Erasmus University Rotterdam

Bachelor Thesis BSc^2 in Econometrics and Economics

A New Metric for Verifying Individual Fairness in Deep Neural Networks

Author

S.J. (Stein) Dijkstra (467846)

Supervisor

B.T.C. (Bart) van Rossum

Second assessor

dr. T.A.B. (Twan) Dollevoet

July 3, 2021

Abstract

In this paper the existing literature on Mixed Integer Linear Programming (MILP) problems for Deep Neural Networks (DNN) is expanded. The formulation by Fishetti and Jo (2018) is replicated and similar results are obtained. Furthermore, this formulation is expanded for non-continuous features, such as binary, ordered categorical, and unordered categorical features. Additionally, a metric for individual fairness, called a weakly unbiased estimator, is introduced. It is shown that this metric can be calculated using the MILP model for a fraud detection DNN. Lastly, numerical results and computational results regarding the newly introduced metric are presented for a DNN trained on an imbalanced dataset regarding car insurance fraud.

The views stated in this thesis are those of the author and not necessarily those of the supervisor, second assessor, Erasmus School of Economics or Erasmus University Rotterdam

Contents

1 Introduction

Deep Neural Networks (DNN) are some of the most popular machine learning algorithms. In essence, neural networks try to emulate a human brain such that it can learn from previous observations. This is done by modeling layers of neurons using non-linear functions. Even with a few layers, the model can capture complex behaviors. Neural networks have a wide range of applications varying from self-driving cars to fraud classification.

However, one of the main concerns about DNNs is their lack of interpretability and their black-box nature. In particular, it is difficult to verify if a model is unbiased. For example, the municipality of Rotterdam has been heavily criticized on the fraud risk indication algorithm "Systeem Risico Indicatie" (SyRI). There has been a growth of papers that analyze fairness in machine learning models. It is believed that determining fairness could increase the trust in machine learning methods (Ribeiro et al., 2016).

Therefore in this paper, fairness in machine learning method is further investigated. In particular, the Mixed Integer Linear Programming (MILP) approach for Deep Neural Networks (DNN) by Fishetti and Jo (2018) is applied to the MNIST dataset. This approach seems promising as it can create provably optimal solutions in little computational time. Their methodology is expanded to verify individual fairness in a classification network that detects credit card fraud.

We thus want to answer the following research questions: Can the results from Fishetti and Jo (2018) be replicated? Secondly, can the MILP model be applied to a DNN with categorical features? The third research question in this paper is if it is possible to apply the MILP formulation to study fairness in machine learning models. Finally, we try to determine the computational times for the fairness model and how they compare to the original model.

This paper is scientifically relevant since it will add to the literature on MILP formulations for neural networks. Furthermore, it will also add a new model to the literature on fairness for machine learning models. The paper is societally relevant since it can help assist in making Neural Networks more robust as well as possibly checking internal biases in classification networks.

The paper is structured as follows: In section 2 the existing literature on machine learning, the MILP model, and fairness is explored. Section 3 explains the methodology of Deep Neural Network, the MILP as well as the methodology on individual fairness. Section 4 describes the data that is used. In section 5 we present the results of the experiments. Then, in section 6 the conclusion is discussed, and lastly in section 7 our recommendations for further research are given.

2 Literature Review

First, literature on the training and application of Neural Networks is discussed. Then, the literature review of the MILP formulation for DNN is presented. Lastly, literature on fairness is discussed based on an economic and mathematical viewpoint.

2.1 Deep Neural Networks

Machine learning is a field in optimization that tries to learn from past experiences in complex situations. In particular, it finds application in operation research, where machine learning algorithms are used to create accurate predictions to assist in strategic management decisions (Bertsimas & Kallus, 2014).

A popular type of machine learning algorithm that will be investigated in this paper are Deep Neural Networks (DNN), also known as Artificial Neural Networks. DNNs have been studied since the eighties, but due to computational complexity, their development was hindered. From around the year 2010 DNNs it gained more momentum as computer power increased, data became widely available, and solution methods improved (Gambella et al., 2021). Nowadays, it has a wide range of applications such as in image processing, speech processing, and other classification tasks (Cheng et al., 2017; Serra et al., 2018).

DNNs are models that have a structure similar to a human neural network. A DNN consists of multiple layers each with a certain number of nodes called neurons. They are after human neurons since the output is dependent on the output of previous neurons (D. Anderson & McNeill, 1992). Within a DNN the following layers can be distinguished, the input layer, the hidden layers, and the output layer. These layers consist of nodes that are activated by some non-linear function. Common functions for nodes are the rectified linear unit (ReLU), tan^{-1}, and sigmoid function. (Cheng et al., 2017)

To optimize a network, the objective value, also called loss, is minimized. Common loss functions are the sum-of-squared errors for continuous output variables and cross-entropy for categorical data. Back-propagation is used to train the model, commonly by means of stochastic gradient descent or by momentum methods (Gambella et al., 2021).

An important step when creating a DNN is data preprocessing. For almost all applications, the obtained features first have to be scaled (Koprinkova & Petrova, 1999). This ensures that all input features are weighted equally and they reduce the order of magnitude of output from the nodes.

A second important part of data preprocessing is transforming (non-numerical) categorical data (Hancock & Khoshgoftaar, 2020). DNNs rely on numerical input features, thus the features should be replaced by numerical labels. For unordered categorical features, these features could then be transformed using one-hot encoding (Hancock & Khoshgoftaar, 2020).

Another common issue for data is imbalanced data. This means that the distribution of the target label is unequal. This can be mitigated using oversampling of unbalanced class, undersampling, or weights (Qiu et al., 2016).

2.2 Mixed Integer Linear Programming for DNNs

Deep Neural Networks can be represented by a 0-1 Mixed-Integer Linear Programming (MILP) formulation (Cheng et al., 2017; Fishetti & Jo, 2018). However, these formulations can not be used to train DNNs (Fishetti & Jo, 2018). Instead, their main advantages are validating and qualifying DNNs. This can be done by calculating the minimum distance to an adversarial example (Tjeng et al., 2017). An adversarial example is an input for which the DNN outputs a misclassification. Thus, the minimum distance to an adversarial example tell us how large of a change is needed for a misclassification to occur. The adversarial examples can be used as new inputs to reduce overfitting. This area of machine learning is commonly called Adversarial Learning(Szegedy et al., 2013). Additionally, the MILP can be used to find a combination of input variables can be determined such that a certain output variable is maximized (Fishetti & Jo, 2018).

Most of the literature regarding the MILP formulation is related to reducing the computational complexity of the formulation. The reason that the model is quite complex relates to the usage of the non-linear ReLU function (Cheng et al., 2017; Fishetti & Jo, 2018). Most commonly, the ReLU function is modeled using a set of big M constraints. However, these constraints result in poor computational results when using state-of-the-art commercial MILP solvers. Therefore, algorithms were developed that bound the big M constraints and this leads to significant improvement of computational time (Cheng et al., 2017; Fishetti & Jo, 2018). A different approach is to model the ReLU functions without using big M constraints (R. Anderson et al., 2020).

Other algorithms for Adversarial Learning are often based on min-max approaches. However, this often is not viable in practical applications (Gambella et al., 2021). Additionally, adversarial examples are often estimated using heuristic approaches, see Gambella et al. (2021) for an overview. On the other hand, the MILP formulation can be used to create provably optimal adversarial examples (Fishetti & Jo, 2018).

2.3 Fairness in DNNs

2.3.1 Fairness in decision making

In the context of decision-making fairness is mainly about avoiding biases. Mehrabi et al. (2019) formalized this as "fairness is the absence of any prejudice or favoritism toward an individual or a group based on their inherent or acquired characteristics". In essence, it states that all individuals and groups should be evaluated based on the same premises.

In machine learning, there are two main sources of unfairness (Mehrabi et al., 2019). One source of unfairness is related to the data collection process. Common problems are representation bias, where the data is imbalanced. This issue is commonly found in fraud detection datasets(Wei et al., 2013). Another bias in data is sampling bias. Sampling bias is caused by an unequal sampling of different subgroups. An example of this is the underrepresentation of people of color in face recognition datasets(Bacchini & Lorusso, 2019). The third type of bias is historical bias. This type of bias is caused by pre-existing biases in society.

The second source of unfairness is related to the classifications of the algorithm (Mehrabi et al., 2019). Unfair classifications can be described using two metrics, group fairness and individual fairness.

Group fairness is often represented by some equality of probabilities. Examples are equalized opportunity which states that the probability that somebody is classified as 1 is equal between certain groups. A different type of group fairness is fairness through unawareness. This definition states that an algorithm is fair as long as it does not use discriminatory features (Dwork et al., 2012).

Individual fairness can be defined as similar individuals should be treated similarly (Chouldechova & Roth, 2018). The main benefit of this approach over group fairness is that it ensures that each individual is classified fairly instead of that individuals are fairly classified on average. However, the main issue is that individual fairness is based on strong assumptions. For example, the assumption is made that there exist some distance functions between individuals. However, this is often done without substantiation (Dwork et al., 2012).

2.3.2 Mathematics of fairness

One of the difficulties of defining fairness is to present it in a mathematical framework. The literature consists of articles of different types. In the paper of Gajane and Pechenizkiy (2017) basic mathematical descriptions of different types of fairness are presented. In particular, they ingrain their mathematical definitions with the social sciences literature and present an excellent discussion about the different types of fairness. The more recent work of Chouldechova and Roth (2018) surveys what type of methods are currently being developed and how and which definitions are used. They find that both individual fairness and group fairness are still being developed and show how the literature tries to combine them. In the paper of Dwork et al. (2012), the relation between group fairness and individual fairness is studied. They define individual fairness similarly to the definition in Gajane and Pechenizkiy (2017), but their mathematical formulation is different. Whereas Gajane and Pechenizkiy (2017) define individual fairness using distance between outcomes, Dwork et al. (2012) define individual fairness using the difference between output probability density functions. Although individual fairness is seen as a good metric for fairness, it assumes that there exists a distance metric between individuals (Chouldechova & Roth, 2018; Dwork et al., 2012; Gajane & Pechenizkiy, 2017). However, it is uncertain if this distance function exists for all problems.

One of the approaches to study fairness is by estimating the prediction sensitivity. Gajane and Pechenizkiy (2017) present a methodology that uses the fact that the derivative of the loss function of a neural network can be easily computed. Using these results they define a method calculating the partial derivative with respect to some feature. These results can lead to a better interpretation of why certain features affect the output.

A different approach of studying fairness is by trying to verify neural networks. This method was already proposed by Dwork et al. (2012) as a promising area. In the paper of John et al. (2020) a meta-algorithm is proposed to verify machine learning models. They were able to successfully apply it to polynomial and RBF kernel models. In their paper, they also define criteria that should hold for a good verifier.

2.4 Contributions

Although most of the concepts in the paper have previously been studied in the literature, to the best of our knowledge applying the MILP formulation by Fishetti and Jo (2018) to create a fairness metric is a new concept. Most similarly is the work of Tjeng et al. (2017) who also studies a similar metric. However, in their paper, no link is made to fairness and their methodology is not directly applicable to DNNs. Additionally, methods in the paper of John et al. (2020) are similar to the methodology in this paper. Nevertheless, their method is not applied to a DNN, and their metric is defined slightly different.

3 Methodology

This section is split into three subsections. First, the methodology on Deep Neural Networks and the estimation is explained. Then, the formulation of the MILP for DNNs is presented. Finally, the methodology on fairness and adaptations to the MILP model is given.

3.1 Deep Neural Networks

A DNN consists of a sequence of $K + 1$ layers, which can be labeled as $0, \dots, K$. In this network the layer 0 corresponds to the input layer, layer K is the output layer, and layers $1, \dots, K - 1$ are called the hidden layers. Each of these layers consists of nodes x_i^k, for $i = 0, \dots d^k$ where d^k is the dimension of layer k. In addition to these d^k nodes, each hidden and input layer also has one bias node b^k. In the neural network, we assume that all nodes n_i^{k-1} are connected to all nodes x_j^k by a direct arc with weight w_{ij}^k. See Figure 1 for an example network.

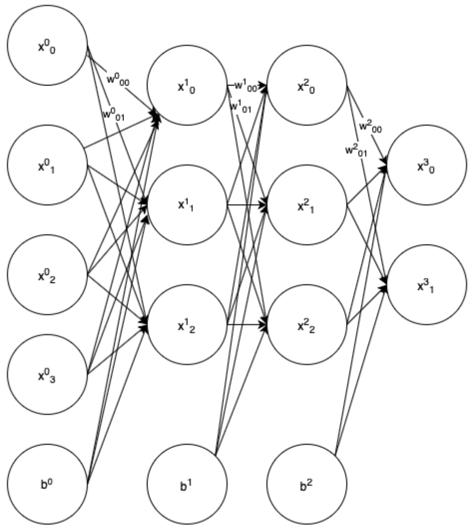


Figure 1: An example of a DNN with 1 input layer with 4 nodes, 2 hidden layers with 3 nodes each, and 1 output layer with 2 nodes. Not all weight arcs are named.

For ease of notation, we will now define x^k as the vector of values of the nodes x_j^k, and W^k as the matrix of values of the weights w_{ij}^k. Now each layer can be computed using

$$x^k = \sigma(W^{k-1}x^{k-1} + b^{k-1}) \tag{1}$$

where $\sigma(.)$ is some nonlinear function. In essence, we first compute a weighted sum of outputs from the previous layer. The nonlinear function is applied to capture more complicated behaviors.

In the model this nonlinear function will be the rectified linear unit (ReLU) function. this can be represented as

$$x^k = max(0, W^{k-1}x^{k-1} + b^{k-1}) \tag{2}$$

This function is piece-wise linear and only outputs a value if the weighted sum is greater than 0.

The DNN can be estimated using software packages such as Tensorflow (Ashish Agarwal et al., 2015) or Keras (Chollet et al., 2015). However, some hyperparameters still need to be specified. In this paper, we will use the categorical loss function in combination with the Adam optimizer. Other hyperparameters are the batch size and number of epochs.

As mentioned in the literature review, it is important to preprocess the data. For the preprocessing, we use the module preprocessing in the software package scikit-learn (Pedregosa et al., 2011).

The model is evaluated on a set of test observations. These observations will not be used for training the model. The accuracy will be computed using the test observation. However, for imbalanced datasets this might lead to poor interpretations, therefore instead the confusion matrix is computed. The confusion matrix can then be used to evaluate the model.

3.2 Mixed integer linear programming formulation for DNN

3.2.1 Mathematical model

The formulation in this section is based on the ideas of Fishetti and Jo (2018), as well as the explanations of Cheng et al. (2017). We note that this model can not be used to estimate a DNN. Instead, the main purpose is to estimate inputs such that some set of given constraints hold.

An important step in creating a MILP formulation for the DNN is the linearization of the ReLU function. We notice that the ReLU function can be split into two nonnegative parts by introducing the variables s^k as

$$W^{k-1}x^{k-1} + b^{k-1} = x^k - s^k \qquad x^k \geq 0, s^k \geq 0 \tag{3}$$

However, given the weights $W^{k-1}x^{k-1}+b^{k-1}$ this will not lead to a unique solution. This is because, assuming the pair x^k and s^k is a solution, the pair $x^k + \delta$ and $s^k + \delta$ will also be a solution since $x^k - s^k = x^k + \delta - (s^k + \delta)$. In order to obtain a unique solution we require that only one of the terms can be nonnegative. To do this we introduce the variable z^k and requiring that

$$\left.\begin{aligned} z_j^k = 1 \quad & \rightarrow x_j^k \le 0 \\ z_j^k = 0 \quad & \rightarrow s_j^k \le 0 \\ & z_j^k \in \{0,1\} \end{aligned}\right\} \tag{4}$$

Here the right arrow indicates a logical constraint. Thus the first equation can be interpreted as if z is equal to one then x is less than zero.

These logical constraints can be introduced in modern MILP solver such as Cplex (Cplex, 2020). These constraints can be modeled using big M constraints as:

$$x_j^k \le M^+(1 - z_j^k) \tag{5}$$

and

$$s_j^k \le M^-(z_j^k) \tag{6}$$

These constraints will be automatically generated by a modern MILP solver. We now give an overview of the parameters and variables in the MILP formulation

Table 1: Description sets, parameters and variables for the MILP formulation

	Indices
k	Index that runs over all layers
j	Index that runs over all nodes in a layer
i	Another index that runs over all nodes in a layer
	Parameters
K	Number of layers
n_k	Number of nodes in layer k
w_{ij}^{k-1}	Weight of arc going from node i of layer $k-1$ to node j of layer k
b_j^k	Bias of node j in layer k, note that this can be modeled by a bias b^k multiplied by a weight to obtain node specific biases
c_j^k	Weight parameter determine what to optimize in the objective functions
γ_j^k	Weight parameter determine what to optimize in the objective functions
lb_j^k	Lower bound for each variable x_j^k

ub_j^k	Upper bound for each variable x_j^k
$\overline{lb}_j^k$	Lower bound for each variable s_j^k
$\overline{ub}_j^k$	Upper bound for each variable s_j^k
Variables	
x_j^k	Positive value of node j in layer k
s_j^k	Negative value of node j in layer k
z_j^k	Auxiliary variable of node j in layer k such that either x_j^k of z_j^k is 0 (or both)

Using these definition we obtain the following mathematical model:

$$\min \sum_{k=0}^{K} \sum_{j=1}^{n_k} c_j^k x_j^k + \sum_{k=1}^{K} \sum_{j=1}^{n_k} \gamma_j^k z_j^k \qquad \text{s.t.} \tag{7}$$

$$\left.\begin{array}{c} \sum_{i=1}^{n_{k-1}} w_{ij}^{k-1} x_i^{k-1} + b_j^{k-1} = x_j^k - s_j^k \\ x_j^k, s_j^k \geq 0 \\ z_j^k \in \{0,1\} \\ z_j^k = 1 \rightarrow x_j^k \leq 0 \\ z_j^k = 0 \rightarrow s_j^k \leq 0 \end{array}\right\} k = 1, \ldots, K, j = 1, \ldots, n_k \tag{8}$$

$$lb_j^0 \leq x_j^0 \leq ub_j^0, \quad j = 1, \ldots, n_0 \tag{9}$$

$$\left.\begin{array}{c} lb_j^k \leq x_j^k \leq ub_j^k \\ \overline{lb}_j^k \leq s_j^k \leq \overline{ub}_j^k \end{array}\right\} k = 1, \ldots, K, j = 1, \ldots, n_k \tag{10}$$

In this model Equation 7 represents the objective value. This objective value is a general formulation that can be changed using the weights c_j^k and γ_j^k for different purposes. Equations 8 represent the nodes in each layer, the linearized formulation for the ReLU function can be seen. Equation 9 and Equation 10 represent the bounds of the variables that can improve the computational time of the model.

The value of the bounds in Equation 9 is determined by the input of the variables. For continuous features, these bounds will be determined by their natural range.

We can now expand the MILP formulation to handle categorical variables. This is achieved by changing Constraint 9 for different kinds of features.

For binary features Constraint 9 is removed and instead the following constraint is added

$$x_j^0 \in \mathbb{B} \tag{11}$$

Similarly, ordered categorical features can be represented using constraints. Now Constraint 9 will be used to give the range of the categorical feature.

$$x_j^0 \in \mathbb{Z} \tag{12}$$

The last category of features are unordered categorical features. In order to use this in the DNN as well as in the MILP this feature first needs to be transformed into multiple binary features, of which exactly one feature will have the value 1. Now assume that nodes x_i untill x_{i+m} correspond to transformed feature. The feature can now be represented by

$$x_j^0 \in \mathbb{B} \qquad \forall j = i, \dots i + m \tag{13}$$

$$\sum_{j=i}^{i+m} x_j^0 = 1 \tag{14}$$

The objective function in Equation 7 can be used to maximize the output of some hidden or output node. However, this model can be modified to produce adversarial examples of the network. First, assume that we have some given input value $\tilde{x}^0$. Now we want to calculate an input value x^0 such that a misclassification occurs. First we add the constraint

$$x_i^k > 1.2 x_j^k \qquad \forall j \neq i \tag{15}$$

where x_i^k represents the output node that x should be classified as and x_j^k represents all other output nodes. This constraint ensures that the misclassification occurs. The value of 1.2 is chosen such that the predicted is at least 1.2 times as large as the second largest output value. Assuming that we want a new image x^0 that is similar to $\tilde{x}^0$ the objective function can be written as

$$\min \sum_{i=1}^{n_0} d_i \tag{16}$$

where the auxilliary variable d_i is calculated using the constraints

$$-d_i \leq x_i^0 - \tilde{x}_i^0 \leq d_i \tag{17}$$

$$d_i \geq 0 \tag{18}$$

This is similar to minimizing the Manhattan distance between the constructed and original input value. We also notice that we can limit the maximum change of an input variable by including a constraint such as

$$d_i \leq 0.2 \tag{19}$$

if the maximum allowed change is 0.2.

3.2.2 Bounding procedure

To improve the computational time of the model (7-8) bounds can be computed. This can be done by scanning nodes in increasing layers. Then, after removing the other nodes in the current and subsequent layers the model is solved twice. First, x_j^k is maximized and then s_j^k is maximized. Due to the nature of the DNN, the bounds of the nodes can be used when computing bounds in a later layer. See Algorithm 1 for a pseudocode.

Algorithm 1 Compute bounds for the hidden layer of the DNN

```
procedure COMPUTEBOUNDS($ubX$,$ubS$)
    Create an array for ubX and for ubS
    for $k = 1 \dots K$ do
        for $j = 1 \dots n_k$ do
            create model for layers $1 \dots k$ and node $j$ of layer $k$
            input already computed bounds for layers $1 \dots k$
            $newUbX = \max x_j^k$
            $newUbS = \max s_j^k$
            put computed Ub in the arrays
        end for
    end for
    return arrays with bounds
end procedure
```

3.3 Fairness

In this subsection, the mathematical definition of fairness is given. Then, the adaptions of the MILP formulation are presented.

3.3.1 Defining fairness

In this paper, we focus on individual fairness. In particular, we use the definition of fairness according to Gajane and Pechenizkiy (2017) that states:

Definition 3.1. A predictor f achieves individual fairness iff $f(x^0) \approx f(\tilde{x}^0) | d(x^0, \tilde{x}^0) \approx 0$ where $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a distance metric for individuals.

In social sciences, this definition is the same as individualistic egalitarianism. It fundamentally states that reasonably close individuals (by some distance function) should be classified the same. In particular, using this definition we do not care about the original labels instead we look at the predicted labels.

Using John et al. (2020) the definition can be formalized by removing the approximation to get the following definition:

Definition 3.2. A predictor f achieves individual fairness iff $D(f(x^0) - f(\tilde{x}^0)) < \delta \; | d(x^0, \tilde{x}^0) < \epsilon$ where $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a distance metric for individuals and D on $\mathbb{R}$ (the co-domain of f) is a distance metric for classifications

For classification models it is appropriate to use the discrete metric $D(y_i, y_j) = \mathbb{I}[y_i = y_j]$, where $\mathbb{I}$ is the indicator function and $\delta = 0$, Since a small change in features should not lead to a change in classification. the definition for distance between individuals d is problem dependent. For now assume that a change is not significant if $d(x^0 - \tilde{x}^0) < \epsilon$. This leads to the notation of an unfair model (also called biased model):

Definition 3.3. A predictor f is said to be biased iff there exists a pair of valid inputs x^0 and $\tilde{x}^0$ with $D(f(x^0), f(\tilde{x}^0)) > \delta$ such that $d(x^0, \tilde{x}^0) < \epsilon$

this can be rewritten to:

Definition 3.4. A predictor f is said to be unbiased iff for all pairs of valid inputs x^0 and $\tilde{x}^0$ with $D(f(x^0) - f(\tilde{x}^0)) > \delta$ we have $d(x^0, \tilde{x}^0) > \epsilon$.

Since the MILP model is currently only applicable for a pair of $(x, \tilde{x})$ where one of them is given. We, therefore, introduce a weak definition of an unbiased predictor

Definition 3.5. A predictor f is said to be weakly unbiased iff for all x^0 and for all $\tilde{x}^0 \in T$ with $D(f(x^0) - f(\tilde{x}^0)) > \delta$ we have $d(x^0, \tilde{x}^0) > \epsilon$. Where T denotes a set of test observations.

Using this definition we test if a predictor is unbiased for a certain group of observations. This may mean that we obtain a biased predictor if the test set is significantly different from new observations. However, this does increase the applicability in practice as well as leading to more freedom in constructing the test set. For example, the test set can be selected based on some feature.

3.3.2 Adaptations to the MILP formulation

Now, similarly to building adversarial examples to Section 3.2.1 we can specify the objective function as

$$\min \sum_{k=1}^{n_f} d_k \tag{20}$$

where d_k denotes the distance between observation x and $\tilde{x}$ for feature k and n_f denotes the number of features. In combination with the constraint

$$x_i^k > 1.2 x_j^k \qquad \forall j \neq i \tag{21}$$

to ensure a misclassification occurs.

In order to determine the distance d_k we use the following constraints for continuous, binary and ordered categorical features

$$-d_k \leq x_j^0 - \tilde{x}_j^0 \leq d_k \tag{22}$$

For unordered categorical features we use the formula

$$d_k = \sum_{i \in k} \frac{q_i}{2} \tag{23}$$

here q_i is an auxiliary variable that can is determined by

$$-q_i \leq x_j^0 - \tilde{x}_j^0 \leq q_i \tag{24}$$

In Constraint 23 the sum is divided by half, because if the categorical feature is changed two input nodes are changed. We thus obtain a model that can check the definition of an unbiased predictor. This model minimizes the distance between observations such that an undesired classification occurs. If we find that the minimum distance is larger than the previously specified parameter ϵ we have found an unbiased predictor.

4 Data

4.1 MNIST

In this paper, the Modified National Institute of Standards and Technology (MNIST) dataset will be used (Lecun et al., 1998). The MNIST dataset contains around 60,000 greyscale images of handwritten digits, see Figure 2 for two examples. These images are 28 by 28 pixels and normalization is applied. Therefore, little time has to be spent on additional preprocessing. This dataset is used to train a DNN to classify the figures according to the digit in the figure. This problem is common in machine learning applications (Fishetti & Jo, 2018).

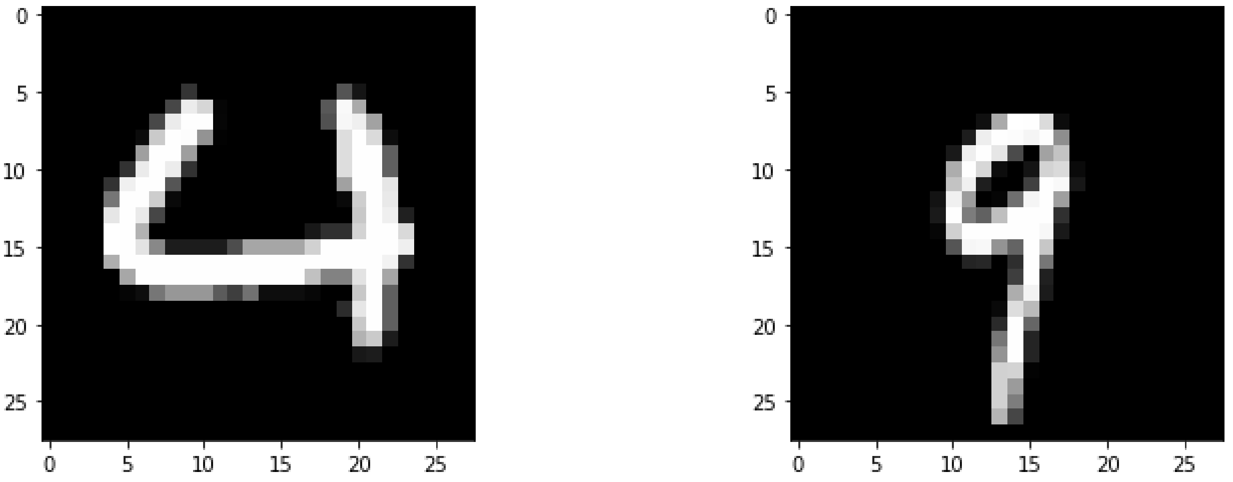


Figure 2: Examples of images present in the MNIST dataset

The only preprocessing that has to be done is rescaling the values of the pixels. As the original data has values between 0 and 256 and by dividing by 256 we change this to values between 0 and 1.

4.2 Fraud detection dataset

The second dataset that will be used in this paper is the open-source Oracle database containing observations on car insurance fraud ("Oracle Database Online Documentation", 2015). The data consists of 15420 observations with originally 33 features. In addition, the dataset contains a column with a binary feature that specifies if an observation is fraudulent or not. This also is the output that the DNN is trying to predict. After feature selection, the following features remained in the model:

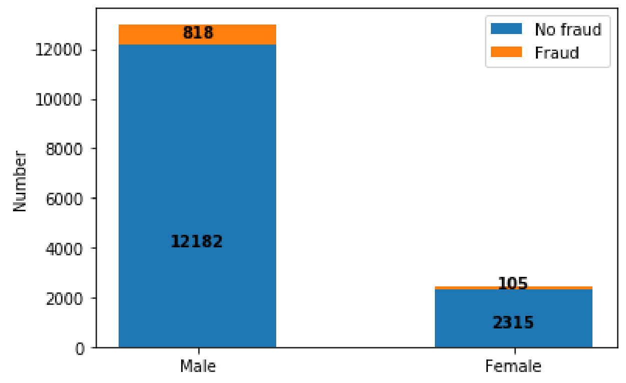
Table 2: Overview of the selected features in the fraud detection dataset

Feature name	Type	Number of categories
Accident area	Binary	N.A.
Age	Numerical	N.A.
Driver rating	Numerical	4
Make	Unordered categorical	19
Marital status	Unordered categorical	4
Number of cars	Ordered categorical	4
Police report filed	Binary	N.A.
Sex	Binary	N.A.
Vehicle category	Unordered categorical	3
Vehicle price	Ordered categorical	4
Witness present	Binary	N.A.

Some of these variables are related to the incident, such as the accident area and if there was a witness present, whereas other information is about the driver, such as sex and age.

In Figure 3 it can be seen that the dataset contains significantly more males (13000) than females (2420). The ratio of fraudulent observations for both males and females can be calculated. For males 6.29% of observations is fraudulent, on the contrary, for females, this is only 4.33%. Thus, relatively fewer females commit fraud.

In Figure 4 the descriptive statistics of marital status are presented. Again, the features are not balanced as only 35 widows are present in the dataset. Now the frequency of fraudulent observations is 6.01% for married persons, 5.93% for single persons, 3.95% for divorced persons, and 8.57% for widowed persons



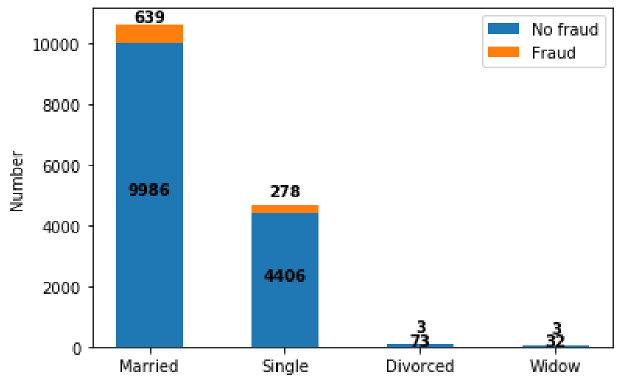
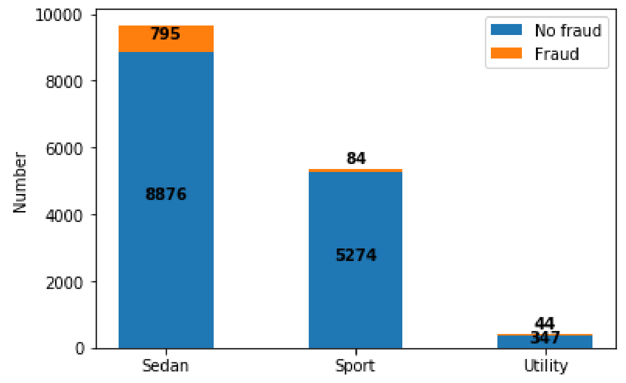


Figure 3: Frequency distribution of fraudulent observations based on sex

Figure 4: Frequency distribution of fraudulent observations based on marital status

The same can be done for features related to the incident. In Figure 5 the descriptive statistics for vehicle category can be seen. The percentage of fraudulent observations are 8.22%, 1.56%, and 11.2% for sedans, sport, and utility vehicles respectively. In Figure 6 the statistic regarding the area where the accident took place is presented. We find that 5.72% of accidents took place in urban areas while 8.32% of accidents took place in rural areas.



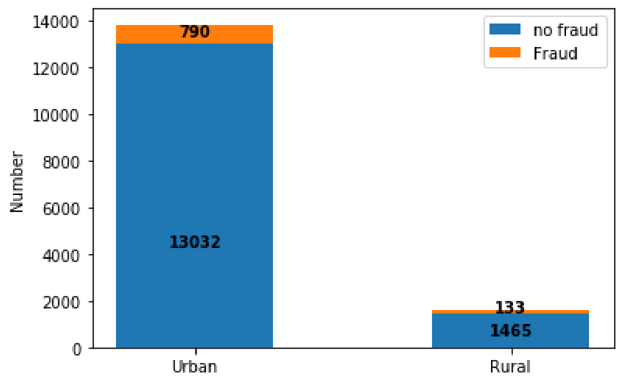


Figure 5: Frequency distribution of fraudulent observation based on vehicle category

Figure 6: Frequency distribution of fraudulent observations based on accident area

The last descriptive statistic is on the make of the vehicles. In Table 3 it can be seen that some of the makes have only a few observations. In particular, we see that owners of Ferrari, Jaguar, Lexus, and Porsche have no fraudulent observations. We also notice that owner of Accura, Mercedes, and Saab have a high frequency of fraudulent observations.

Table 3: Descriptive statistics regarding the amount and frequency of fraudulent observation based on make

Make	Amount no fraud	Amount fraud	Fraud (%)
Accura	413	59	12.50
BMW	14	1	6.67
Chevrolet	1587	94	5.59
Dodge	107	2	1.83
Ferrari	2	0	0
Ford	417	33	7.33
Honda	2622	179	6.39
Jaguar	6	0	0
Lexus	1	0	0
Mazda	2231	123	5.23
Mercedes	3	1	25.00
Nisson	29	1	3.33
Pontiac	3624	213	5.55
Porsche	5	0	0
Saab	97	11	12.22
Saturn	52	6	10.34
Toyota	2935	186	5.96
VW	275	8	2.83

Many of the features of the dataset are non-numeric. Thus, as described in Section 3 these features will be transformed using a software package. In addition, the data is also highly unbalanced since only 923 observations are fraudulent. Thus only 5.99% of observations are fraudulent. For training purposes oversampling is applied. After oversampling, we obtain 173296 observations of which 2889 (16.60%) are fraudulent.

5 Results

In Subsections 5.1, 5.2, and 5.3 the results regarding the output of the model will be described while in Subsection 5.4 computational time will be studied in more detail. In Subsections 5.1, 5.2, and 5.3 for the MNIST dataset a DNN with 4 hidden layers with 20, 10, 8, and 8 neurons is used. This Neural network was trained for 50 epochs using the optimizer Adam with batch size equal to 32 and a categorical cross-entropy loss function. After 50 epochs this DNN obtained a test set accuracy of around 98%.

The fraud detection DNN consists of 3 hidden layers with 25, 15, and 10 neurons. It was trained for 50 epochs using the optimizer Adam with the batch size equal to 16 and a binary cross-entropy loss function. The model was trained on the preprocessed data as explained in Section 4. The confusion matrix is presented in Table 4

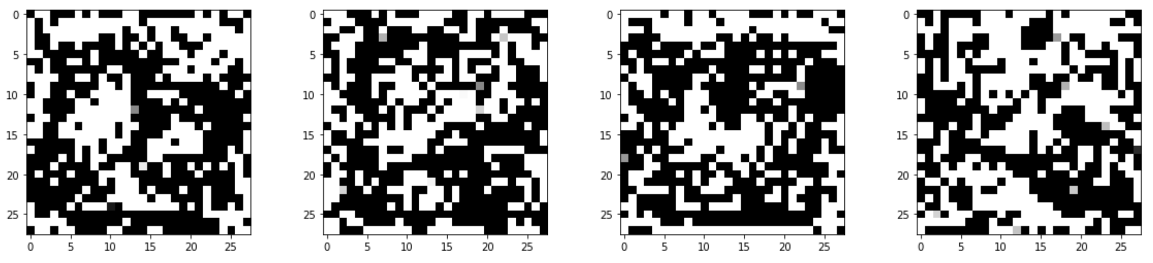
Table 4: Confusion matrix of the classification of the fraud detection DNN

	Not fraud predicted	Fraud predicted
Not fraud	14263	234
Fraud	2544	355

The confusion matrix shows that the network has a combined accuracy of 84%. However, if a fraud flagged is raised there still is a large chance that the classification is wrong. Based on this, in practice, this DNN would likely not be used. However, for now, it still will be used to test the model for computing the fairness metric.

5.1 Feature visualization

One of the applications of the MILP formulation is to maximize or minimize the output of some node in the network. In the figures below the output of some of the nodes in MNIST DNN are given. Unfortunately these figures do not show any meaningful results.



(a) Input for max node 0 (b) Input for max node 1 (c) Input for max node 2 (d) Input for max node 3

Figure 7: Examples of input figures that maximize one of the output nodes

The same can be done for the insurance fraud DNN. Here we maximize the output node of either fraud or not fraud. We now obtain the following table.

Table 5: Input values such that either a fraudulent or non fraudulent output is maximized

Fraud	Make	Marital Status	Vehicle category	Accident area	Sex	Police report	Witness present	Driver Rating	Age	Price	Number of Cars
No	Jaguar	Divorced	Sport	Rural	Female	Yes	Yes	4	80	<2000	> 8
Yes	Accura	Married	Sedan	Rural	Male	No	No	4	54	3000-3900	1

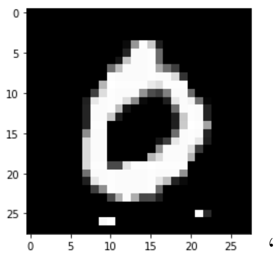
In Table 5 the inputs that maximize fraud and no fraud can be seen. Interestingly accident area and driver rating are equivalent across both observations. This either means that these features do not affect the model or that the model has found some multivariate non-linear behavior. The variables witness present and police report are flipped between the two observations which might indicate that have a simple correlation with the output node.

If the results from Table 5 are compared to the descriptive statistics presented in Section 4, some interesting trends can be seen. For example, Jaguar, the brand that maximizes the no fraud output, has a 0% frequency of fraud in the original dataset. Moreover, the make that maximizes a fraudulent classification, Accura, was found to have the highest percentual fraud of brands with more than 10 observations.

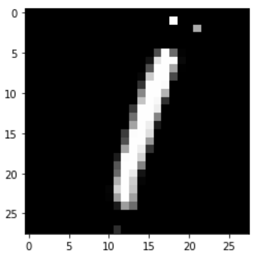
For other features, such as marital status, sex, and vehicle category, this relation can also be seen. Thus, we suspect that the frequency of fraud in the original data correlates to the features seen in Table 5. This is expected since the DNN tries to fit the data as well as possible.

5.2 Adversarial examples of the MNIST dataset

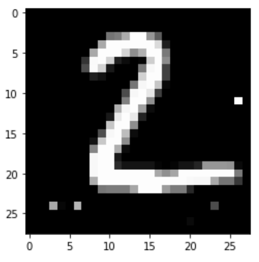
Using the MILP formulation for the DNN we are able to find examples of figures that are a minimum distance away from misclassification. In the figures below the model has been specified to find an example such that each figure is classified as the original label plus 5 modulo 10. In Figure 9, the absolute change of the pixels is presented. For clarity, the colors have been changed.



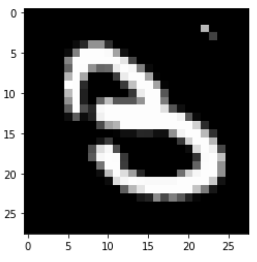
(a) label 0,
now classified as 5



(b) label 1,
now classified as 6

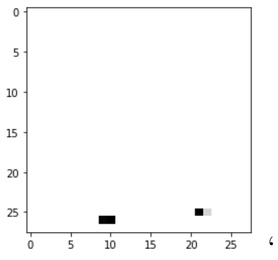


(c) label 2,
now classified as 7

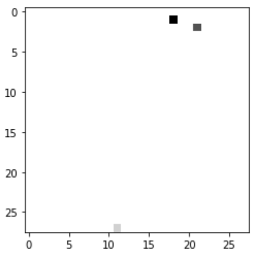


(d) label 3,
now classified as 8

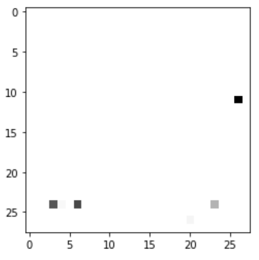
Figure 8: Examples of figures such that the figure is wrongly classified by the DNN



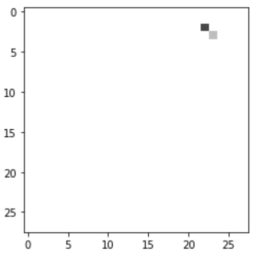
(a) label 0



(b) label 1



(c) label 2

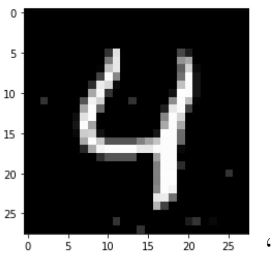


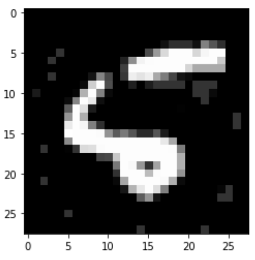
(d) label 3

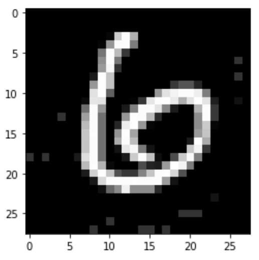
Figure 9: Absolute change of pixel values between the original figures and the adversarial examples in Figure 8

In the figures above it can be seen that to change the classification only a few pixels need to be modified. In Figures 8a and 8b with labels 0 and 1 only 2 to 4 pixels are changed. In Figures 9a and 9b this can be more clearly seen. However, the pixels that change often change by a large amount. More examples can be found in Appendix A.

As mentioned in the methodology, the model can also be used such that we limit the maximum change of each pixel from the original. This is interesting since in Figure 8 pixels change by a large amount. The maximum change of a pixel is limited to 0.2. The results of this can be seen in the figures below. Again in Figure 11, the absolute change of pixels is shown. For clarity, the colors have been reversed and the values have been scaled between 0 and 0.2.


(a) label 4,
now classified as 9


(b) label 5,
now classified as 0


(c) label 6,
now classified as 1

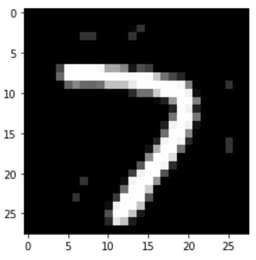
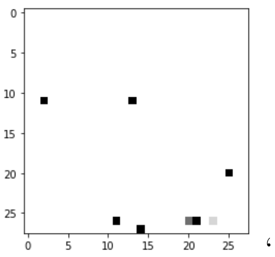
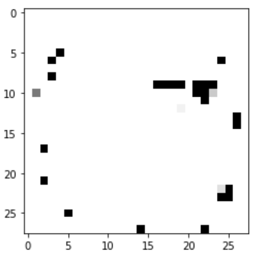
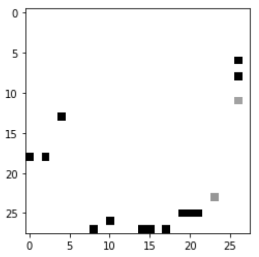

(d) label 7,
now classified as 2

Figure 10: Examples of figures such that the figure is wrongly classified by the DNN, where each pixel can change by at most 0.2


(a) label 4


(b) label 5


(c) label 6

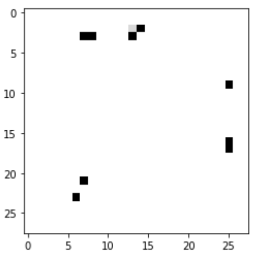

(d) label 7

Figure 11: Absolute scaled change of pixel values between the original figures and the adversarial examples in Figure 10

Now it is more difficult to see any outlying pixel, while the number still remain recognizable. The changes of figures 10a-10d can be more clearly seen in Figures 11a-11d. More examples can be found in Appendix B.

5.3 Fairness metric

For the DNN for insurance fraud, we first study examples of minimum distance to a opposite classification. In Table 6 the difference in features such that a misclassification occurs is presented for 6 observations. The complete results can be found in Appendix C. In the Table 6, the Column predicted label shows the original classification of the observations. The Column distance shows the distance according to the objective value as explained in Section 3. The last two Columns show the first and possibly second difference between the original and adversarial example.

Table 6: Examples of differences in features such that a misclassification occurs. Full results can be found in Appendix C

.

Example	Predicted label	Distance	First difference		Second difference	
1	No Fraud	2.0	Make	VW to Toyota	Vehicle Category	Sport to utility
2	No Fraud	1.0	Marital status	Divorced to widow	-	-
3	No Fraud	1.0	Make	Pontiac to Saab	-	-
4	No Fraud	1.1	Make	Honda to Accura	Age	42 to 43
5	Fraud	0.2	Age	26 to 23	-	-
6	Fraud	0.2	Driver rating	2 to 1	-	-

Here we see that for our 6 examples the minimum distance between a misclassification is 0.2 while the maximum distance is equal to 2.0. The reason that some of the distances are not integer is caused by the scaling of the ordered categorical variables.

Overall, it can be seen that only a few features have to change in order for a misclassification to occur. Some of the changes can be explained by looking at the descriptive statistic presented in Section 4, such as the change of Honda to Accura. However, for other changes such as the difference in driver rating the explanation is less clear.

Now this method is performed for 100 test observation. Using the distances it can be evaluated if the model is a weakly unbiased estimator. In Table 7 the results of the distance are presented.

Table 7: Fairness metric based on a test set of 100 observations

	Minimum	Average	Maximum
All observations	0.155	1.367	2.960
Not fraud observations	0.615	1.409	2.960
Fraud observations	0.155	0.356	0.591

For a fair model, we expect that the minimum distance between two individuals should be at least 2 features since in that case, no single change could modify the classification. Unfortunately, this model has a minimum distance of 0.155, found to be a change in age of 2 years. This also means that the model is not weakly unbiased. We also see a large difference between the fraud and not fraud groups. This is likely due to the fact that the model is not certain of potential fraud cases, while most non-fraud cases seem more clear to the model. Based on these metrics the DNN should not be used in practice.

The average changes of each feature are presented to show the effect of the features on the fairness metric in Table 8. For the features driver rating, age, price and number of cars the change can not be compared to the original categories since the change is determined for the scaled categories.

Table 8: Number of changes and average change for each feature

	Make	Marital status	Vehicle category	Accident area	Sex	Police Report	Witness present	Driver rating	Age	Price	Number of car
Amount changes	69	9	27	8	1	0	2	35	25	29	11
Average absolute change	0.69	0.09	0.27	0.08	0.01	0.00	0.02	0.02	1.50	0.08	0.01

In Table 8 it can be seen that most often the brand changes. Additionally, it can be seen that the age features change by a relatively large amount. The features sex, police report, and witness present change very little thus likely have a smaller influence on the classification than other variables.

To see if the effect of the features make and age on the fairness metric, the fairness metric is also computed with the restriction that make and age cannot change. The results can be found in Table 9.

Table 9: Fairness metric based on a test set of 100 observations where the features make and age are not allowed to change

	Minimum	Average	Maximum
All observations	0.207	2.292	6.401
Not fraud observations	0.591	2.373	6.401
Fraud observations	0.207	1.545	3.613

Comparing the results of Table 9 to Table 7, it can be seen that minimum distance only slightly increases. It can be noticed that the average and maximum distance do increase. Thus by limiting the features make and age, the average distance for a misclassification increases. However, since the definition of a weakly unbiased predictor is dependent on the minimum distance the model is still not weakly unbiased.

One of the main advantages of the definition of the weakly unbiased estimator is that it is dependent on a test set of observations. These observations can be selected to have a certain property. For example, if we want to compare the fairness metric between males and females this can be done by creating two different test sets. In Table 10 the results for two different test set are presented.

Table 10: fairness metric based on a test set of 100 observations that either are all male or all female

	Minimum	Average	Maximum
Female observations	0.168	1.482	2.779
Male observations	0.155	1.303	2.487

It can be seen that the distances are pretty similar. In particular, the minimum distance, relevant for the definition of a weakly unbiased estimator, is only different by 0.013. Thus no major difference in fairness can be observed between males and females.

5.4 Computational times

To study the computational times of the MILP 5 different DNN were tested for the MNIST dataset. These have the following specifications:

Table 11: Structure of the hidden layers in the different DNN

Name model	Hidden layers	Nodes hidden layer
DNN1	3	8, 8, 8
DNN2	6	8, 8, 8, 8, 8, 8
DNN3	4	20, 10, 8, 8
DNN4	5	20, 10, 8, 8, 8
DNN5	5	20, 20, 10, 10, 10

The DNN used for the Insurance fraud dataset is the same as in subsections 5.1-5.3 and consists of 3 hidden layers with 25, 15, and 10 nodes.

In Table 12 the computational times of the model are presented. The values in the table are the mean of 50 instances. In each instance, an MNIST figure is modified such that it is misclassified, without any bound on the maximum change of a pixel. For the Insurance fraud model, each instance was a problem to calculate the minimum distance such that the classification flipped. For each DNN two models were tested. First, a model without a bound for the nodes in the network and secondly an improved model for which the bound have been calculated using the procedure explained in subsection 3.2.2, without a time limit. Each model is limited to 300 seconds per instance.

In Table 12 four metrics are presented. The first metric shows how many of the instances are solved within the 300-second limit, the second metric shows the average gap to the optimal solution. The column Nodes represents the average number of nodes used in the model and the last column represent the average time per instance. The basic model is the MILP formulation without any computed bounds and the improved model is the MILP formulation with bounds computed using the bounding procedure.

All of the computations where done on a laptop with a Dual-Core Intel i5 @2,6 GHz processor with 16 GB of RAM. To solve the MILP the state-of-the-art solver IBM ILOG CPLEX 20.1.0.0 was used (Cplex, 2020).

Table 12: Computational times of the model with no bounds or exact bound, based on 50 instances.

	Basic model				Improved model			
	%solved	%gap	Nodes	Time (s)	%solved	%gap	Nodes	Time(s)
DNN1	100%	0.00%	785.4	1.00	100%	0.00%	368.6	0.59
DNN2	100%	0.00%	3571.6	3.03	100%	0.00%	618.9	0.80
DNN3	100%	0.00%	30235.6	18.62	100%	0.00%	5348.6	5.02
DNN4	100%	0.00%	51961.6	33.64	100%	0.00%	3052.9	3.01
DNN5	48%	31.96%	186937.3	213.73	100%	0.00%	49686.4	27.34
Insurance	100%	0.00%	240.6	0.31	100%	0.00%	101.3	0.22

Table 12 shows that the model is able to solve all but the largest DNN. This comes as a surprise as in the paper of Fishetti and Jo (2018) the three largest DNNs were not solvable. An explanation for this could be the improvement of the software package CPLEX. In particular, in the update to

CPLEX 20.1.0.0 algorithm improvements are introduced. This is consistent with the observation that all computing times are significantly faster.

We also notice that the fraud detection network is faster than all DNNs for the MNIST dataset. This is likely caused by a significantly smaller input layer with only 34 nodes for the fraud detection network, compared to 784 nodes for the other DNNs. The addition of other restrictions such as binary variables does not seem to significantly slow down the model.

Comparing the basic model to the improved model, it is clear that the improved model significantly improves the computing times. Now, all DNNs are solvable within the 300 second time limit.

Strangely, DNN4 is faster than the smaller network DNN3. This is unexpected since the formulation of DNN4 contains more big M constraints. However, comparing the bounds of DNN4 to DNN3, we found that the bounds for DNN4 were stricter. Thus, this could be a possible explanation.

The preprocessing of the bounds took 108.3 and 819.6 seconds for DNN4 and DNN5, respectively. Similar to the paper of Fishetti and Jo (2018) we also test the DNNs using the bounds computed with a time limit of 1 second per node. In Table 13. No major time difference can be seen between the exact and weaker bounds. It can be seen that for both the exact and weaker bound that DNN4 is faster than DNN3.

Table 13: Computational times comparing the exact bounds vs the weaker bounds, based on 50 instances

	Improved model									
	Exact bounds					Weaker bound				
	t.pre.(s)	%sol.	%gap	Nodes	Time(s)	t.pre.(s)	%sol.	%gap	Nodes	Time(s)
DNN3	150.8	100	0.00	5348.6	5.02	54.0	100	0.00	6526.9	5.71
DNN4	108.3	100	0.00	3052.9	3.01	55.60	100	0.00	2481.2	3.93
DNN5	819.6	100	0.00	49686.4	27.34	80.06	100	0.00	16011.0	29.81

In Table 14 the computational times are presented for computing a solution with a 1% gap to an optimal solution. The maximum running time for each instance was 3600 seconds. The column #timlim denotes the number of times instance took longer than the time limit of 3600 seconds to complete. Again, 50 instances are considered.

Table 14: Computational times to find a solution within 1% of optimality without bounds or with weaker bound, based on 50 instances

	Basic Model				Improved Model (Weaker bounds)			
	#timlim	Time (s)	Nodes	%gap	#timlim	Time (s)	Nodes	%gap
DNN1	0	0.86	1104.3	0.37	0	0.52	365.9	0.25
DNN2	0	2.76	4282.9	0.53	0	0.77	617.0	0.28
DNN3	0	18.5	29700.9	0.78	0	5.91	6913.0	0.69
DNN4	0	24.5	32621.3	0.69	0	2.50	2427.4	0.56
DNN5	7	1435.18	1664204	3.80	0	30.71	15018.6	0.90

In Table 14 it can be seen that for the first four DNNs the model is solvable. Only the largest model has 7 instances for which the optimal solution can not be computed within the 3600 second time limit. The improved model with weaker bounds is able to solve all DNNs. Additionally, we see again that DNN3 is solved faster than DNN4 for the improved model.

Comparing the results to the results of Fishetti and Jo (2018) leads to a similar comparison as the results in Table 12 where the computational times are significantly faster.

6 Conclusion

This paper investigated the MILP formulation for DNNs. This model can be used to create adversarial examples. The results of Fishetti and Jo (2018) were replicated using this model. Additionally, we have introduced a new definition called weakly unbiased estimator to determine individual fairness for DNNs. The MILP formulation was then extended for the fairness metric.

We have found that it is possible to use the MILP formulation to obtain results similar to Fishetti and Jo (2018). In particular, we have found that to create adversarial examples for a DNN trained on the MNIST dataset only a few pixels need to be changed. Similarly to Fishetti and Jo (2018) we also find that it is possible to maximize some output or internal node and that no pattern can be recognized.

Additionally, we have found that it is possible to extend the formulation of the MILP to non-continuous features. For the fraud detection dataset used in this paper the computational time was not longer than the computational time for the MNIST dataset.

Furthermore, we have been able to analyze fairness within the fraud detection network. It is shown that the fraud detection network cannot be considered fair. The fraud detection network is then further analyzed to see the changes in features. However, even after adding a constraint for the features that change most often the model can still not be considered fair. Moreover, we have shown that the model can also be used for a test set with specific features. Using this we find no major differences in fairness between males and females.

Although the results regarding the fairness metric look promising it is still uncertain how well the model would work for other datasets. In particular, we note that the fraud detection network has a low accuracy of only around 85%. Furthermore, to solve the problems with imbalanced data oversampling was performed. However, it is unknown how this would affect the results. Lastly, as shown in Section 4 some features of the data occur extremely infrequently.

Finally, we have determined the computational time of the MILP model for different DNNs. We have found significantly faster computational times than Fishetti and Jo (2018). This is likely caused by software improvements. In addition, it is found that by adding bounds to the model that the computational times improve.

Looking at the computational times of the network we find the larger network DNN4 is faster than the network DNN3. We suspect that this is caused by better bounds of the nodes, however, further research needs to be performed to confirm this. In particular, the solver, IBM ILOG CPLEX 12.7, used in the paper by Fishetti and Jo (2018) can be used.

7 Further research

In this paper, we have introduced a new concept to verify individual fairness in machine learning methods. This concept can be explored further in several ways. For example, it would be interesting to see what kind of results can be produced for different datasets. In particular, a dataset with balanced data such that will not affect the model. In addition, it could be interesting to test a dataset with significantly more features. This reduces the importance of each feature and thus may lead to a DNN that does satisfy the fairness criterion.

The criterion can also be further analyzed by comparing it to other fairness criteria to see if similar results are achieved. Additionally, computational times between different methods could also be compared.

Another idea for future research is to test different specifications of the criterion. For example, in the methodology of the fairness metric, we have used a basic distance function. Another distance function that took into account the frequency of the features might perform better. In the results a basic test set was used, the relationship selection of the test set relates to the fairness metrics can be examined.

After the criterion is evaluated against other fairness metrics and it ensured to produce accurate results, it would be interesting to test DNN in practice, such as “Syteem Risco Indicatie” (SyRI) by the Rotterdam municipality, on how well they score for the fairness metric. In particular, it could show the robustness of a DNN as well as the effect of each feature on the fairness metric. This could lead to more trust and understanding of classification networks in practice.

References

Anderson, D., & McNeill, G. (1992). Artificial neural networks technology. *Kaman Sciences Corporation*, *258*(6), 1–83.

Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., & Vielma, J. P. (2020). Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 1–37.

Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, ... Xiaoqiang Zheng. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems [Software available from tensorflow.org]. https://www.tensorflow.org/

Bacchini, F., & Lorusso, L. (2019). Race, again: How face recognition technology reinforces racial discrimination. *Journal of information, communication and ethics in society.*

Bertsimas, D., & Kallus, N. (2014). From predictive to prescriptive analytics. *arXiv preprint arXiv:1402.5481.*

Cheng, C.-H., Nührenberg, G., & Ruess, H. (2017). Maximum resilience of artificial neural networks. *International Symposium on Automated Technology for Verification and Analysis*, 251–268.

Chollet, F. et al. (2015). *Keras.* https://github.com/fchollet/keras

Chouldechova, A., & Roth, A. (2018). The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810.*

Cplex, I. I. (2020). V20. 1: User's manual for cplex. *International Business Machines Corporation*, *46*(53), 157.

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., & Zemel, R. (2012). Fairness through awareness. *Proceedings of the 3rd innovations in theoretical computer science conference*, 214–226.

Fishetti, M., & Jo, J. (2018). Deep neural networks and mixed integer linear optimization. *Constraints*, *23*, 296–309. https://doi.org/https://doi-org.eur.idm.oclc.org/10.1007/s10601-018-9285-6

Gajane, P., & Pechenizkiy, M. (2017). On formalizing fairness in prediction with machine learning. *arXiv preprint arXiv:1710.03184.*

Gambella, C., Ghaddar, B., & Naoum-Sawaya, J. (2021). Optimization problems for machine learning: A survey. *European Journal of Operational Research*, *290*, 807–828. https://doi.org/https://doi-org.eur.idm.oclc.org/10.1016/j.ejor.2020.08.045

Hancock, J. T., & Khoshgoftaar, T. M. (2020). Survey on categorical data for neural networks. *Journal of Big Data*, *7*, 1–41.

John, P. G., Vijaykeerthy, D., & Saha, D. (2020). Verifying individual fairness in machine learning models. *Conference on Uncertainty in Artificial Intelligence*, 749–758.

Koprinkova, P., & Petrova, M. (1999). Data-scaling problems in neural-network training. *Engineering Applications of Artificial Intelligence*, *12*(3), 281–296.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2019). A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*.

Oracle database online documentation. (2015). http://docs.oracle.com/database/121/index.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Qiu, J., Wu, Q., Ding, G., Xu, Y., & Feng, S. (2016). A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, *2016*(1), 1–16.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

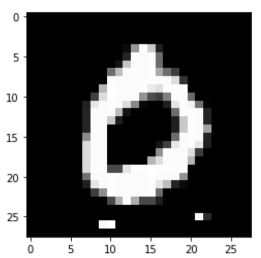
Serra, T., Tjandraatmadja, C., & Ramalingam, S. (2018). Bounding and counting linear regions of deep neural networks. *International Conference on Machine Learning*, 4558–4566.

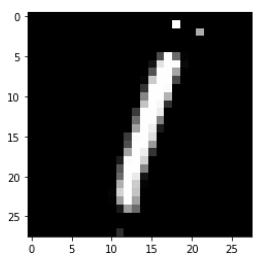
Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

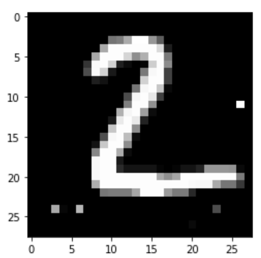
Tjeng, V., Xiao, K., & Tedrake, R. (2017). Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*.

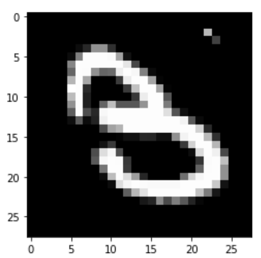
Wei, W., Li, J., Cao, L., Ou, Y., & Chen, J. (2013). Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, *16*(4), 449–475.

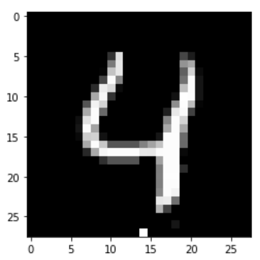
A Adversarial examples of the MNIST dataset

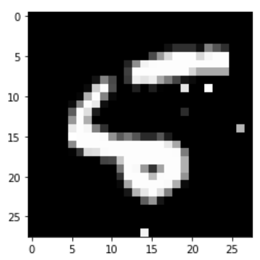

(a) label 0, now classified as 5

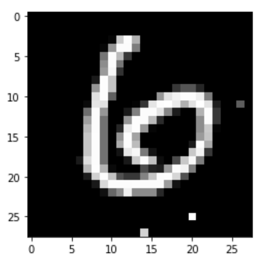

(b) label 1, now classified as 6

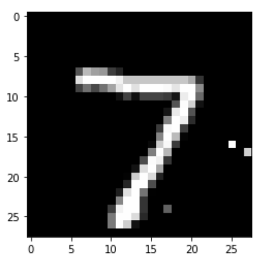

(c) label 2, now classified as 7

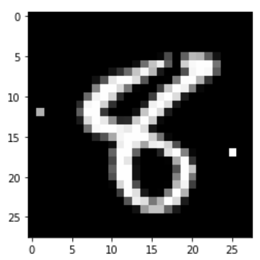

(d) label 3, now classified as 8

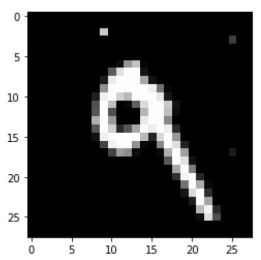

(a) label 4, now classified as 9


(b) label 5, now classified as 0

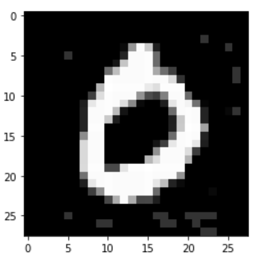

(c) label 6, now classified as 1

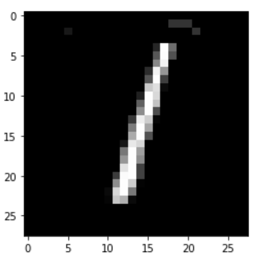

(d) label 7, now classified as 2

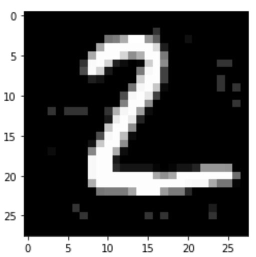

(a) label 8, now classified as 3

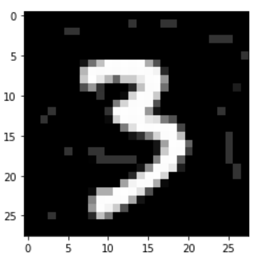

(b) label 9, now classified as 4

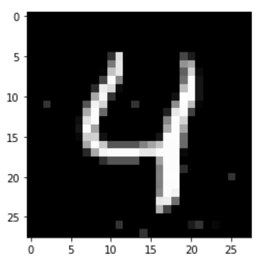
B Adversarial examples of the MNIST dataset with maximum change

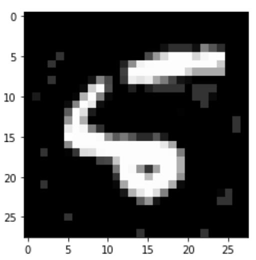

(a) label 0,
now classified as 5

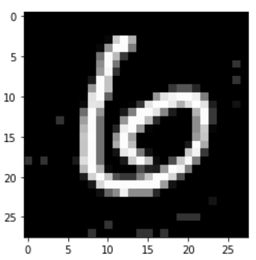

(b) label 1,
now classified as 6

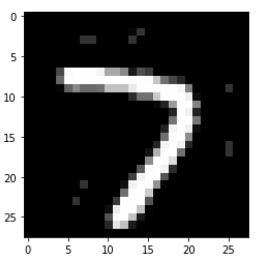

(c) label 2,
now classified as 7

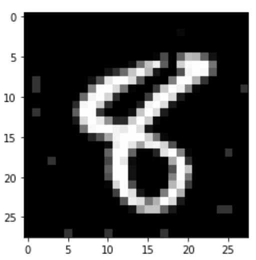

(d) label 3,
now classified as 8

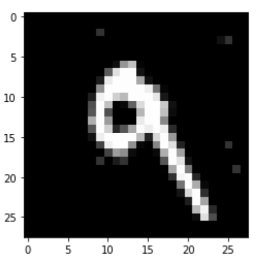

(a) label 4,
now classified as 9


(b) label 5,
now classified as 0


(c) label 6,
now classified as 1


(d) label 7,
now classified as 2


(a) label 8,
now classified as 3


(b) label 9,
now classified as 4

C Results of fairness criterion

Type	Classification	distance to original	Make	Marital Status	Vehicle category	Accident area	Sex	Police report	Witness present	Driver Rating	Age	Price	Number of Cars
Original (1)	no Fraud		VW	Married	Sport	Urban	Male	no	no	1	34	30000-39000	1
Missclassified	Fraud	2.0	Toyota	Maried	Utility	Urban	Male	no	no	1	34	3000-39000	1
Original (2)	No Fraud		Toyota	Divorced	Sedan	Urban	Female	no	no	1	32	<20000	1
Missclassified	Fraud	1.0	Toyota	Widow	Sedan	Urban	Female	no	no	1	32	<20000	1
Original (3)	No Fraud		Pontiac	Single	Sedan	Rural	Male	no	no	2	38	<20000	1
Missclassified	Fraud	1.0	Saab	Single	Sedan	Rural	Male	no	no	2	38	<20000	1
Original (4)	No Fraud		Honda	Married	Sedan	Rural	Male	yes	no	1	42	<2000	1
Missclassified	Fraud	1.1	Accura	Married	Sedan	Rural	Male	yes	no	1	43	<2000	1
Originaln (5)	Fraud		Toyota	Single	Sedan	Rural	Male	no	no	4	23	<20000	1
Missclassified	No Fraud	0.2	Toyota	Single	Sedan	Rural	Male	no	no	4	26	<20000	1
Original (6)	Fraud		Mazda	Married	Sedan	Urban	Male	no	no	2	23	>69000	1
Missclassified	No Fraud	0.2	Mazda	Married	Sedan	Urban	Male	no	no	1	23	>69000	1

D Overview code

Can also be found in the readme.md file after unzipping the code.

This code in this project correspond to the concepts and formulations introduced in Dijkstra (2021).

The files in the subfolder MNIST model and Fairness model are largly similar. The Utils.java file contains some helper function that can read and write weights of a model. The NeuralNetwork.java and Layer.java classes represent the datastructure of a Deep Neural Network (DNN). The Pair.java class contains a basic datastructure that is used to store and retrieve weights from a Map using coordinates. The files Model.java is different for both models since the formulation is also slightly different. In this class the details regarding cplex implementations can also be found. Lasty, BoundModel.java contains the implementation of the bounding procedure (See Algorithm 1 in Dijkstra(2021)).

The subfolder Python code contains files for pre and post processing of the data. The files DNN fraud detection.py and DNN MNIST datset.py contain the code that was used to train the DNNs. The file Data selection computationl results.py and Data selection.py contain the code necessary to transform the data such that it can be used by the MILP model. The file Descriptive statistics.py was used to create Figures 3-6 in Dijkstra (2021). Lastly, the file Output visualization.py contains the code to create the figures of the adveserial examples.

Dijkstra S.J. (2021) A New Metric for Verifying Individual Fairness in Deep Neural Networks. Unpublished manuscript.