# *DeepGemini*: Verifying Dependency Fairness for Deep Neural Networks

**Xuan Xie**[1], **Fuyuan Zhang**[2*], **Xinwen Hu**[3], **Lei Ma**[1,2,4]
[1]University of Alberta, Canada
[2]Kyushu University, Japan
[3]Hunan Normal University, China
[4] The University of Tokyo, Japan

## Abstract

Deep neural networks (DNNs) have been widely adopted in many decision-making industrial applications. Their fairness issues, i.e., whether there exist unintended biases in the DNN, receive much attention and become critical concerns, which can directly cause negative impacts in our daily life and potentially undermine the fairness of our society, especially with their increasing deployment at an unprecedented speed. Recently, some early attempts have been made to provide fairness assurance of DNNs, such as fairness testing, which aims at finding discriminatory samples empirically, and fairness certification, which develops sound but not complete analysis to certify the fairness of DNNs. Nevertheless, how to formally compute discriminatory samples and fairness scores (i.e., the percentage of fair input space), is still largely uninvestigated. In this paper, we propose *DeepGemini*, a novel fairness formal analysis technique for DNNs, which contains two key components: discriminatory sample discovery and fairness score computation. To uncover discriminatory samples, we encode the fairness of DNNs as safety properties and search for discriminatory samples by means of state-of-the-art verification techniques for DNNs. This reduction enables us to be the first to formally compute discriminatory samples. To compute the fairness score, we develop counterexample guided fairness analysis, which utilizes four heuristics to efficiently approximate a lower bound of fairness score. Extensive experimental evaluations demonstrate the effectiveness and efficiency of *DeepGemini* on commonly-used benchmarks, and *DeepGemini* outperforms state-of-the-art DNN fairness certification approaches in terms of both efficiency and scalability.

## Introduction

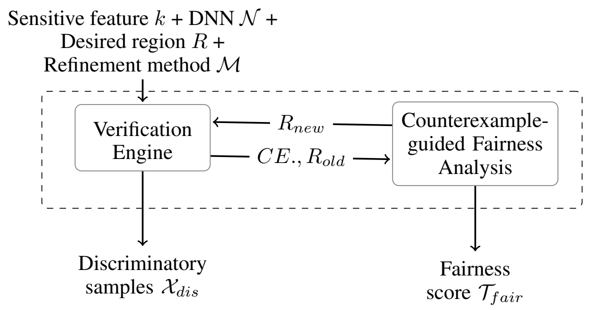
Deep neural networks (DNNs) have become the core components of many social-critical decision-making processes, e.g., credit classification, income prediction (Dua and Graff 2017), and recidivism risk estimation (Angwin, Larson, and Mattu 2018), which can directly impact our daily life. Their continuously rapid growth and widespread deployment also pose critical fairness concerns, which have received much attention recently. For example, in image search, when searching for certain words, such as CEO, there can be a gender bias (Kay, Matuszek, and Munson 2015). Consequently, DNN fairness has been an important requirement and critical property of the software system, with urgent industrial demands for rigorous and systematical analysis.

Up to the present, there have been quite a few fairness criteria used in the literature from different perspectives. For instance, group fairness (Feldman et al. 2015) and equalised odds (Hardt, Price, and Srebro 2016) are the two most common probability-based fairness. In this work, we take a special focus on the important *dependency fairness* (Urban et al. 2020; Galhotra, Brun, and Meliou 2017) of DNNs, which is considered as a stronger notion of group fairness because it imposes fairness on every individual sample. In particular, a DNN satisfies dependency fairness if its output prediction is not influenced by values of sensitive features. Sensitive features, typically, are the features that are considered sensitive to bias, e.g., gender of the client in credit approval prediction. We select this fairness notion, over the other notions, because the found individual discriminatory samples can be verified on the original network, and it is a stronger notion of group fairness, which is amenable to verification.

Although various testing and verification techniques have been designed against dependency fairness, challenges arise when it comes to formally compute discriminatory samples and fairness scores, i.e., the fraction of the region in which no discriminatory samples are contained. For instance, Libra (Urban et al. 2020) is an abstract interpretation based technique to certify dependency fairness with definite guarantees. Nevertheless, it relies on over-approximation of the space, and therefore a concrete discriminatory sample is out of its reach. ADF (Zhang et al. 2020) is a testing technique to search for discriminatory samples based on gradient information. However, this technique cannot give formal guarantees on the desired fairness.

Towards addressing these challenges, we propose a novel technique *DeepGemini*, for formally computing discriminatory samples and fairness scores, which is schematically shown in Figure 1. It mainly consists of two novel key components, *discriminatory sample computation* and *Counterexample Guided Fairness Analysis (CEGFA)*. First, to formally compute discriminatory samples, we propose a simple and effective reduction to reduce dependency fairness to safety properties of DNNs. After reduction, discriminatory samples of DNNs can be viewed as violations of safety properties so that we can use existing safety verification techniques, e.g., techniques based on Satisfiability Modulo The-

[*]Corresponding authors



Sensitive feature $k$ + DNN $\mathcal{N}$ +
Desired region $R$ +
Refinement method $\mathcal{M}$
Verification Engine
$R_{new}$
$CE., R_{old}$
Counterexample-guided Fairness Analysis
Discriminatory samples $\mathcal{X}_{dis}$
Fairness score $\mathcal{T}_{fair}$

Figure 1: Overview of *DeepGemini*.

ories (Katz et al. 2017) or Mixed-Integer Linear Programming (Tjeng, Xiao, and Tedrake 2017), for DNNs to find discriminatory samples. To the best of our knowledge, we are the first to formally compute discriminatory samples for deep neural networks. Second, to compute fairness score, we propose counterexample guided fairness analysis, with iterative region refinement, towards fairness region discovering (Chowdhary 2020). By taking the formally computed counterexamples into consideration, we design four novel iterative region refinement approaches: random based, saliency map based, important neuron based, and $\varepsilon$-greedy based approaches. These novel refinement approaches are shown empirically to be effective in the fairness score computation.

To demonstrate the usefulness of *DeepGemini*, we evaluate its performance on two commonly-used benchmarks, German credit dataset, and COMPAS dataset. The results confirm that *DeepGemini* can discover discriminatory samples effectively and efficiently. Moreover, the effectiveness of diverse fairness region refinement approaches is also demonstrated. Furthermore, *DeepGemini* achieves higher efficiency and more scalability than state-of-the-art fairness certification approaches.

In summary, the key contributions of this paper are three-fold:

- We introduce a simple and effective technique to verify dependency fairness by reducing the fairness verification to the safety verification of DNNs.
- We develop an effective refinement strategy, *Counterexample Guided Fairness Analysis (CEGFA)*, to reduce the verification of large input space into that of smaller input space, and formally compute fairness scores.
- We conduct experimental evaluations on three popular verification datasets, and show the effectiveness of our approach in answering bias queries and computing fairness scores. We also compare the performance of *DeepGemini* with the state-of-the-art verification method.

## Background and Problem Definition

In this section, we first briefly introduce deep neural network verification and fairness criteria in machine learning, which is the background for defining our problem. Following that, we discuss our fairness verification problem.

A neural network $\mathcal{N}$ that classifies inputs on $R^n$ to $m$ classes $\{l_1, ..., l_m\}$ is considered as a function $\mathcal{N} : R^n \to R^m$. Assume that we write $\mathcal{N}_i : R^n \to R$ for the function that returns the evaluation of $\mathcal{N}$ on the $i$th class, i.e., $\mathcal{N}(x) = (\mathcal{N}_1(x), ..., \mathcal{N}_m(x))$. $\mathcal{N}$ classifies an input $x$ into class $l_j$ if and only if $j = \arg\max_i \mathcal{N}_i$. In the rest of the paper, we use $x = (x_1, ..., x_n)$ to denote input variables of $\mathcal{N}$ and write $y = (y_1, ..., y_m)$ for output variables of $\mathcal{N}$.

### Neural Network Verification

Several safety and fairness issues of DNNs have been identified in recent years (Goodfellow, Shlens, and Szegedy 2014; Szegedy et al. 2013), which results in the study on the problem of *Neural Network Verification (NNV)*. NNV (Katz et al. 2017; Gehr et al. 2018; Obermeyer et al. 2019) aims at making sure DNNs run correctly and giving rigorous guarantees to neural network models. Given a DNN model $\mathcal{N}$ and the desired specification $\varphi$, the neural network verifier verifies whether $\mathcal{N}$ satisfies $\varphi$ on all inputs $x$. The specification holds if no counterexample exists. Otherwise, a counterexample, found by the neural network verifier, is returned. Typically, $\varphi$ is of the form:

$$\{\varphi_{pre}(x)\}\ y \leftarrow \mathcal{N}(x)\ \{\varphi_{post}(x, y)\},$$

where $\{\varphi_{pre}(x)\}$ is the precondition over the input of $\mathcal{N}$, $\{\varphi_{post}(x, y)\}$ is the postcondition over the input and the output of $\mathcal{N}$, and $y \leftarrow \mathcal{N}(x)$ refers to the procedure of computing the output of $\mathcal{N}$ for input $x$ and assigning the output value to $y$. The meaning of the above triple is equivalent to the implication: $\forall x, \varphi_{pre}(x) \wedge y \leftarrow \mathcal{N}(x) \implies \varphi_{post}(x, y)$.

Although DNN verification can be quite challenging, recent researches have made some early attempts to design various verification techniques. For instance, Reluplex (Katz et al. 2017) is an SMT solver specialized for neural networks, which is based on the classical simplex algorithm and aims at handling the theory of linear real arithmetic with ReLU constraints. AI2 (Gehr et al. 2018) certifies adversarial robustness by transforming DNN analysis into the framework of abstract interpretation, which over-approximates the computation of a deep neural network on a symbolic set of the infinite set of inputs.

### Fairness Criteria

The discrimination, hidden in the training data or the generalization process, causes the discriminative prediction of machine learning models, and this motivates researchers to develop several fairness criteria to measure the degree of fairness of the models. There are a significant number of fairness metrics that mainly fall into three categories: *individual*, *group*, and *subgroup* (Mehrabi et al. 2019). Individual fairness, such as dependency fairness and counterfactual fairness, requires that the model gives similar predictions to similar individuals (Urban et al. 2020; Kusner et al. 2017). Group fairness, e.g., demographic parity, equalized odds, and equal opportunity, tries to ensure the machine learning model treats different groups of people equally (Hardt, Price, and Srebro 2016; Dwork et al. 2012). Subgroup fairness applies the constraint of group fairness to a subset of the group (Kearns et al. 2018).

In this work, we target at verifying dependency fairness, i.e., the prediction result of a DNN model does not depend on the sensitive input features. Dependency fairness is a stronger notion than the group-based fairness notions (Urban et al. 2020). This specification requires the investigation of full input space, or a fraction of the space. Notice that, in general, this problem is very challenging because of the intractability of the potentially infinite number of inputs exploration.

## Problem Definition

In this work, we focus on DNNs that are used to perform classification tasks in decision-making procedures. Due to the existence of explicit or underlying bias in the DNNs, we are motivated to give rigorous guarantees to the degree of dependency fairness of DNN models. More specifically, our goal is to find discriminatory samples and further quantify the fairness regions.

As a first step towards this goal, we define the notion of *discriminatory samples*. Intuitively speaking, a discriminatory sample is an input such that changing its sensitive features could cause different prediction results. A DNN is *fair* on an input region if there exist no discriminatory samples in that region.

**Definition 1 (Discriminatory Samples).** *Let $\mathcal{N}$ be a neural network and $K$ be the set of indices of sensitive features. An input $x = (x_1, x_2, ..., x_n)$ is a discriminatory sample if there exists $x' = (x'_1, x'_2, ..., x'_n)$, such that*

- $\forall i \notin K : x_i = x'_i$,
- $\exists i \in K$, $x_i \neq x'_i$,
- *and $\mathcal{N}$ classifies $x$ and $x'$ into two different classes.*

In the context of fairness-critical DNN-enabled software, however, it is not enough to identify discriminatory samples only. It is also necessary to know the percentage of the region where there are no discriminatory samples, i.e., fairness score. It serves as a fairness metric for DNN models that reflect how fair the DNN is in the desired region. Intuitively, *fairness score* is the fraction of input space where there is no violation of dependency fairness. For example, consider a DNN $\mathcal{N}$ with a small input space, consisting of only 1000 inputs. Assume that there are only 800 inputs for which changing sensitive features cannot alter their network predictions. Then, the fairness score of $\mathcal{N}$ is 80%.

**Definition 2 (Fairness Score).** *Let $\mathcal{N}$ be a DNN and $\mathcal{S}_{in}$ be the set of its inputs. Assume that $X_K$ is a set of sensitive features. The fairness score of $\mathcal{N}$, denoted $\mathcal{T}_{fair}$, with respect to $X_K$ is the fraction of inputs in $\mathcal{S}_{in}$ for which changing sensitive features cannot change the prediction of $\mathcal{N}$.*

Having defined discriminatory samples and fairness scores, we now define the main problem that we focus in this paper.

**Problem 1 (Fairness Verification).** *Given a DNN $\mathcal{N}$ and the sensitive features, find discriminatory samples and compute the fairness score of $\mathcal{N}$.*

In general, Problem 1 is very challenging as it involves reasoning about an infinite number of inputs. In the following section, we propose a novel technique, named *DeepGemini*, to tackle this problem.

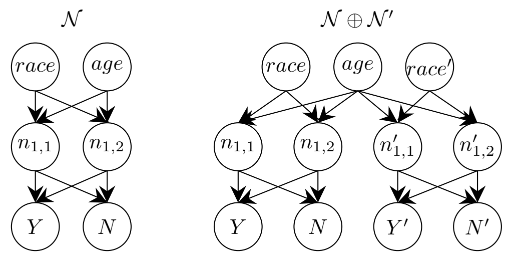


Figure 2: A small illustrative example of DNN $\mathcal{N}$ used for credit approval. $\mathcal{N} \oplus \mathcal{N}'$ is the newly constructed DNN for verification.

## *DeepGemini*: A Dependency Fairness Analysis Framework

In this section, we present our fairness analysis framework, which consists of two key parts. The first part is about how we verify dependency fairness through reducing it to the safety verification problem so that we can leverage the state-of-the-art verification engines, e.g. Reluplex (Katz et al. 2017) or Neurify (Wang et al. 2018). The second part is our counterexample guided fairness analysis, which first analyzes the entire input region and then performs iterative refinement to regions that contain discriminatory samples for further analysis.

### Reducing Fairness Verification to Safety Verification

Our key insight is that we identify that the recent development of safety verification of DNNs could serve as the basis for fairness verification. If we could properly formulate and transform the dependency fairness verification into the similar context of DNN safety verification, our technique could be general to utilize the existing safety verification as the basis to perform fairness verification.

To verify dependency fairness of $\mathcal{N}$ w.r.t. sensitive features, we construct a new neural network to compare the output difference of $\mathcal{N}$ caused by changes in sensitive features.

The new network is constructed by composing $\mathcal{N}$ and an identical copy of $\mathcal{N}$, denoted $\mathcal{N}'$, in parallel, such that

- $\mathcal{N}$ and $\mathcal{N}'$ share the same input values for variables on non-sensitive dimensions, and
- $\mathcal{N}$ and $\mathcal{N}'$ have independent input values for sensitive features.

We denote the newly constructed network as $\mathcal{N} \oplus \mathcal{N}'$. It is easy to see that $\mathcal{N}$ satisfies dependency fairness if and only if for an arbitrary input of $\mathcal{N} \oplus \mathcal{N}'$, $\mathcal{N}$ and $\mathcal{N}'$ always make the same predictions.

Figure 2 shows an example of our construction. The original network $\mathcal{N}$ is used for deciding credit approval. It has two inputs $age$ and $race$, two output variables $yes$ and $no$, and one hidden layer with two nodes. Assume that $race$ is the sensitive feature. In $\mathcal{N} \oplus \mathcal{N}'$, both $\mathcal{N}$ and $\mathcal{N}'$ have the same input $age$, but they have their own variables $race$ and $race'$ for race.

Assume that $X_K = \{x_{k_1}, ..., x_{k_j}\}$ is the set of sensitive features. Then, the network $\mathcal{N} \oplus \mathcal{N}'$ is a function $\mathcal{N} \oplus \mathcal{N}' : R^{n+j} \rightarrow R^{2m}$. We use primed variables for sensitive features and output in $\mathcal{N}'$. Hence, the input and output variables of $\mathcal{N} \oplus \mathcal{N}'$ are denoted as $x_{com} = (x_1, ..., x_n, x'_{k_1}, ..., x'_{k_j})$ and $y_{com} = (y_1, ..., y_m, y'_1, ..., y'_m)$, respectively.

Let $pred_i := \bigwedge_{1 \leq j \leq m, j \neq i}(y_i > y_j)$ and $pred'_i := \bigwedge_{1 \leq j \leq m, j \neq i}(y'_i > y'_j)$ be two predicates, which means that the prediction of network $\mathcal{N}$ (resp. $\mathcal{N}'$) is class $l_i$. We have the following theorem that reduces dependency fairness to safety properties of DNNs.

**Theorem 1 (Soundness and Completeness).** *Let $\mathcal{N}$ be a neural network and $X_K$ be a set of sensitive features. Let $\mathcal{N}'$ be an identical copy of $\mathcal{N}$. We have that $\mathcal{N}$ satisfies dependency fairness if and only if the following specification holds: $\forall x_{com}, y_{com} : y_{com} = \mathcal{N} \oplus \mathcal{N}'(x_{com}) \implies \bigvee_i (pred_i \wedge pred'_i)$.*

## Counterexample Guided Fairness Analysis

For deep neural networks deployed in fairness-critical domains, we are also motivated to compute the percentage of the regions where there are no discriminatory samples that could be easily identified, i.e., fairness score. However, it is not a trivial task, because even a fraction of the global space could contain infinitely many inputs. Therefore, we develop *Counterexample Guided Fairness Analysis (CEGFA)*, shown in Algorithm 1, to discover fair regions by leveraging discriminatory samples.

The inputs of Algorithm 1 are the DNN under verification $\mathcal{N}$, a set of sensitive features $X_K$, the total input space $\mathcal{S}_{in}$, and the iterative region refinement method $\mathcal{M}$. The outputs are the set of discriminatory samples $\mathcal{X}_{dis}$, and the fairness score $\mathcal{T}_{fair}$.

First, we construct $\mathcal{N} \oplus \mathcal{N}'$ to leverage the state-of-the-art safety verification engine to perform fairness verification (Line 1 in Algorithm 1). We initialize the set of discriminatory samples $\mathcal{X}_{dis}$, the region queue $Q$, and the set of fairness region $\mathcal{S}$ (Line 2). The region queue maintains a list of regions to be analysed. It collects regions that are computed by our region splitting methods and possibly contain no discriminatory samples. Computed regions are added to the region queue in a FIFO manner, so that the regions that have not been visited for a long period will be analyzed first. Then, the algorithm goes into the loop of computing discriminatory samples and the fairness score. The head element of $Q$ is dequeued and assigned to $R$ (Line 4). By reducing fairness to safety properties, introduced in Section , we verify dependency fairness of $\mathcal{N}$ on region $R$ using a state-of-the-art safety verification engine (Line 5). If there is no counterexample, i.e. a discriminatory sample, on $R$, we append region $R$ to $\mathcal{S}$ as a fair region (Line 6-7). Otherwise, the discriminatory sample $x$ is appended to $\mathcal{X}_{dis}$. In the case of returning unknown, we skip the step of adding $x$ to $\mathcal{X}_{dis}$ (Line 9-10). The region refinement method $\mathcal{M}$, described in the following subsection, generates new regions by intelligently refining regions that contain discriminatory samples, and the algorithm enqueues new regions to $Q$ (Line 11-12). Based on the size of fair regions in $\mathcal{S}$, we update the fairness score (line 13). Finally, if the running time exceeds the time budget or the fairness score is one, the algorithm returns $\mathcal{X}_{dis}$ and $\mathcal{T}_{fair}$ (Line 14-15).

**Lemma 1** $\mathcal{T}_{fair}$, computed by the counterexample guided fairness analysis, is an approximation of the lower bound of the ground-truth fairness score.

*Proof.* Starting from the full input space, our fairness analysis bisects the space and performs verification on each subspace. If the subspace is proven to be fair, it will be added to the computation of $\mathcal{T}_{fair}$. On the other hand, if a discriminatory sample is found, the subspace will be further split for analysis. Therefore, $\mathcal{T}_{fair}$ is indeed an approximation of the lower bound of the ground-truth fairness score. □

Algorithm 1 incrementally computes an under-approximation of the fairness score, meaning that at each iteration the computed score is less than or equal to the real fairness score. In this way, one can stop the algorithm at any point in time and use the fairness score result.

### Iterative Region Refinement

When $\mathcal{N}$ is not fair on region $R$, we further partition $R$ and verify whether $\mathcal{N}$ is fair on some of the subregions of $R$. As mentioned in Section , we partition regions that contain discriminatory samples via *iterative region refinement*. More specifically, we design four kinds of region refinement approaches: random based, saliency map based, important neuron based, and $\varepsilon$-greedy based approaches.

- ***Random based.*** This method chooses a non-sensitive input dimension randomly and bisects the original regions into two subregions. For instance, for a region with one dimension $[l, u]$, the bisection generates two new subregions: $\left[l, \frac{l+u}{2}\right]$ and $\left[\frac{l+u}{2}, u\right]$.
- ***Saliency Map based (SM).*** This method identifies the non-sensitive input dimension that is most influential to the network's output, and split the input from the selected dimension. In particular, to determine the influence of input neurons, we utilize the idea of *adversarial saliency maps* (Papernot et al. 2016), which indicates the input features that an adversary should perturb to achieve the goal to change the prediction of the DNN. The input neuron that is the most adversarial in the saliency map is selected as the refinement dimension.
- ***Important Neuron based (IN).*** This approach includes the following two key steps. (1) It conducts the *important neuron analysis* to recognize the most *important* neuron in the hidden layers. More specifically, the goal of *important neuron analysis* (Gerasimou et al. 2020) is to identify neurons that have key contributions to the final decision making process. (2) It computes the gradient of the most important neuron with respect to the input of the network. The non-sensitive input dimension that has the largest gradient is prioritized to be split, to produce new regions. A

Algorithm 1: CounterExample Guided Fairness Anlaysis

**Input**: A DNN $\mathcal{N}$, a set of sensitive feature $X_K$, the input space $\mathcal{S}_{in}$, and the region splitting method $\mathcal{M}$
**Output**: A set of discriminatory samples $\mathcal{X}_{dis}$, and the fairness score $\mathcal{T}_{fair}$

```
1:  Constuct $\mathcal{N} \oplus \mathcal{N}'$, where $\mathcal{N}'$ is a copy of $\mathcal{N}$;
2:  Let $\mathcal{X}_{dis} \leftarrow \emptyset$, $Q \leftarrow \{\mathcal{S}_{in}\}$, $\mathcal{S} \leftarrow \emptyset$;
3:  while True do
4:      $R = \textsc{DeQueue}(Q)$;
5:      $x = \textsc{VerifyFairness}(\mathcal{N} \oplus \mathcal{N}', X_K, R)$;
6:      if $x = \emptyset$ then
7:          $\mathcal{S} = \mathcal{S} \cup R$;
8:      else
9:          if $x \neq$ unknown then
10:             $\mathcal{X}_{dis} = \mathcal{X}_{dis} \cup x$;
11:         end if
12:         $R_{new} = \mathcal{M}(R)$;
13:         $Q = \textsc{EnQueue}(Q, R_{new})$;
14:     end if
15:     $\mathcal{T}_{fair} = \frac{\Sigma_{\mathcal{R} \in \mathcal{S}} |\mathcal{R}|}{|\mathcal{S}_{in}|}$;
16:     if timeout or $\mathcal{T}_{fair} = 1$ then
17:         return $\mathcal{X}_{dis}$, $\mathcal{T}_{fair}$;
18:     end if
19: end while
```

greater gradient value of the input node signifies it has a larger effect on the important neurons.

- ***$\varepsilon$-greedy based.*** Inspired by the *exploration and exploitation* strategy in reinforcement learning, we design $\varepsilon$-greedy based approach, which mixes two refinement strategies, e.g., *a* and *b*, and assigns probability $\varepsilon$ to one of the methods, say *a*, and $1 - \varepsilon$ to the other, say *b*. Here, $\varepsilon$ is a hyperparameter that indicates how frequently the corresponding method is chosen. We primarily show the effect of *SM-random* and *IN-random* in Section .

**Remark 1** Although we can compute the fairness score of training data, due to the generalization ability of DNNs and their non-deterministic training process, the ground truth fairness score of DNNs is different from the fairness score of their training data. In fact, the ground truth fairness score of neural networks is typically difficult, if not impossible, to acquire. The fairness score computed by Algorithm 1 is an under-approximation of the ground truth fairness score. The precision of Algorithm 1 improves as the number of iterations of region refinement increases. Theoretically, given enough time, the computed fairness score would be an asymptotic solution to the ground truth, as shown by Lemma 1.

**Remark 2** Our approach can be applied to judge dependency fairness of DNNs in practice. For instance, for DNNs used in credit approval, banks could require that the fairness score of DNNs should be higher than $95\%$. Moreover, not only the fairness score should be considered, but the found bias should also be utilized as a reference to check the severity of bias.

| Model | Dataset | Time | #*Ce.* | #Fair |
|---|---|---|---|---|
| **GERMAN-4*10** | `bias` | 0.42 | 7 | 3 |
| | `fair` | 0.64 | 3 | 7 |
| **GERMAN-4*20** | `bias` | 0.34 | 9 | 1 |
| | `fair` | 2.54 | 3 | 7 |
| **GERMAN-4*30** | `bias` | 1.23 | 10 | 0 |
| | `fair` | 14.65 | 5 | 5 |
| **COMPAS-4*10** | `bias` | 1.91 | 8 | 2 |
| | `fair` | 0.16 | 1 | 9 |
| **COMPAS-4*20** | `bias` | 37.0 | 10 | 0 |
| | `fair` | 69.68 | 3 | 7 |
| **COMPAS-4*30** | `bias` | 225.96 | 10 | 0 |
| | `fair` | 213.34 | 2 | 8 |

Table 1: Experiment results for RQ1. Time is the average verification time in seconds. #*Ce.* is the number of models in which a counterexample is found. #Fair is the number of models that are verified as fair.

## Experiment

In this section, we perform an evaluation of *DeepGemini* to demonstrate its usefulness. In particular, we mainly investigate the following research questions:

- **RQ1:** Is *DeepGemini* able to answer bias queries?
- **RQ2:** Can *DeepGemini* compute fair regions?
- **RQ3:** Is *DeepGemini* more scalable and efficient than the state-of-the-art fairness verification method?

### Experimental Setup

**Implementation and Hardware Configurations** We implemented *DeepGemini* in Python, supporting Pytorch 1.0.0. and built on top of Marabou (Katz et al. 2019), which is an SMT-based tool specialized for neural network verification. We made our source code available online in a repository.[1] All experiments were run on a server with Intel(R) Core(TM) i9-10940X CPU @ 3.30GHz Processor, 28 CPUs, 62G RAM, and two NVIDIA RTX A6000.

**Datasets and Sensitive Features** We choose two widely used fairness benchmark datasets for evaluation: (1) The *UCI German-credit* dataset (Dua and Graff 2017), which assigns credits based on the information of the clients, e.g., age and gender, with 3,356 training samples. (2) The *ProPublica's COMPAS recidivism* dataset (Angwin, Larson, and Mattu 2018). It measures the recidivism-risk score, i.e., how likely the offenders will reoffend, of the offenders. This dataset contains 6,762 training samples. We use the fair and biased version, adapted by Urban et al. (Urban et al. 2020). The fairness score of the fair and biased version are $100\%$ and $80\%$, respectively. The sensitive features are *age* and *race* for German dataset and COMPAS dataset, respectively.

**DNN Model Architecture** The model architectures range from four hidden layers with 10 neurons each to five hidden layers with 50 neurons each. We show the corresponding model architectures in each RQ. For each model size,

[1]https://github.com/LebronX/DeepGemini-public

we typically train ten neural networks for verification procedures and report the average result, to counteract the non-deterministic and randomized training process of DNNs.

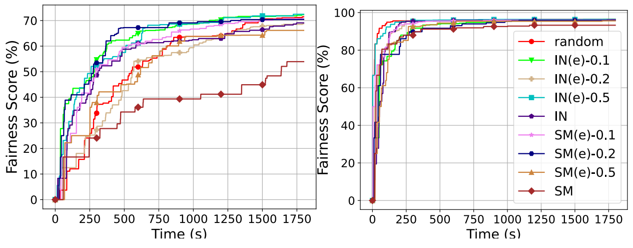
**Baseline Comparison** In RQ3, we choose Libra (Urban et al. 2020) for baseline comparison. There is also work on analyzing dependency fairness of deep neural networks (Bastani, Zhang, and Solar-Lezama 2019; Albarghouthi et al. 2017). However, their work is either on providing probabilistic guarantees (Bastani, Zhang, and Solar-Lezama 2019), or only scale to DNNs with two hidden neurons (Albarghouthi et al. 2017).

## Experimental Results

**RQ1 (Answering Bias Queries)** In this RQ, we investigate whether our technique can effectively answer bias queries for the given model, and we use the German credit dataset and the COMPAS dataset for evaluation. The models we select have four hidden layers and the following number of neurons per layer: 10, 20, and 30. For the German credit dataset, we ask the following bias queries: $Q_1^G$: Is there bias with respect to age for people holding a house? For COMPAS dataset, we ask: $Q_1^C$: Is there bias with respect to race for females younger than 25?

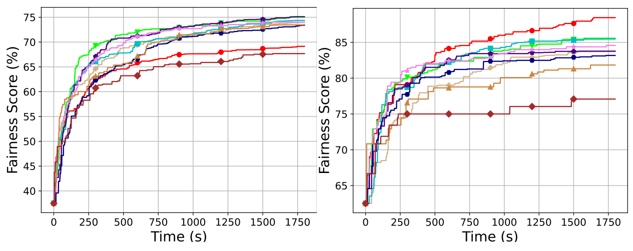
Table 1 summarizes the experimental results. For each model trained on different versions of the dataset, it reports the average verification time, the number of models, in which a counterexample is discovered, and the number of models that are verified to be fair. Our in-depth analysis reveals the following observations from the results. First, *DeepGemini* can efficiently capture the underlying bias in the models. On the German credit dataset, the query answering time for models trained on the biased dataset is shorter than that for models trained on the fair dataset. For instance, the average query answering time for $Q_1^G$ in the fair version is 5.94 seconds, whereas the time for $Q_1^G$ in the biased version is 0.66 seconds. However, on the COMPAS dataset, the query answering time for models trained on both versions are close. For example, the average query answering time for $Q_1^C$ in the fair version is 94.39 seconds, whereas the corresponding time in the biased version is 88.29 seconds. Second, it takes more time for querying as the size of the networks increase. The average querying time for COMPAS-4*20 in fair version is 69.68 seconds, and the average querying time for COMPAS-4*30 in fair version is 213.34 seconds. Note that there is no timeout case on two benchmarks. The backend SMT-based verification engine, although developed specifically for the verification of DNNs, is based on an NP-complete algorithm (Katz et al. 2017). As the size of models increases, the verification time increases dramatically on models for the COMPAS dataset. Third, there are typically more discriminatory samples hidden in the DNNs trained on the biased dataset, where 9.0 counterexamples on average are found, than the ones trained on the fair dataset, where 2.8 counterexamples are discovered.

**Answer to RQ1:** *DeepGemini* is effective and efficient in terms of answering bias queries, which indicates its usefulness when applying it in practice.



(a) BIAS (b) FAIR

Figure 3: Fairness scores computed on the German dataset



(a) BIAS (b) FAIR

Figure 4: Fairness score computed on COMPAS dataset

**RQ2 (Computing Fairness Score)** In this RQ, we study the effectiveness of the counterexample guided fairness analysis, by evaluating on the COMPAS dataset and the German credit dataset (both with the fair and the biased version). The models we select for both datasets have four hidden layers and 15 neurons per layer. The query for the COMPAS dataset is: *Is there bias with respect to race?* The query for the German credit dataset is: *Is there bias with respect to age?* The time budget for computing the fairness score in all the experiments is 1800 seconds.

Figure 3 and 4 show the result of the fairness analysis. The x-axis is the time in second, and the y-axis is the fairness score. We mainly compare random, pure important neuron-based (IN), pure saliency map (SM) and the $\varepsilon-$greedy method.

We make the following observations. First, it is clear that the fairness score increases over time, which indicates the effectiveness of our iterative region refinement technique. Second, for both datasets, models trained on the fair version have higher fairness scores than models trained on the biased version. This is consistent with our expectations as models trained on biased datasets would contain more discriminatory samples. Third, sometimes the random approach has a better performance than other region refinement approaches, e.g., in Figure 3(b). We also noticed that in Figure 4(b), the random approach is one of the best among all the compared approaches.

The performance of fair analysis varies across different region refinement strategies. Our experimental results show that no single refinement method can outperform all other refinement methods. In practice, we would recommend users to try the random method first, as it does not involve parameter tuning and has very good performance. If the result is not satisfying, we recommend users tune the parameters of

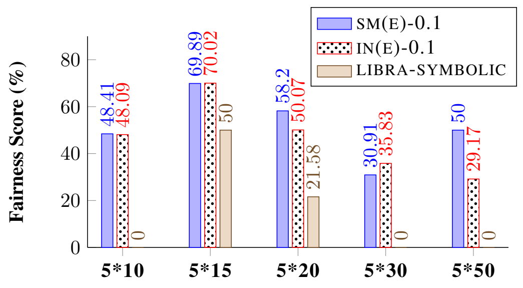


Figure 5: Comparing the performance of *DeepGemini* to Libra in the COMPAS dataset.

the $\varepsilon$-greedy approach to obtain better results if possible.

**Answer to RQ2:** The experimental result demonstrates that CEGFA is effective and efficient in computing fairness scores. In general, there is no best iterative region refinement method over all heuristics.

**RQ3 (Comparison with Libra)** In this RQ, we compare the performance of *DeepGemini* with the abstract interpretation based verification technique Libra (Urban et al. 2020), in the COMPAS dataset. We use the default parameters in Libra, i.e., symbolic as abstract domains, four as the number of tolerated disjunctions, and 0.25 as the smallest partition for each dimension. The DNN models, in the COMPAS dataset, have five hidden layers and the following number of neurons per layer: 10, 15, 20, 30, and 50. We train four neural networks for each model size for the experiment. The query for the COMPAS dataset is: *Is there bias with respect to race?* The verification time budget is 1,800 seconds.

Figure 5 summarizes the obtained results. Overall, *DeepGemini* can efficiently compute fairness scores in small and big networks. The average computed fairness score when using saliency map mixed with random approach is $51.48\%$. However, the corresponding average fairness score computed by Libra is $26.79\%$. Our in-depth analysis finds that this is because Libra spends most of the time in its pre-analysis period, which is designed for analyzing abstract activation patterns, and thus cannot progress for the fairness verification procedure. The data shows that *DeepGemini* can analyze the fairness region efficiently, which indicates its usefulness in practical scenarios.

**Answer to RQ3:** *DeepGemini* exhibits better scalability and efficiency in verifying dependency fairness than the state-of-the-art verification technique Libra.

## Related Work

**Verification of Fairness.** Libra (Urban et al. 2020) is an abstract interpretation based approach to certifying dependency fairness for DNNs with ReLU activation function. Libra performs backward analysis by analyzing the output of the network and acquires the abstraction of the input space to conduct the intersection check. It leverages the abstract activation pattern to reduce the intractability of the explosion of ReLU analysis and several abstract domains, e.g., box and symbolic domains, are incorporated. As shown in the experiment, for a small network that contains 150 neurons in the hidden layers, Libra cannot finish fairness verification within 30 minutes, whereas *DeepGemini* can perform efficient verification, which demonstrates the scalability and efficiency of our technique.

For other fairness criteria, Justicia (Ghosh, Basu, and Meel 2021) is a stochastic satisfiability framework to verify multiple fairness criteria, such as disparate impact, statistical parity, and equalized odds. Sun et al. (Sun et al. 2021) propose a technique for verifying group fairness, which learns a Discrete-Time Markov Chain by sampling with the Probably Approximate Correctness guarantee and then performs probabilistic model checking, e.g., probabilistic reachability analysis, on the learned surrogate model.

**Formal Analysis of Deep Neural Networks.** Numerous techniques for giving rigorous guarantees to DNNs has been developed. Reluplex (Katz et al. 2017) and AI2 (Gehr et al. 2018) are two early steps on verifying safety and robustness of DNNs. Recently, bound propagation based verifiers on GPU has recently become the main trend in this field (Zhang et al. 2018; Wang et al. 2021; Ferrari et al. 2022). Other work also considers performing model-based analysis on recurrent neural networks (Khmelnitsky et al. 2021; Zhang et al. 2021), or verifying neuro-symbolic property (Xie, Kersting, and Neider 2022).

The reduction to safety proposed in the paper, composing the neural network with an identical copy, is also called *self-composition* (Clarkson and Schneider 2010) and a similar idea is used for verifying the monotonicity properties of DNN models for predicting future blood glucose levels (Kushner, Sankaranarayanan, and Breton 2020). Our work focus on verifying dependency fairness, which can be considered as a type of global robustness, as it requires the DNN to be robust with respect to the sensitive features.

## Conclusion

In this work, we propose *DeepGemini*, a novel framework for formally computing discriminatory samples and fairness scores. We formulate and transform the dependency fairness verification to the safety specification problem, which enables us to leverage existing safety verification techniques to formally compute discriminatory samples. We also propose counterexample-guided fairness analysis, which is based on four iterative region refinement techniques: random based, saliency map based, important neuron based, and $\varepsilon-$greedy based approaches. The empirical evaluation of our approach demonstrates that we are able to answer bias queries, and perform scalable and efficient fairness computation. For future work, we plan to study how to leverage the computed discriminatory samples in the retraining process in order to enhance the fairness of DNNs with rigorous formal analysis.

## Acknowledgments

This work was supported in part by Canada First Research Excellence Fund as part of the University of Alberta's Future Energy Systems research initiative, Canada CIFAR AI Chairs Program, Amii RAP program, the Natural Sciences and Engineering Research Council of Canada (NSERC No.RGPIN-2021-02549, No.RGPAS-2021-00034, No.DGECR-2021-00019), as well as JSPS KAKENHI Grant No.JP20H04168, No.JP21H04877, JST-Mirai Program Grant No.JPMJMI20B8.

## References

Albarghouthi, A.; D'Antoni, L.; Drews, S.; and Nori, A. V. 2017. Fairsquare: probabilistic verification of program fairness. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA): 1–30.

Angwin, J.; Larson, J.; and Mattu, L., S.and Kirchner. 2018. Machine bias risk assessments in criminal sentencing. In *ProPublica*.

Bastani, O.; Zhang, X.; and Solar-Lezama, A. 2019. Probabilistic verification of fairness properties via concentration. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA): 1–27.

Chowdhary, K. 2020. Natural language processing. *Fundamentals of artificial intelligence*, 603–649.

Clarkson, M. R.; and Schneider, F. B. 2010. Hyperproperties. *Journal of Computer Security*, 18(6): 1157–1210.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. In *URL http://archive.ics.uci.edu/ml.*

Dwork, C.; Hardt, M.; Pitassi, T.; Reingold, O.; and Zemel, R. S. 2012. Fairness through awareness. In *Innovations in Theoretical Computer Science 2012*, 214–226. ACM.

Feldman, M.; Friedler, S. A.; Moeller, J.; Scheidegger, C.; and Venkatasubramanian, S. 2015. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 259–268.

Ferrari, C.; Muller, M. N.; Jovanovic, N.; and Vechev, M. 2022. Complete verification via multi-neuron relaxation guided branch-and-bound. *arXiv preprint arXiv:2205.00263*.

Galhotra, S.; Brun, Y.; and Meliou, A. 2017. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017*, 498–510. ACM.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. T. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *2018 IEEE Symposium on Security and Privacy, SP 2018*, 3–18.

Gerasimou, S.; Eniser, H. F.; Sen, A.; and Çakan, A. 2020. Importance-driven deep learning system testing. In *ICSE '20: 42nd International Conference on Software Engineering, Companion Volume*, 322–323. ACM.

Ghosh, B.; Basu, D.; and Meel, K. S. 2021. Justicia: A Stochastic SAT Approach to Formally Verify Fairness. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, 7554–7563. AAAI Press.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Hardt, M.; Price, E.; and Srebro, N. 2016. Equality of Opportunity in Supervised Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, 3315–3323.

Katz, G.; Barrett, C. W.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Computer Aided Verification - 29th International Conference, CAV 2017*, Lecture Notes in Computer Science. Springer.

Katz, G.; Huang, D. A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljic, A.; Dill, D. L.; Kochenderfer, M. J.; and Barrett, C. W. 2019. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In *Computer Aided Verification - 31st International Conference, CAV 2019*, Lecture Notes in Computer Science, 443–452. Springer.

Kay, M.; Matuszek, C.; and Munson, S. A. 2015. Unequal Representation and Gender Stereotypes in Image Search Results for Occupations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015*, 3819–3828. ACM.

Kearns, M. J.; Neel, S.; Roth, A.; and Wu, Z. S. 2018. Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018 10-15, 2018*, Proceedings of Machine Learning Research, 2569–2577. PMLR.

Khmelnitsky, I.; Neider, D.; Roy, R.; Xie, X.; Barbot, B.; Bollig, B.; Finkel, A.; Haddad, S.; Leucker, M.; and Ye, L. 2021. Property-directed verification and robustness certification of recurrent neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, 364–380. Springer.

Kushner, T.; Sankaranarayanan, S.; and Breton, M. 2020. Conformance verification for neural network models of glucose-insulin dynamics. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 1–12.

Kusner, M. J.; Loftus, J. R.; Russell, C.; and Silva, R. 2017. Counterfactual Fairness. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017*, 4066–4076.

Mehrabi, N.; Morstatter, F.; Saxena, N.; Lerman, K.; and Galstyan, A. 2019. A Survey on Bias and Fairness in Machine Learning. *CoRR*, abs/1908.09635.

Obermeyer, Z.; Powers, B.; Vogeli, C.; and Mullainathan, S. 2019. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464): 447–453.

Papernot, N.; McDaniel, P. D.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016*, 372–387. IEEE.

Sun, B.; Sun, J.; Dai, T.; and Zhang, L. 2021. Probabilistic Verification of Neural Networks Against Group Fairness. In *Formal Methods - 24th International Symposium, FM 2021*. Springer.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tjeng, V.; Xiao, K.; and Tedrake, R. 2017. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*.

Urban, C.; Christakis, M.; Wüstholz, V.; and Zhang, F. 2020. Perfectly parallel fairness certification of neural networks. *Proc. ACM Program. Lang.*, 4(OOPSLA): 185:1–185:30.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018. Efficient Formal Safety Analysis of Neural Networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 6369–6379.

Wang, S.; Zhang, H.; Xu, K.; Lin, X.; Jana, S.; Hsieh, C.-J.; and Kolter, J. Z. 2021. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34: 29909–29921.

Xie, X.; Kersting, K.; and Neider, D. 2022. Neuro-Symbolic Verification of Deep Neural Networks. *arXiv preprint arXiv:2203.00938*.

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31.

Zhang, P.; Wang, J.; Sun, J.; Dong, G.; Wang, X.; Wang, X.; Dong, J. S.; and Dai, T. 2020. White-box fairness testing through adversarial sampling. In *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, 949–960. ACM.

Zhang, X.; Du, X.; Xie, X.; Ma, L.; Liu, Y.; and Sun, M. 2021. Decision-Guided Weighted Automata Extraction from Recurrent Neural Networks. In *AAAI*, 11699–11707.