13h. Column INPUT shows, for each domain used for the forward pre-analysis, the coverage of the queried input space (i.e., the percentage of the input space that satisfies the query and was completed by the analysis) and the corresponding input-space coverage (i.e., the same percentage but this time scaled to the entire input space). Columns U, |C|, |F|, and TIME are as before. Where a timeout is indicated (i.e., TIME > 13h) and the values for the INPUT, |C|, and |F| columns are missing, it means that the timeout occurred during the pre-analysis; otherwise, it happened during the backward analysis.

For each model and query, we highlighted the configuration (i.e., the abstract domain used for the pre-analysis) that achieved the highest input-space coverage with the shortest analysis running time. Note that, where the |F| column only contains zeros, it means that the backward analysis had no activation patterns to explore; this implies that the entire covered input space (i.e., the percentage shown in the INPUT column) was already certified to be fair by the forward analysis.

Overall, we observe that whenever the analyzed input space is small enough (i.e., queries $D - F$), the size of the neural network has little influence on the input space coverage and slightly impacts the analysis running time, independently of the domain used for the forward pre-analysis. Instead, for larger analyzed input spaces (i.e., queries $A - C$) performance degrades quickly for larger neural networks. These results thus support our claim. Again, as expected, we observe that the SYMBOLIC domain generally is the better choice for the forward pre-analysis, in particular for queries exercising a larger input space or larger neural networks.

**RQ5: Scalability-vs-Precision Tradeoff.** To evaluate the effect of the analysis budget (bounds L and U), we analyzed a model using different budget configurations. For this experiment, we used the Japanese Credit Screening\(^8\) dataset, which we made fair with respect to gender. Our 2-class model (17 inputs and 4 hidden layers with 5 nodes each) had a classification accuracy of 86%. Note that accuracy does not increase by adding more layers or nodes per layer, in fact, it may significantly decrease — we tried up to 100 hidden layers with 100 nodes each.

Table 5 shows the results of the analysis for different budget configurations and choices for the domain used for the forward pre-analysis. The best configuration in terms of input-space coverage and analysis running time is highlighted. The symbol next to each domain name introduces the marker used in the scatter plot of Figure 4a, which visualizes the coverage and running time. Figure 4b zooms on 90.00% $\leq$ INPUT and 1000s $\leq$ TIME $\leq$ 1000s.

Overall, we observe that the more precise SYMBOLIC and DEEPPOLY domains boost input coverage, most noticeably for configurations with a larger L. This additional precision does not always result in longer running times. In fact, a more precise pre-analysis often reduces the overall running time. This is because the pre-analysis is able to prove that more partitions are already fair without requiring them to go through the backward analysis (cf. columns |F|).

Independently of the chosen domain for the forward pre-analysis, as expected, a larger U or a smaller L increase precision. Increasing U or L typically reduces the number of completed partitions (cf. columns |C|). Consequently, partitions tend to be more complex, requiring both forward and backward analyses. Since the backward analysis tends to dominate the running time, more partitions

---

\(^7\)This corresponds to $age \leq 0.5$ with min-max scaling between 0 and 1.

\(^8\)https://archive.ics.uci.edu/ml/datasets/Japanese+Credit+Screening
Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang

Table 5. Comparison of Different Analysis Configurations (Japanese Credit Screening) — 12 CPUs

<table>
<thead>
<tr>
<th>L</th>
<th>U</th>
<th>INPUT</th>
<th>BOXES</th>
<th>SYMBOLIC</th>
<th>DEEPPOLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>4</td>
<td>15.28%</td>
<td>37</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>17.01%</td>
<td>39</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>51.39%</td>
<td>90</td>
<td>28</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>79.86%</td>
<td>89</td>
<td>34</td>
<td>89</td>
</tr>
<tr>
<td>0.25</td>
<td>4</td>
<td>59.09%</td>
<td>1115</td>
<td>20</td>
<td>415</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>83.77%</td>
<td>1404</td>
<td>79</td>
<td>944</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>96.07%</td>
<td>869</td>
<td>140</td>
<td>761</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>99.54%</td>
<td>409</td>
<td>93</td>
<td>403</td>
</tr>
<tr>
<td>0.125</td>
<td>4</td>
<td>97.13%</td>
<td>12449</td>
<td>200</td>
<td>9519</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>99.83%</td>
<td>5919</td>
<td>276</td>
<td>4460</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>99.98%</td>
<td>1926</td>
<td>203</td>
<td>1568</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>100.00%</td>
<td>428</td>
<td>95</td>
<td>427</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>100.00%</td>
<td>19299</td>
<td>295</td>
<td>15446</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>100.00%</td>
<td>4843</td>
<td>280</td>
<td>3679</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>100.00%</td>
<td>1919</td>
<td>208</td>
<td>1567</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>100.00%</td>
<td>486</td>
<td>102</td>
<td>475</td>
</tr>
</tbody>
</table>

Fig. 4. Comparison of Different Analysis Configurations (Japanese Credit Screening)

Analyzed Input Space

Analysis Time

(a)

(b) Zoom on 90.00% ≤ INPUT and 1000s ≤ TIME ≤ 1000s

Analyzed Input Space

Analysis Time

Analyzed Input Space

Analysis Time

Analyzed Input Space

Analysis Time

generally increase the running time (when comparing configurations with similar coverage). Based on our experience, the optimal budget largely depends on the analyzed model.

**RQ6: Leveraging Multiple CPUs.** To evaluate the effect of parallelizing the analysis using multiple cores, we re-ran the analyses of RQ5 on 4 CPU cores instead of 12. Table 6 shows these results. We observe the most significant increase in running time for 4 cores for the **symbolic** and **deeppoly** domains, the running time with 4 cores increases less drastically, on average by a factor of 1.6 and 2, respectively. This is again explained by the increased precision of the forward analysis; fewer partitions require a backward pass, where parallelization is most effective.

The appendix includes the same experiment on 24 vCPUs (see Table 13).

**12 RELATED WORK**

Significant progress has been made on testing and verifying machine-learning models. We focus on fairness, safety, and robustness properties in the following, especially of deep neural networks.
Fairness Criteria. There are countless fairness definitions in the literature. In this paper, we focus on causal fairness (specifically the fairness notion considered by Galhotra et al. [Galhotra et al. 2017]) and compare here with the most popular and related notions.

Demographic parity or group fairness [Feldman et al. 2015] is the most common non-causal notion of fairness. It states that individuals with different values of sensitive features, hence belonging to different groups, should have the same probability of being predicted to the positive class. For example, a loan system satisfies group fairness with respect to gender if male and female applicants have equal probability of getting loans. If unsatisfied, this notion is also referred to as disparate impact. Our notion of fairness is stronger, as it imposes fairness on every pair of individuals that differ only in sensitive features. A classifier that satisfies group fairness does not necessarily satisfy causal fairness, because there may still exist pairs of individuals on which the classifier exhibits bias.

Another group-based notion of fairness is equality of opportunity [Hardt et al. 2016]. It states that qualified individuals with different values of sensitive features should have equal probability of being predicted to the positive class. For a loan system, this means that male and female applicants who are qualified to receive loans should have an equal chance of being approved. By imposing fairness on every qualified pair of individuals that differ only in sensitive features, we can generalize causal fairness to also concern both prediction and actual results. We can then adapt our technique to consider only the part of the input space that includes qualified individuals.

Other causal notions of fairness [Chiappa 2019; Kilbertus et al. 2017; Kusner et al. 2017; Nabi and Shpitser 2018, etc.] require additional knowledge in the form of a causal model. A causal model can drive the choice of the sensitive input(s) for our analysis.

Testing and Verifying Fairness. Galhotra et al. [Galhotra et al. 2017] proposed an approach, Themis, that allows efficient fairness testing of software. Udeshi et al. [Udeshi et al. 2018] designed an automated and directed testing technique to generate discriminatory inputs for machine-learning models. Tramer et al. [Tramèr et al. 2017] introduced the unwarranted-associations framework and instantiated it in FairTest. In contrast, our technique provides formal fairness guarantees.

Bastani et al. [Bastani et al. 2019] used adaptive concentration inequalities to design a scalable sampling technique for providing probabilistic fairness guarantees for machine-learning models. As mentioned in the Introduction, our approach differs in that it gives definite (instead of probabilistic) guarantees. However, it might exclude partitions for which the analysis is not exact.
Albarghouthi et al. [Albarghouthi et al. 2017b] encoded fairness problems as probabilistic program properties and developed an SMT-based technique for verifying fairness of decision-making programs. As discussed in the Introduction, this technique has been shown to scale only up to neural networks with at most 3 inputs and a single hidden layer with at most 2 nodes. In contrast, our approach is designed to be perfectly parallel, and thus, is significantly more scalable.

A recent technique [Ruoss et al. 2020] certifies individual fairness of neural networks, which is a local property that coincides with robustness within a particular distance metric. In particular, individual fairness dictates that similar individuals should be treated similarly. Our approach, however, targets certification of neural networks for the global property of causal fairness.

For certain biased decision-making programs, the program repair technique proposed by Albarghouthi et al. [Albarghouthi et al. 2017a] can be used to repair their bias. Albarghouthi and Vinitsky [Albarghouthi and Vinitsky 2019] further introduced fairness-aware programming, where programmers can specify fairness properties in their code for runtime checking.

Robustness of Deep Neural Networks. Robustness is a desirable property for traditional software [Chaudhuri et al. 2012; Goubault and Putot 2013; Majumdar and Saha 2009], especially control systems. Deep neural networks are also expected to be robust. However, research has shown that deep neural networks are not robust to small perturbations of their inputs [Szegedy et al. 2014] and can even be easily fooled [Nguyen et al. 2015]. Subtle imperceptible perturbations of inputs, known as adversarial examples, can change their prediction results. Various algorithms [Carlini and Wagner 2017b; Goodfellow et al. 2015; Madry et al. 2018; Tabacof and Valle 2016; Zhang et al. 2019] have been proposed that can effectively find adversarial examples. Research on developing defense mechanisms against adversarial examples [Athalye et al. 2018; Carlini and Wagner 2016, 2017a,b; Cornelius 2019; Engstrom et al. 2018; Goodfellow et al. 2015; Huang et al. 2015; Mirman et al. 2018, 2019] is also active. Causal fairness is a special form of robustness in the sense that neural networks are expected to be globally robust with respect to their sensitive features.


13 CONCLUSION AND FUTURE WORK

We have presented an automated, perfectly parallel analysis for certifying fairness of neural networks. The analysis is configurable to support a wide range of use cases throughout the development lifecycle of neural networks: ranging from short sanity checks during development to formal fairness audits before deployments.

In future work, we plan to extend our technique in various ways, for instance, by automatically tuning parameters (such as the upper bound $U$) during the analysis or by feeding analysis results to other tools. Such tools may be used to provide probabilistic fairness guarantees for partitions that could not be certified or repair networks by eliminating bias.

REFERENCES


Eric Goubault and Sylvie Putot. 2013. Robustness Analysis of Finite Precision Implementations. In APLAS. 50–57. https://doi.org/10.1007/978-3-319-03542-0_4


Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety Verification of Deep Neural Networks. In CAV. 3–29. https://doi.org/10.1007/978-3-319-63387-9_1


Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. How We Analyzed the COMPAS Recidivism Algorithm. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm


Table 7. Analysis of Neural Networks Trained on Fair and [Age, Credit > 1000]-Biased Data (German Credit Data) — Full Table (boxes Domain)

<table>
<thead>
<tr>
<th>CREDIT</th>
<th>FAIR DATA</th>
<th></th>
<th></th>
<th>BIASED DATA</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U</td>
<td>BIAS</td>
<td>[C]</td>
<td>[F]</td>
<td>TIME</td>
<td>U</td>
</tr>
<tr>
<td>≤ 1000</td>
<td>8</td>
<td>0.33%</td>
<td>144</td>
<td>32</td>
<td>39</td>
<td>7m 7s</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.17%</td>
<td>182</td>
<td>35</td>
<td>56</td>
<td>30m 59s</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.09%</td>
<td>167</td>
<td>13</td>
<td>24</td>
<td>2m 2s</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>0.15%</td>
<td>157</td>
<td>30</td>
<td>34</td>
<td>17m 30s</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.23%</td>
<td>169</td>
<td>36</td>
<td>67</td>
<td>4m 24s</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>0.30%</td>
<td>173</td>
<td>57</td>
<td>82</td>
<td>12m 36s</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.20%</td>
<td>134</td>
<td>30</td>
<td>38</td>
<td>3m 13s</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.16%</td>
<td>172</td>
<td>16</td>
<td>19</td>
<td>47s</td>
</tr>
<tr>
<td>MIN MEDIAN MAX</td>
<td>0.09%</td>
<td>47s</td>
<td>0.09%</td>
<td>2m 17s</td>
<td>0.19%</td>
<td>5m 46s</td>
</tr>
<tr>
<td>&gt; 1000</td>
<td>13</td>
<td>12.20%</td>
<td>208</td>
<td>76</td>
<td>139</td>
<td>53m 27s</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>7.43%</td>
<td>211</td>
<td>86</td>
<td>185</td>
<td>3h 45m 20s</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2.21%</td>
<td>207</td>
<td>23</td>
<td>42</td>
<td>1m 42s</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>4.29%</td>
<td>180</td>
<td>45</td>
<td>75</td>
<td>36m 36s</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>9.73%</td>
<td>433</td>
<td>139</td>
<td>329</td>
<td>16m 14s</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>14.96%</td>
<td>230</td>
<td>92</td>
<td>197</td>
<td>7h 11m 12s</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>6.00%</td>
<td>243</td>
<td>99</td>
<td>145</td>
<td>22m 1s</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>4.61%</td>
<td>237</td>
<td>63</td>
<td>96</td>
<td>26m 48s</td>
</tr>
<tr>
<td>MIN MEDIAN MAX</td>
<td>2.21%</td>
<td>1m 42s</td>
<td>4.52%</td>
<td>2m 11s</td>
<td>6.72%</td>
<td>31m 42s</td>
</tr>
</tbody>
</table>

Table 8. Analysis of Neural Networks Trained on Fair and [Age, Credit > 1000]-Biased Data (German Credit Data) — Full Table (symbolic Domain)

<table>
<thead>
<tr>
<th>CREDIT</th>
<th>FAIR DATA</th>
<th></th>
<th></th>
<th>BIASED DATA</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U</td>
<td>BIAS</td>
<td>[C]</td>
<td>[F]</td>
<td>TIME</td>
<td>U</td>
</tr>
<tr>
<td>≤ 1000</td>
<td>7</td>
<td>0.33%</td>
<td>138</td>
<td>22</td>
<td>32</td>
<td>32s</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.17%</td>
<td>165</td>
<td>19</td>
<td>23</td>
<td>4m 8s</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.09%</td>
<td>140</td>
<td>8</td>
<td>10</td>
<td>29s</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.15%</td>
<td>159</td>
<td>21</td>
<td>22</td>
<td>2m 5s</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.23%</td>
<td>157</td>
<td>14</td>
<td>25</td>
<td>1m 49s</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.30%</td>
<td>173</td>
<td>23</td>
<td>32</td>
<td>1m 10s</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.20%</td>
<td>135</td>
<td>23</td>
<td>25</td>
<td>1m 0s</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.16%</td>
<td>168</td>
<td>13</td>
<td>14</td>
<td>13s</td>
</tr>
<tr>
<td>MIN MEDIAN MAX</td>
<td>0.09%</td>
<td>13s</td>
<td>0.09%</td>
<td>1m 10s</td>
<td>0.19%</td>
<td>1m 5s</td>
</tr>
<tr>
<td>&gt; 1000</td>
<td>12</td>
<td>12.20%</td>
<td>202</td>
<td>56</td>
<td>101</td>
<td>32m 1s</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>7.43%</td>
<td>215</td>
<td>60</td>
<td>103</td>
<td>2h 23m 9s</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4.21%</td>
<td>161</td>
<td>11</td>
<td>18</td>
<td>38s</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>4.29%</td>
<td>203</td>
<td>41</td>
<td>54</td>
<td>6m 53s</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>9.73%</td>
<td>234</td>
<td>38</td>
<td>74</td>
<td>2m 56s</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>14.96%</td>
<td>282</td>
<td>82</td>
<td>168</td>
<td>4h 16m 52s</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>6.00%</td>
<td>261</td>
<td>106</td>
<td>80</td>
<td>6m 6s</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>4.61%</td>
<td>228</td>
<td>51</td>
<td>66</td>
<td>11m 4s</td>
</tr>
<tr>
<td>MIN MEDIAN MAX</td>
<td>2.21%</td>
<td>38s</td>
<td>4.52%</td>
<td>3m 7s</td>
<td>6.72%</td>
<td>8m 59s</td>
</tr>
</tbody>
</table>
A. **SUPPLEMENTARY MATERIAL FOR SECTION 11 (EXPERIMENTAL EVALUATION)**

### A.1 RQ1: Detecting Seeded Bias

Tables 7, 8 and 9 show the analysis results for all eight models trained on the German Credit dataset. Column U shows the chosen upper bound for each model. As before, column BIAS shows the detected bias, in percentage of the entire input space. We also again show minimum, median, and maximum bias percentage for each credit request group. Column |C| shows the total number of analyzed (i.e., completed) input space partitions. Column |F| shows the total number of abstract activation patterns (left) and feasible input partitions (right) that the backward analysis had to explore. Finally, column TIME shows the analysis running time. Again, we also show minimum, median, and maximum running time for each credit request group. For all models, we highlighted across all tables the choice of the abstract domain that entailed the shortest analysis time.

### A.2 RQ2: Answering Bias Queries

Table 10, 11 and 12 show the analysis results for all eight models trained on the COMPAS dataset from ProPublica. All columns are shown as before and, again, we highlighted across all tables the choice of the abstract domain that entailed the shortest analysis time.

### A.3 RQ6: Leveraging Multiple CPUs.

Table 13 shows the results of the experiment with the Japanese Credit Screening dataset on 24 vCPU.

---

<table>
<thead>
<tr>
<th>CREDIT</th>
<th>U</th>
<th>BIAS</th>
<th>FAIR DATA</th>
<th>DEEPPOLY</th>
<th>BIASED DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>≤ 1000</td>
<td>8</td>
<td>0.33%</td>
<td>70</td>
<td>21</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.17%</td>
<td>213</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.09%</td>
<td>176</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.15%</td>
<td>212</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.23%</td>
<td>217</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>&gt; 1000</td>
<td>12</td>
<td>0.30%</td>
<td>213</td>
<td>17</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.20%</td>
<td>193</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.16%</td>
<td>193</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

---

**Table 9. Analysis of Neural Networks Trained on Fair and [Age, Credit > 1000]-Biased Data (German Credit Data) — Full Table (DEEPPOLY Domain)**
## Table 10. Queries on Neural Networks Trained on Fair and Race-Biased Data (ProPublica’s compas Data) — Full Table (boxes Domain)

<table>
<thead>
<tr>
<th>QUERY</th>
<th>FAIR DATA</th>
<th>BOXES</th>
<th>BIASED DATA</th>
<th>BOXES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U</td>
<td>BIAS</td>
<td>[C]</td>
<td>[F]</td>
</tr>
<tr>
<td>AGE &lt; 25</td>
<td>10</td>
<td>0.22%</td>
<td>93</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.64%</td>
<td>98</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.22%</td>
<td>51</td>
<td>22</td>
</tr>
<tr>
<td>RACE BIAS?</td>
<td>10</td>
<td>0.23%</td>
<td>191</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.29%</td>
<td>221</td>
<td>113</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.33%</td>
<td>107</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1.19%</td>
<td>70</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2.46%</td>
<td>32</td>
<td>20</td>
</tr>
<tr>
<td>MIN</td>
<td>0.22%</td>
<td>0.09%</td>
<td>24m 32s</td>
<td>0.12%</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>0.31%</td>
<td>1h 54m 48s</td>
<td>0.99%</td>
<td>1h 54m 48s</td>
</tr>
<tr>
<td>MAX</td>
<td>2.46%</td>
<td>2h 44m 11s</td>
<td>8.33%</td>
<td>2h 44m 11s</td>
</tr>
<tr>
<td>CAUCASIAN PRIORS BIAS?</td>
<td>10</td>
<td>7.92%</td>
<td>237</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2.60%</td>
<td>776</td>
<td>265</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>4.29%</td>
<td>1175</td>
<td>410</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>5.16%</td>
<td>397</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>7.54%</td>
<td>338</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>8.00%</td>
<td>415</td>
<td>103</td>
</tr>
<tr>
<td>MIN</td>
<td>2.60%</td>
<td>24m 14s</td>
<td>4.51%</td>
<td>24m 14s</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>6.08%</td>
<td>1h 49m 42s</td>
<td>6.95%</td>
<td>1h 49m 42s</td>
</tr>
<tr>
<td>MAX</td>
<td>5h 50m 6s</td>
<td>12.56%</td>
<td>5h 50m 6s</td>
<td>5h 26m 55s</td>
</tr>
</tbody>
</table>
### Table 11. Queries on Neural Networks Trained on Fair and Race-Biased Data (ProPublica’s compas Data) — Full Table (SYMBOLIC Domain)

<table>
<thead>
<tr>
<th>QUERY</th>
<th>FAIR DATA</th>
<th>SYMBOLIC</th>
<th>BIASED DATA</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MIN</strong></td>
<td>0.32%</td>
<td>36m 0s</td>
<td>0.12%</td>
<td>1h 14m 0s</td>
</tr>
<tr>
<td><strong>MEDIAN</strong></td>
<td>2.46%</td>
<td>2h 17m 3s</td>
<td>8.50%</td>
<td>3h 34m 50s</td>
</tr>
<tr>
<td><strong>MAX</strong></td>
<td>9.27%</td>
<td>164 49 136</td>
<td>1h 46m 28s</td>
<td>4h 55m 35s</td>
</tr>
<tr>
<td><strong>MALE</strong></td>
<td>10.93%</td>
<td>166 60 127</td>
<td>30m 13s</td>
<td>50m 53s</td>
</tr>
<tr>
<td><strong>AGE BIAS?</strong></td>
<td>10.836%</td>
<td>197 39 135</td>
<td>38m 46s</td>
<td>29m 19s</td>
</tr>
<tr>
<td><strong>RACE BIAS?</strong></td>
<td>10.2.64%</td>
<td>734 170 322</td>
<td>2h 2m 23s</td>
<td>1h 8m 47s</td>
</tr>
<tr>
<td><strong>CAUSAL BIAS</strong></td>
<td>10.5.64%</td>
<td>227 70 159</td>
<td>1h 21m 58s</td>
<td>1h 21m 58s</td>
</tr>
<tr>
<td><strong>PARENT BIAS</strong></td>
<td>10.7.84%</td>
<td>276 61 249</td>
<td>25m 13s</td>
<td>56m 4s</td>
</tr>
<tr>
<td><strong>MIN</strong></td>
<td>2.64%</td>
<td>25m 13s</td>
<td>5.20%</td>
<td>29m 19s</td>
</tr>
<tr>
<td><strong>MEDIAN</strong></td>
<td>6.77%</td>
<td>1h 1m 51s</td>
<td>7.02%</td>
<td>1h 2m 26s</td>
</tr>
<tr>
<td><strong>MAX</strong></td>
<td>8.40%</td>
<td>2h 2m 23s</td>
<td>12.71%</td>
<td>4h 55m 35s</td>
</tr>
<tr>
<td><strong>CAUSAL BIAS</strong></td>
<td>12.2.18%</td>
<td>46 14 39</td>
<td>4h 30m 18s</td>
<td>5h 29m 22s</td>
</tr>
<tr>
<td><strong>PARENT BIAS</strong></td>
<td>14.5.66%</td>
<td>68 34 57</td>
<td>2h 26m 43s</td>
<td>1h 16m 36s</td>
</tr>
<tr>
<td><strong>MIN</strong></td>
<td>2.18%</td>
<td>1h 20m 41s</td>
<td>2.92%</td>
<td>3h 32m 23s</td>
</tr>
<tr>
<td><strong>MEDIAN</strong></td>
<td>2.95%</td>
<td>4h 12m 28s</td>
<td>4.21%</td>
<td>3h 32m 52s</td>
</tr>
<tr>
<td><strong>MAX</strong></td>
<td>5.36%</td>
<td>60h 53m 6s</td>
<td>6.98%</td>
<td>4h 51m 42s</td>
</tr>
<tr>
<td>QUERY</td>
<td>DEEPPOLY</td>
<td>FAIR DATA</td>
<td>BIASED DATA</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>-----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>BIAS</td>
<td>[C]</td>
<td>[F]</td>
</tr>
<tr>
<td>0.25%</td>
<td>10</td>
<td>0.23%</td>
<td>31</td>
<td>18</td>
</tr>
<tr>
<td>0.75%</td>
<td>10</td>
<td>0.75%</td>
<td>33</td>
<td>14</td>
</tr>
<tr>
<td>AGE &lt; 25</td>
<td>10</td>
<td>0.22%</td>
<td>34</td>
<td>17</td>
</tr>
<tr>
<td>RACE BIAS?</td>
<td>10</td>
<td>0.24%</td>
<td>118</td>
<td>28</td>
</tr>
<tr>
<td>0.31%</td>
<td>10</td>
<td>0.31%</td>
<td>117</td>
<td>49</td>
</tr>
<tr>
<td>0.33%</td>
<td>10</td>
<td>0.33%</td>
<td>59</td>
<td>18</td>
</tr>
<tr>
<td>1.36%</td>
<td>10</td>
<td>1.36%</td>
<td>39</td>
<td>17</td>
</tr>
<tr>
<td>2.12%</td>
<td>10</td>
<td>2.12%</td>
<td>33</td>
<td>17</td>
</tr>
<tr>
<td>MIN</td>
<td>0.22%</td>
<td>0.22%</td>
<td>5m 18s</td>
<td>0.12%</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>0.32%</td>
<td>0.32%</td>
<td>47m 16s</td>
<td>0.99%</td>
</tr>
<tr>
<td>MAX</td>
<td>2.12%</td>
<td>2.12%</td>
<td>1h 11m 43s</td>
<td>6.48%</td>
</tr>
</tbody>
</table>

Table 13. Comparison of Different Analysis Configurations (Japanese Credit Screening) — 24 vCPUs