Reformulating deep neural networks as mathematical programming problems

Joonatan Linkola

School of Science

Bachelor's thesis
Espoo 29.5.2023

Supervisor

Prof. Fabricio Oliveira

Advisor

M.Sc. (Tech.) Nikita Beylak

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the bachelor's thesis

Author Joonatan Linkola		
Title Reformulating deep neural networks as mathematical programming problems		
Degree programme Bachelor's Programme in Science and Technology		
Major Mathematics and Systems Sciences		**Code of major** SCI3029
Teacher in charge Prof. Fabricio Oliveira		
Advisor M.Sc. (Tech.) Nikita Beylak		
Date 29.5.2023	**Number of pages** 23	**Language** English

Abstract

Deep neural networks (DNNs) are popular machine learning models aimed at the recognition of patterns in the data. Often processes involving DNNs can be interesting in an optimization framework: one might want to maximize the output of a DNN (e.g., corresponding to production profit), such that the input variables (e.g., corresponding to materials or equipment) are constrained in some way. However, due to the complex structures of DNNs, the optimization procedure can be hard to implement. Therefore, mathematical approximations often referred to as surrogate models are used as alternatives to DNNs due to their easier computational tractability.

In this thesis, a surrogate model for a certain type of trained DNN is implemented. The surrogate model is used to generate adversarial images, i.e., images that are visually indistinguishable from the original images to the human eye but cause misclassification when processed by the DNN. The images are generated by solving two optimization problems representing the surrogates of DNNs: the first one minimizes the differences between the pixel values of the original and adversarial images (L^1-norm problem), whereas the second one minimizes the squares of the differences (L^2-norm problem).

The results show that both optimization problems for adversarial image generation have their strengths and weaknesses. The solution time for the L^1-norm problem is noticeably shorter than for the L^2-norm problem. However, the resulting adversarial images have clearly identifiable modifications made compared to the originals, which potentially makes them easier for an outside observer to detect, and hence weaker. The L^2-norm problem produces visually better adversarial images, but the solution time is longer.

Overall, the DNN surrogate models, i.e., L^1-norm and L^2-norm problems efficiently manage the task of building optimal adversarial examples, but there is still room for further improvements regarding the solution time. For instance, it is possible to determine optimal upper and lower bounds for the constraints in the model, which would reduce solution time. The adversarial image quality can also be increased by experimenting with different model parameters in L^1-norm and L^2-norm problems, or by improving the objective function and adding new constraints to these problems.

Keywords Deep neural networks, Surrogate modeling, Mathematical optimization, Mixed-integer linear programming, Adversarial examples

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Tekniikan kandidaatintyön tiivistelmä

Tekijä Joonatan Linkola					
Työn nimi Reformulating deep neural networks as mathematical programming problems					
Koulutusohjelma Teknistieteellinen kandidaattiohjelma					
Pääaine Matematiikka ja systeemitieteet				**Pääaineen koodi** SCI3029	
Vastuuopettaja Prof. Fabricio Oliveira					
Työn ohjaaja M.Sc. (Tech.) Nikita Beylak					
Päivämäärä 29.5.2023		**Sivumäärä** 23		**Kieli** Englanti	

Tiivistelmä

Syväoppivat neuroverkot ovat suosittuja koneoppimismalleja, joita käytetään tunnistamaan datassa esiintyviä epälineaarisia riippuvuussuhteita. Syväoppivilla neuroverkoilla mallinnetut prosessit ovat myös usein mielenkiintoisia optimoinnin näkökulmasta: neuroverkon tulosta (esim. tuotannon voitto) voidaan haluta maksimoida siten, että syötearvoihin (esim. saatavilla olevat materiaalit tai välineistö) on liitetty rajoitusehtoja. Syväoppivien neuroverkkojen rakenteet ovat usein monimutkaisia ja niiden optimointi voi tästä syystä olla haastavaa, joten niiden sijaan voidaan käyttää surrogaatti-malleiksi kutsuttuja laskennallisesti nopeampia matemaattisia approksimaatioita.

Tässä kandidaatintyössä kehitetään surrogaatti-malli korvaamaan eräänlainen syväoppiva neuroverkko. Mallia hyödynnetään vastakkainasettelullisten kuvien (eng. adversarial images) generoimiseen. Nämä ovat neuroverkon koulutusdatan kuvia pienillä, ihmissilmälle näkymättömillä muutoksilla, joiden tarkoitus on aiheuttaa väärä luokittelu neuroverkossa. Tässä työssä kuviin tehdyt muutokset optimoidaan mahdollisimman pieniksi. Surrogaatti-mallia hyödynnetään kahdenlaisten vastakkainasettelullisten kuvien generoimiseen ratkaisemalla kaksi eri optimointiongelmaa: näistä ensimmäinen minimoi kuvien pikselien välistä erotusta, kun taas jälkimmäinen minimoi kuvien pikselien välisen erotuksen neliötä.

Tuloksia analysoimalla huomataan, että kummallakin lähestymistavalla vastakkaisasettelullisten kuvien generoimiseen on omat hyödyt ja haitat. Ensimmäisen lähestymistavan optimaaliset ratkaisut löydetään nopeammin, mutta kuvissa on selkeitä näkyviä viittauksia siihen, että muutoksia on tehty. Tässä mielessä näitä kuvia on helpompi tunnistaa muiden neuroverkon testaamiseen käytettyjen kuvien joukosta, mikä tekee niistä heikompia. Jälkimmäinen lähestymistapa puolestaan tuottaa visuaalisesti parempia kuvia, mutta optimaaliseen ratkaisuun kuluva aika on huomattavasti suurempi.

Kehitetty surrogaatti-malli toimii siis toivotulla tavalla, mutta mallissa on vielä kehittämisen varaa ratkaisuaikojen pienentämisen sekä uusien kuvien generoimismenetelmien kehittämisen suhteen. Surrogaatti-mallin rajoitusehtojen ylä- ja alarajoille voidaan määrittää optimaaliset arvot, mikä vähentäisi lopulliseen laskentaan kulunutta aikaa. Vastakkainasettelullisten kuvien laatua voidaan myös parantaa kehittämällä uusia kuvien generoimismenetelmiä, jotka voidaan toteuttaa uuden vaihtoehtoisen kohdefunktion ja uusien rajoitusehtojen lisäämisellä optimointiongelmaan.

Avainsanat Syväoppivat neuroverkot, Surrogaatti-mallintaminen, Matemaattinen optimointi, Lineaarinen sekalukuoptimointi, Vastakkainasettelulliset esimerkit

Contents

1 Introduction

Deep neural networks (DNNs) have proven to be highly efficient in solving complex problems in various fields such as computer vision, natural language processing, and robotics to mention just a few. DNNs are machine learning (ML) models that are designed to recognize patterns in data (Bishop, 1994). The building blocks of a DNN are nodes or "neurons", which are grouped into consecutive layers. Each layer receives inputs from the previous layer, applies a non-linear function to the nodes, and produces output values to the following layer. The values at the last layer are the output values of the DNN.

DNNs are widely used in numerous applications involving mathematical optimization (Malek and Beidokhti, 2006; Jia et al., 2019; Fischetti and Jo, 2018). However, due to the non-linear nature and often complex structure, DNNs can be challenging to optimize. As a solution, one could consider replacing these complex non-linear structures with mathematical models that are often referred to as surrogate models. A surrogate model is a simplified mathematical approximation of the underlying process that is less challenging in terms of computational tractability (Sobester et al., 2008). The replacement of DNNs with surrogate models offers several benefits, especially regarding optimization. The resulting surrogate model can be more efficient and flexible in an optimization framework, enabling faster optimization time and problem modification with additional constraints. As an example, one could maximize the output of a DNN when constraints on the input values are present.

In this thesis, we focus on DNNs having the rectified linear unit (ReLU) (Nair and Hinton, 2010) as the chosen activation function. ReLU is a piecewise linear function that returns its input value if it is positive, and zero otherwise. We use a 0-1 mixed-integer linear program (0-1 MILP) as the surrogate model for trained ReLU DNNs. The notion of a trained ReLU DNN implies that the values of the internal parameters of the ReLU DNN are fixed. The 0-1 MILP surrogate model is implemented following Grimstad and Andersson (2019). The surrogate model is then used to solve image classification and adversarial image generation problems. The computational time regarding the optimization problems is also analyzed.

The outline of the thesis is as follows. Section 2 reviews previous work regarding surrogate modeling and adversarial learning. In Section 3 the 0-1 MILP formulation is presented and the case study is explained, followed by the results and computational analysis of the optimization problems in Section 4. Discussion on the results, as well as further development ideas for the 0-1 MILP surrogate model, is described in Section 5.

2 Literature review

Recently, different types of surrogate models have been utilized in various problems across many fields. As an example, polynomial response surface (PRS) models, which are statistical frameworks involving regression and variance analysis, are used by Hosder et al. (2001) in the civil transport optimization context. In Simpson et al. (1998), PRS models are used as a part of polynomial approximations for multidisciplinary optimization. Another type of surrogate model exploited for regression analysis is the support vector regression (SVR) model. SVR models were used by Parbat and Chakraborty (2020) for COVID-19 case prediction and by Dash et al. (2021) for stock forecasting. Both PRS and SVR models are supervised learning models implying that labeled training data is used to iteratively train the model such that the relationships between input and output values are learned. This procedure enables the model to accurately classify or predict the outcome of input data that has not been part of the training data set.

In addition to the supervised learning models discussed above, non-supervised interpolating surrogate models are an alternative choice for problems where data is sparse, irregularly sampled, or missing. Contrary to supervised learning, these models are trained with unlabeled data with the goal of finding patterns in the training data and defining key features. It is also worth mentioning that the training of non-supervised models often involves clustering or dimensionality reduction techniques. As an example of a non-supervised surrogate model, in Sun et al. (2011), the authors used a radial basis function (RBF) model as an effective surrogate in optimizing sheet metal forming. In RBF, the value of the function depends on the distance between a sampled and a measured point. The RBF model was demonstrated by the authors to more accurately capture changes in key metal forming attributes (e.g. cracking, wrinkling, and spring back) compared to other previously used surrogates in sheet metal fracture and wrinkle modeling.

Surrogate models are also applicable as alternative representations of various ML models. In particular, the substitution of DNNs with surrogate models offers numerous advantages that can be later exploited. For instance, surrogate model substitutions for DNNs are often designed such that the evaluation time is significantly reduced. Furthermore, if an optimization framework is used as the surrogate, this allows, for example, optimization over the DNN output. Due to their piece-wise linearity, ReLU DNNs can be represented as a set of linear constraints and binary variables and can hence be translated into 0-1 MILP frameworks for optimization purposes. Fischetti and Jo (2018) presented a 0-1 MILP problem formulation for ReLU DNNs. The formulation is applied to a digit image classification problem, where the model is used for generating adversarial images with optimally minimal changes, i.e., images that lead to misclassification by the original ReLU DNN. Subsequent to this, the authors apply bound-tightening (BT) techniques to the adversarial image 0-1 MILP problem, which results in vast improvements in solution time. The authors also note that the main drawback of the 0-1 MILP formulation is its scalability, particularly for larger ReLU DNNs, where the solution time for the adversarial image problem can grow large even for small grayscale images. It is also mentioned that an interesting avenue for future research on the topic is exploring new deep-learning applications for the 0-1 MILP model.

Grimstad and Andersson (2019) demonstrated an adaptation of the 0-1 MILP problem formulation introduced by Fischetti and Jo (2018). The authors present the 0-1 MILP formulation, after which a detailed description of various BT techniques that are later employed is given. In their paper, an emphasis is placed on the use of BT techniques to reduce the solution time of the 0-1 MILP model. Three optimization problems using the 0-1 MILP model with BT techniques are described: bound-tightening for ReLU DNNs with output constraints, optimization of n-dimensional quadratic functions, and an oil production optimization problem. The authors point out that the use of BT techniques is critical for surrogate models, especially ones based on large ReLU DNNs, in order to avoid excessively lengthy solution time. Furthermore, as a result of their 0-1 MILP solver quickly running out of memory due to a large number of variables, the model failed to find an optimal solution to the oil production optimization problem.

Adversarial examples are specifically designed objects serving as inputs to ML models and purposely causing misclassification. These inputs are often created by adding small, imperceptible perturbations to the original input data (Goodfellow et al., 2014). In image recognition, the modifications often involve changing a few pixels or adding noise to the image. One useful application of adversarial images is data augmentation. In situations where obtaining a large training data set is computationally expensive, additional synthetic samples can be generated to expand the training data set. The augmentation of the training data set, in turn, makes the DNN more robust and generalizable. In Zhao et al. (2017), a framework for generating *natural* adversarial images is introduced. These images aim to be meaningfully similar to the original images, meaning that no indication of change (e.g., obvious noise or changed pixels) is visible to the human eye. An approach for both text and image recognition is examined. The authors report that computational experiments and human evaluation point out the efficiency of their modeling approach in the context of evaluating the robustness of image-recognizing DNNs.

Adversarial examples can also be used in malicious applications, where an attacker might want to purposefully give misclassification-causing data to an ML model. The study of adversarial examples can aid in preventing such attacks, as it allows us to increase the robustness of ML models. Another novel application of adversarial examples is introduced by Athalye et al. (2018). The authors were able to manufacture physical adversarial objects that consistently cause misclassification in image recognition when changes to the background, camera distance, or rotation of the object are made. The authors presented the algorithm for adversarial object generation, after which the existence of adversarial 3D render examples is illustrated. With the help of commercial 3D printing technology and by using these 3D renders as a reference, the authors could construct physical adversarial objects. The authors also highlight that such adversarial objects can cause serious concern in practical image recognition applications.

3 Methodology

3.1 The 0-1 MILP surrogate model

We consider ReLU DNNs that contain $K+1$ layers, numbered from 0 to K. Here, layer 0 corresponds to the input (in the literature, the input is usually not counted as its own layer) whilst layer K corresponds to the output of the ReLU DNN. The remaining layers $k \in \{1, ..., K-1\}$ are referred to as the hidden layers of the ReLU DNN. The ReLU DNN's structure also includes numerically valued weights W and biases b. The weights W are associated with the connections between nodes in adjacent layers, and biases b are associated with each node in a layer. Each layer $k \in \{0, ..., K\}$ consists of nodes numbered from 1 to n_k.

Let $x^k \in \mathbb{R}^{n_k}$ be the vector corresponding to the values of the nodes in layer k and $x_j^k \in \mathbb{R}$ be the value of the j-th node in layer k, for $j \in \{1, \ldots, n_k\}$. Using this notation, we can refer to the x_j^0 as to the value in the j-th input node and to x_j^K as to the value in the j-th output node. For each hidden layer $k \in \{1, \ldots, K-1\}$, the output vector x^k is computed using the values of the nodes from the previous layer as

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

where σ is a non-linear activation function and $W^k \in \mathbb{R}^{n_k \times n_{k-1}}$ and $b^k \in \mathbb{R}^{n_k}$ are corresponding matrices for the weights and biases for the layer k. The activation function in this 0-1 MILP formulation in all hidden layers is the piecewise-linear ReLU function. Therefore, function $\sigma(x)$ in (1) is defined as $\sigma(x) := \text{ReLU}(0, x) = \max\{0, x\}$ and is applied component-wise in the layers. The output layer K of the ReLU DNN is calculated without an activation function, so we have

$$x^K = W^K x^{K-1} + b^K.$$

The piecewise-linear ReLU activation function is represented by a set of 0-1 MILP constraints following the implementation in Grimstad and Andersson (2019). To achieve this, we consider the following linear equation

$$w^{\mathrm{T}} x + b = x - s, \quad x \geq 0, \quad s \geq 0, \tag{2}$$

where w represent the weights, y the previous layer values, b the bias term, and the output of the ReLU function is divided into a positive part x and a negative part $-s$. Note that a solution to (2) is not unique: if (x, s) is a solution, then $(x+\delta, s+\delta)$, $\delta \geq 0$ is also a solution. We circumvent this problem by requiring at least one of the terms in $(x+\delta, s+\delta)$ to be zero, which corresponds to the case $\delta = 0$. This requirement is achieved with the use of a binary activation variable $z \in \{0, 1\}$ such that the implications $z = 0 \Rightarrow x = 0$ and $z = 1 \Rightarrow s = 0$ hold. Assuming that one can calculate non-negative values U and L such that $-L \leq w^{\mathrm{T}} y + b \leq U$, the ReLU activation function can be implemented using the big-M constraints

$$\begin{aligned} & x \leq Uz, \\ & s \leq -L(1-z), \\ & z \in \{0, 1\}. \end{aligned}$$

After incorporating the above big-M constraints to correspond with each hidden layer node, the following 0-1 MILP surrogate model can be constructed to represent ReLU DNNs:

Input layer bounds

$$L_j^0 \leq x_j^0 \leq U_j^0 \qquad j = 1, ..., n_0, \tag{3a}$$

Hidden ReLU layers

$$\left.\begin{array}{l} W^k x^{k-1} + b^k = x^k - s^k \\ x^k, s^k \geq 0 \end{array}\right\} \qquad k = 1, ..., K-1, \tag{3b}$$

Binary activation variables

$$z_j^k \in \{0, 1\} \qquad k = 1, ..., K-1,\ j = 1, ..., n_k, \tag{3c}$$

Big-M constraints

$$\left.\begin{array}{l} x_j^k \leq U_j^k z_j^k \\ s_j^k \leq -L_j^k (1 - z_j^k) \end{array}\right\} \qquad k = 1, ..., K-1,\ j = 1, ..., n_k, \tag{3d}$$

Output layer

$$W^K x^{K-1} + b^K = x^K, \tag{3e}$$

Output layer bounds

$$L_j^K \leq x_j^K \leq U_j^K \qquad j = 1, ..., n_K. \tag{3f}$$

The above 0-1 MILP model (3) represents a trained ReLU DNN, i.e., the values for the weight and bias matrices (W^k and b^k) are given and hence the model cannot be used for training a ReLU DNN. Values for the bounding terms U_j^k and L_j^k are also provided as parameters. The variables in (3) are the aforementioned non-negative real variables x_j^k and s_j^k and the binary activation variables z_j^k.

It is important to note that the constraint set (3a)-(3f) precisely represents a trained ReLU DNN: for a given input vector x^0, the output layer values (and also the hidden layers values) are the same for both the surrogate model and the original ReLU DNN. However, a solution to a given 0-1 MILP problem is not necessarily unique, and thus, degenerate solutions can exist. If a node in layer k receives zero as its input from the previous layer $k-1$, the corresponding binary variable z_j^k can freely be set to either 0 or 1 without having any effect on the outcome of the model.

3.2 Image classification

We considered the MNIST (Deng, 2012) handwritten digit data set for the training and testing of our ReLU DNN, which we then represented as a 0-1 MILP using the surrogate model (3). Examples of the digit data set can be seen in Figure 1. After the training phase is complete, the values for the parameters in the weight and bias matrices W^k and b^k in model (3) can be extracted from the trained ReLU DNN. We fixed the values of bounding parameters as follows: $L_j^0 = 0$ and $U_j^0 = 1$ in (3a) (pixel value range), $L_j^k = L_j^K = -1000$ and $U_j^k = U_j^K = 1000$ in (3d) and (3f) (arbitrary sufficiently large bounds to not affect the outcome of the model). It is important to highlight that calculating tighter bounds values U_j^k and L_j^k in the hidden layers can significantly improve the computational tractability of the resulting optimization Problem (3). However, the implementation of this procedure is not considered in this thesis.

Just like in the case of the ReLU DNN, the model (3) can be used to classify an input image. To achieve this, the input layer variables x_j^0 are fixed to correspond to an input image, such that x_1^0 equals the value of the first pixel, x_2^0 the value of the second pixel, and so forth, following the same pattern for each subsequent pixel. An artificial objective function (e.g., maximizing any hidden or output node) is added to the model (3), after which the model is optimized. The reason for introducing the artificial objective function in Problem (3) is solely driven by the requirement imposed by the optimization software, which mandates the presence of an objective function in every optimization problem.

It is important to note that each of the variables x_j^k, s_j^k and z_j^k can only obtain one specific value once the input variables x_j^0 have been initialized (disregarding possible degenerate solutions), as discussed in Section 3.1. After the optimization process is complete, the index of the output variable x_j^K, $j \in \{0, \dots, 9\}$ with the highest value corresponds to the predicted digit. We refer to this optimization problem as 0-1 MILP classification.

3.3 Generating optimal input images

In order to utilize the model (3) to solve various problems, we can add additional constraints to it and modify the objective function to represent a desired optimization goal. We considered the following objective function

$$\text{Max. } x_j^K, \quad j \in \{0, ..., 9\}$$

that maximizes the output node value corresponding to a given digit and constructed 10 different optimization problems for every $j \in \{0, ..., 9\}$. Once optimal values for the variables x_j^k, s_j^k and z_j^k have been found by optimizing the problem, the values in the variables x_j^0 corresponding to grayscale values of a pixel can be extracted to produce input images. These images can be seen in Figure 2. One of the interesting highlights one can pinpoint from Figure 2 is that despite containing noise, many of the images still clearly resemble the digit label given by the ReLU DNN. We refer to this optimization problem as optimal input images.

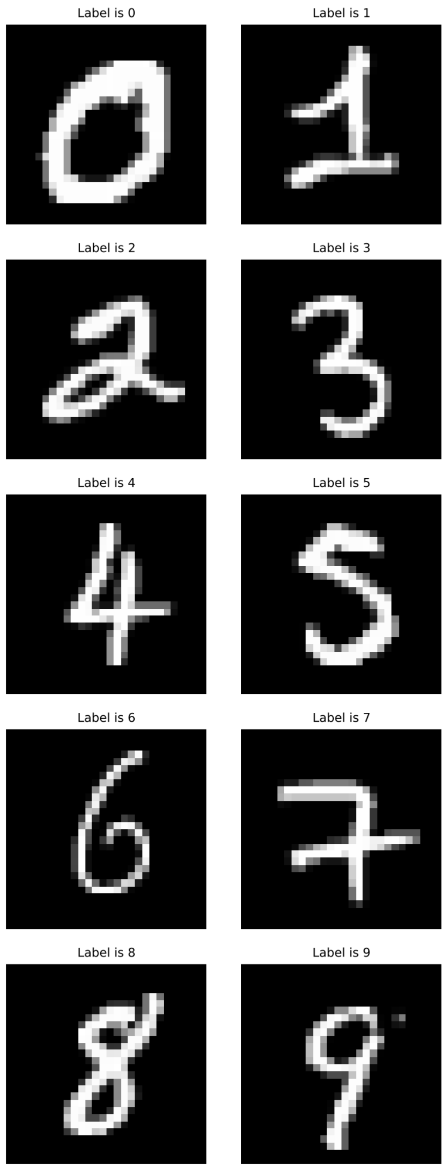

Figure 1: Randomly picked images of each digit from the MNIST handwritten digit data set

3.4 Adversarial images

In this section, we apply model (3) to the problem of adversarial images discussed in Section 2. The model is utilized for generating adversarial digit images with optimally minimal changes with respect to the MNIST training data set.

We assume that an image with its given training label d must be misclassified as the digit $\hat{d} = (d + 5) \bmod 10$, following Table 1. To introduce the aforementioned misclassification condition into model (3), we introduce the constraint in the output layer requiring the value in the output node $x^K_{\hat{d}}$ to be at least 20% higher than in the other output nodes. Due to the flexibility of the surrogate model (3), this constraint

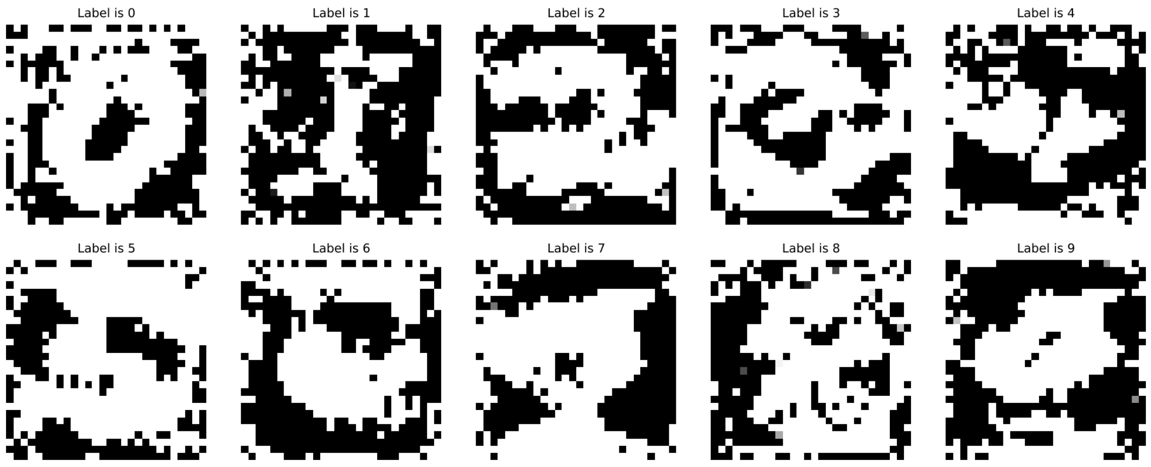

Figure 2: Optimal input images to the optimization problem discussed in section 3.3. The label in each image is the classification given by the ReLU DNN after extracting the variables corresponding to pixel values from the optimization problems

can be implemented by adding the following linear constraints to Problem (3):

$$x^K_{\hat{d}} \geq 1.2 \cdot x^K_j, \quad j \in \{0, ..., 9\} \backslash \{\hat{d}\}. \tag{4}$$

Table 1: Misclassification labels for adversarial images

Actual label	Misclassified label
Digit 0	Digit 5
Digit 1	Digit 6
Digit 2	Digit 7
Digit 3	Digit 8
Digit 4	Digit 9
Digit 5	Digit 0
Digit 6	Digit 1
Digit 7	Digit 2
Digit 8	Digit 3
Digit 9	Digit 4

Regarding the objective function in Problem (3), we considered minimization of the differences between the corresponding pixels' values of the original and adversarial images. We considered two cases by minimizing: 1) the L^1-norm and 2) the L^2-norm, where the generalized L^p-norm for a vector v of length n is defined as

$$\|v\|_p = \left(\sum_1^n v_n^p \right)^{\frac{1}{p}}, \quad p \geq 1. \tag{5}$$

The implementation of the above optimization problems was achieved by introducing additional variables $\delta_j \geq 0$ and adding the following linear constraints into the surrogate model (3):

$$-\delta_j \leq x_j^0 - \hat{x}_j^0 \leq \delta_j, \quad j \in \{1, ..., n_0\}, \tag{6}$$

where x_j^0 and $\hat{x}_j^0$ correspond to a pixel in the original and adversarial image, respectively. With these constraints in place, the L^p-norm can be minimized by adding the objective function Min. $\sum_1^{n_0} \delta_{n_0}^p$ to Problem (3). Note that due to the $\frac{1}{p}$-th power from the definition of (5) being an increasing function, the optimal solution $\delta = (\delta_1, \dots, \delta_{n_0})$ to Min. $(\sum_1^{n_0} \delta_{n_0}^p)^{\frac{1}{p}}$ is also the optimal solution to Min. $\sum_1^{n_0} \delta_{n_0}^p$, and therefore we can neglect the use of the $\frac{1}{p}$-th power in the objective function. Hence, the objective functions for the two problems were

$$L^1\text{-norm:} \quad \text{Min.} \sum_1^{n_0} \delta_{n_0}, \tag{7}$$

$$L^2\text{-norm:} \quad \text{Min.} \sum_1^{n_0} \delta_{n_0}^2. \tag{8}$$

Note that the L^1-norm objective function (7) is linear, whereas the L^2-norm objective function (8) is quadratic. Hence, the former optimization problem is a 0-1 MILP problem and the latter is a 0-1 mixed-integer quadratic programming (0-1 MIQP) problem. We refer to these optimization problems as L^1-norm problem and L^2-norm problem respectively. The optimal adversarial images and the computational time for the optimization problems are discussed more in Section 4.

4 Computational experiments

4.1 Design of the experiments

The 0-1 MILP problem (3) described in Section 3 was implemented using the programming language Julia version 1.8.5 (Julia, 2023). The source code can be found in the GitHub repository ML_as_MO. The adversarial image optimization problems are solved using the Gurobi Optimizer version 10.0.1 (Gurobi-Optimizer, 2023). For computational time analysis and reproducibility, all of the experiments are performed on a MacBook Pro with an M2 processor.

The MNIST data set consists of 70 000 unique 28×28 pixel grayscale images, of which 60 000 are used for training and 10 000 for testing of the ReLU DNN. Each image has a label $j \in \{0, \dots, 9\}$ corresponding to a digit, and the grayscale values are normalized to the range $[0, 1]$ where 0 corresponds to a black pixel and 1 to a white pixel. The ReLU DNN was generated and trained using Flux.jl (Flux.jl, 2023). Within each training cycle, all of the training data is propagated through the ReLU DNN, and subsequently, the internal parameters of the network are adjusted to minimize the discrepancy between the ReLU DNN predictions and the corresponding labels. The ReLU DNN consists of the input and output layers and two hidden layers. The shape of the ReLU DNN is $(784, 32, 16, 10)$, where the number in the list represents the number of nodes in each layer. The numbers of neurons in the input and output layers are fixed to correspond with the number of pixels in an image and the number of classification predictions the ReLU DNN can have, whereas the number of hidden layers, as well as the number of neurons they consist of, are arbitrarily chosen but with the intent of keeping the ReLU DNN relatively small. The ReLU DNN reached an accuracy of $93,31\%$ on the test set after 50 training cycles.

As mentioned previously, we considered four optimization problems: 0-1 MILP classification, optimal input images, the L^1-norm problem, and the L^2-norm problem. However, in this and the subsequent sections, our primary focus was on the L^1-norm and the L^2-norm problems as they have numerous applications as discussed in Section 2.

4.2 Adversarial images

Figure 3 shows adversarial images resulting from the optimization problem described in Section 3.4. These adversarial images are generated using the training set, and specifically training images in Figure 1 as input parameters. The images on the left-hand side of Figure 3 are generated considering the objective function (7) minimizing the L^1-norm distance, and for the images on the right-hand side, the objective function (8) minimizing the L^2-norm distance.

Figure 4 shows the difference in pixels between generated adversarial images presented in Figure 3 and the original images presented in Figure 1. It is important to mention that the grayscale values in each image of Figure 4 are inverted (i.e., pixel value $p \in [0,1]$ becomes $1-p$) to improve the presentation quality. These images visualize the exact alterations required to be made for each original image presented in Figure 1 to cause misclassification as in Figure 3.

From the left-hand side of Figure 4 one can notice that when minimizing over the L^1-norm distance, it is sufficient to change only a few pixels in the original images in Figure 1 to cause misclassification as in Table 1. However, the changes made in the pixels are drastic: in most cases, the pixel value has switched between the two extremes, changing from 0 (black) to 1 (white). Due to this phenomenon, the position of changed pixels is easily detectable by the human eye, and therefore, simply by looking at the left-hand side of Figure 4 one can effortlessly pinpoint the changes made in each image.

However, from the right-hand side of Figure 4 we can notice that when minimizing the L^2-norm distance, almost all pixels in each image have experienced a change in color. Contrary to the L^1-norm problem, the changes are visually a lot more subtle, meaning that the same pixels in the original and adversarial image exhibit a high degree of similarity in their grayscale intensity. Additionally, one will not be able to effortlessly detect the differences between the images on the right-hand side of Figure 4 and in Figure 1: at a quick glance it might not be obvious that any changes have been made. Therefore, in the context of difficult-to-detect adversarial images, the images generated using L^2-norm seem to be more appealing than L^1-norm images.

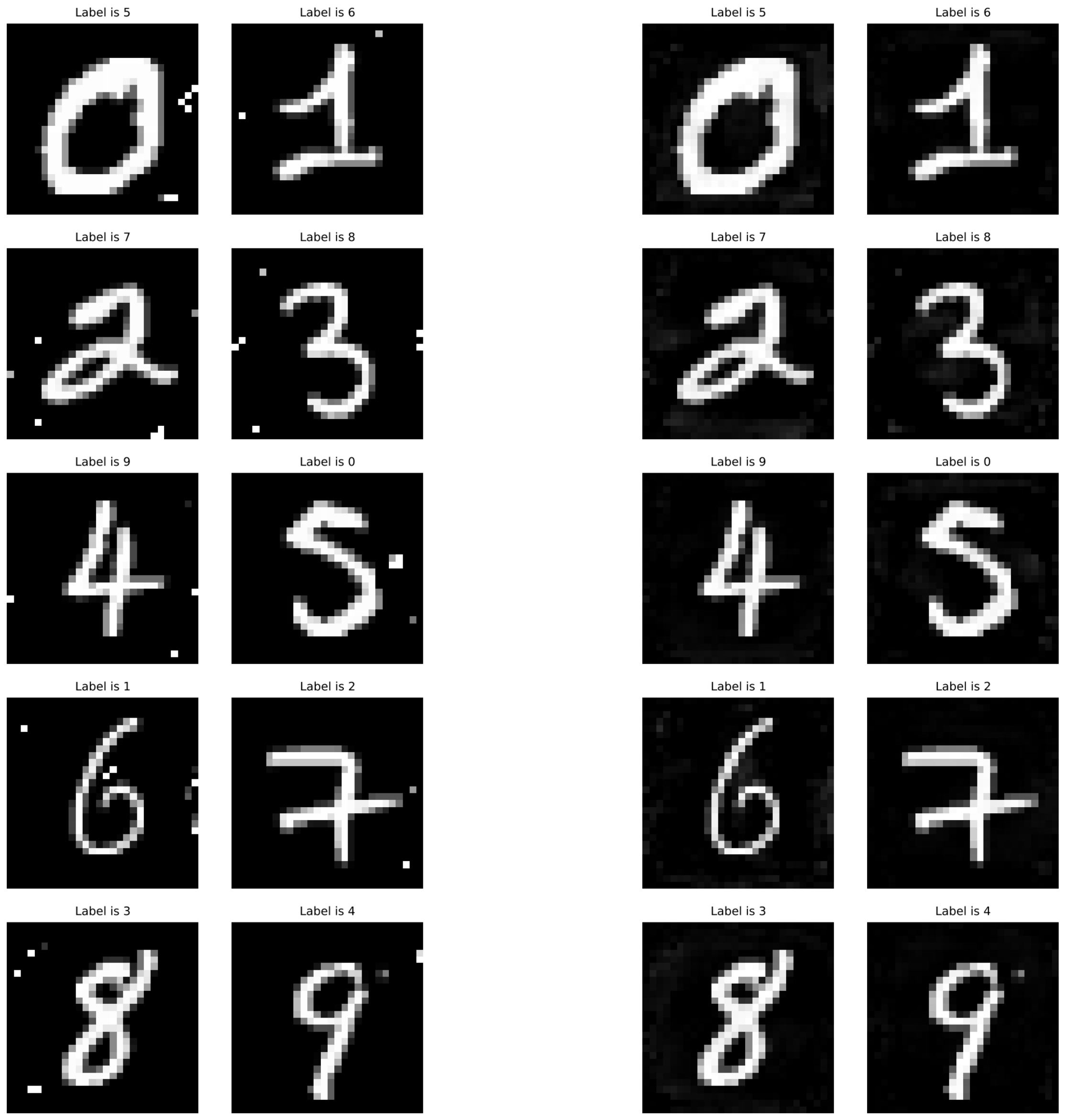

Figure 3: Adversarial images generated for the original images in Figure 1 when minimizing the L^1-norm distance (left-hand side) and the L^2-norm distance (right-hand side) and ensuring misclassification as in Table 1

4.3 Computational time analysis

Table 2 shows the number of variables and constraints for the 0-1 MILP classification problem presented in Section 3.2 and L^1-norm and L^2-norm problems described in Section 3.4 as well as the computational time required to solve the corresponding optimization problems. Table 2 also presents computational time required by the ReLU DNN with the structure described in Section 4.1 and generated using Flux.jl (Flux.jl, 2023) to make a prediction, i.e., propagate forward given input. For each of the optimization problems and ReLU DNN, we considered 100 different images as input. Therefore, the columns titled "Avg. time (s)", "Min. time (s)" and "Max.

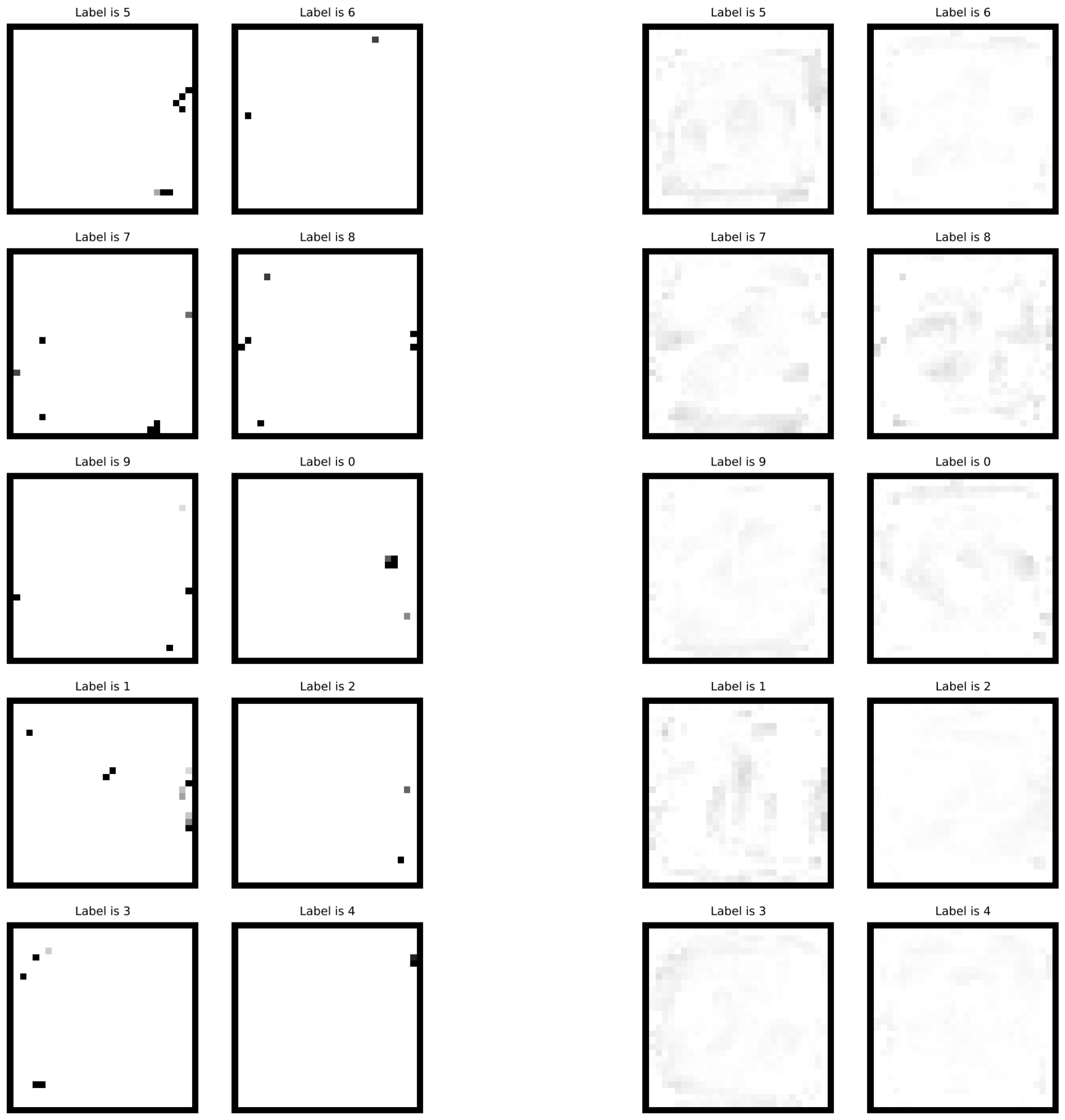

Figure 4: The inverted pixel differences between the original images in Figure 1 and their adversarial images in Figure 3. The left(right)-hand side corresponds to the differences between the original images in Figure 1 and left(right)-hand side images in Figure 3

time (s)" in Table 2 presents the average, minimum, and maximum computational time accordingly over 100 instances.

While approximately two orders of magnitude slower in terms of computational time compared to the ReLU DNN, performing digit classification with the 0-1 MILP model (3) remains computationally fast, with an average solution time of $4,7$ milliseconds. This is due to the forward propagation nature of the model discussed in Section 3.2. The only feasible and hence optimal values for the model variables are found directly by feeding input layer values to the model without the need for further optimization procedures by the optimization software.

More interesting results are regarding the optimization time for the L^1-norm and L^2-norm problems. We can see that the number of variables and constraints in these optimization problems have increased from 958 to 1742 and from 25 760 to 27 338, respectively, compared to the image classification problem discussed in Section 3.2. This increase is due to the presence of the additional variables $\delta_j \geq 0$ needed for the objective functions (7) and (8) and constraints in (6). Furthermore, optimizing the L^1-norm is significantly faster compared to the L^2-norm, with an average solution time of $6,31$ and $107,12$ seconds, respectively. There is also a drastic difference between the minimum and maximum computational time for both problems: the maximum solution time is approximately 143 times greater in the L^1-norm problem and 97 times greater in the L^2-norm problem, compared to the respective minimum solution time.

Table 2: Number of variables and constraint and computational time of the optimization problems, as well as the computational time of image classification by the ReLU DNN

	Variables	Constraints	Avg. time (s)	Min. time (s)	Max. time (s)
ReLU DNN classification	-	-	6,5 $\cdot 10^{-5}$	3,1 $\cdot 10^{-5}$	33,1 $\cdot 10^{-5}$
MILP classification	958	25760	470 $\cdot 10^{-5}$	420 $\cdot 10^{-5}$	670 $\cdot 10^{-5}$
L^1-norm	1742	27338	6,31	0,32	43,12
L^2-norm	1742	27338	107,12	8,13	784,62

5 Discussion and conclusions

In this thesis, we implemented the transformation of a trained ReLU DNN into a 0-1 MILP surrogate model (3) following Grimstad and Andersson (2019). Using the surrogate model, we generated adversarial images based on the MNIST handwritten digit data set. Two sets of adversarial images were generated by minimizing the proposed 0-1 MILP surrogate model over L^1-norm and L^2-norm distances (as defined in (5)) between the original and the adversarial images such that the adversarial image would be misclassified as a predefined digit.

We analyzed the generated adversarial images and solution time required to solve the 0-1 MILP problem with the aforementioned L^1-norm and L^2-norm distances. The original images can be seen in Figure 1, the complete adversarial images in Figure 3, the highlighted pixel changes in Figure 4, and the solution time of the optimization problems in Table 2. The two optimization problems discussed in Section 3.4 that were used for adversarial image generation have some notable differences in both solution time and appearance of the resulting images. The images corresponding to the L^1-norm problem have only a few pixels changed, but the changes are drastic (in most cases, the pixel changes to the extreme opposite, from black to white) and hence, easy to discern. On the other hand, the images corresponding to the L^2-norm problem have a larger number of pixels changed in total, but the changes are considerably more subtle for the human eye. This demonstrates that the L^2-norm adversarial images are more likely to go unnoticed by an outside observer in an image recognition context.

However, despite the visual advantages for the human eye of the L^2-norm problem adversarial images, this approach has a drawback regarding the solution time. The solution time for the L^2-norm problem is on average around 17 times higher when compared to the L^1-norm problem. This illustrates the tradeoff between the two formulations: the L^1-norm problem offers faster image generation whereas the L^2-norm problem offers visually more challenging images to detect. It is important to note that while the solution time for individual images in both cases is relatively small when considering the MNIST data set, this can change for other applications. For instance, processing larger colorful images can be significantly more computationally demanding, and due to the quadratic objective function in the L^2-norm problem, the solution time might also grow exponentially and hence, be too large for a more complex ReLU DNN.

One obvious improvement that can be implemented into the 0-1 MILP surrogate model (3) is the bound tightening techniques discussed in Fischetti and Jo (2018) and Grimstad and Andersson (2019). Including these bound-tightening techniques in the surrogate model can lead to substantial improvements in solution time without any drawbacks to the visual adversarial image quality.

Another possible improvement that can lead to visually better results in the L^1-norm and L^2-norm problems is a more sophisticated adversarial label selection process compared to the arbitrary $\hat{d} = (d + 5) \bmod 10$ that is used in this thesis. For example, if a label that already has a high activation value in a trained ReLU DNN was used as the misclassified label, the adversarial image should (in theory) have fewer differences compared to the original image, which might make the adversarial image harder to detect for the human eye. Another advantage of an improved adversarial label selection process is that we could generate a large number of adversarial images considering a single predefined false label for all digits. For instance, this could be advantageous in scenarios where a malicious attacker aims to manipulate a digit recognition processor to misclassify all images as zeros while avoiding detection by external observers.

Alongside these, one could also modify the alternative objective function and additional constraints to be added into the 0-1 MILP Problem (3) with different goals in mind. The value of 20% chosen arbitrarily in constraint (4) could be increased to generate adversarial images that are misclassified with greater confidence by the ReLU DNN. One could also limit the extent to which a pixel is allowed to change with respect to its surrounding pixels. This could lead to more subtle visible changes and hence more challenging adversarial images to detect for the human eye. The optimal adversarial image approach presented in this thesis could be used as a basis to create more visually subtle adversarial images through minor alterations in the L^1-norm or L^2-norm problems.

Overall, the 0-1 MILP surrogate model is a flexible tool for representing trained

ReLU DNNs as mathematical optimization problems. The flexibility of the model allows us to solve a given problem from alternative viewpoints by adding constraints and changing the objective function. Additionally, we can compare the results of optimizing the 0-1 MILP surrogate model with an emphasis on different key attributes, such as the computational time of the optimal solution or the level of difficulty in detecting the adversarial images, similar to the scenarios explored here. Nevertheless, there are still prospects for further research regarding the 0-1 MILP surrogate model and its applications. As an example, possible future developments could involve the development of an approach to introduce different activation functions or types of layers (e.g., convolutional layers) into the surrogate model. Such developments could provide a strong foundation for future advancements in the surrogate modeling context.

References

A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. *International conference on machine learning*, 29(6):284–293, 2018.

C.M. Bishop. Neural networks and their applications. *Review of scientific instruments*, 65(6):1803–1832, 1994.

R.K. Dash, T.N. Nguyen, K. Cengiz, and A. Sharma. Fine-tuned support vector regression model for stock predictions. *Neural Computing and Applications*, pages 1–15, 2021.

L Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

M. Fischetti and J. Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018.

Flux.jl. https://fluxml.ai/Flux.jl/stable/. 2023.

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

B. Grimstad and H. Andersson. Relu networks as surrogate models in mixed-integer linear programs. *Computers & Chemical Engineering*, 131:106580, 2019.

Gurobi-Optimizer. https://www.gurobi.com/. 2023.

S. Hosder, L.T. Watson, B. Grossman, W.H. Mason, H. Kim, R.T. Haftka, and S.E. Cox. Polynomial response surface approximations for the multidisciplinary design optimization of a high speed civil transport. *Optimization and Engineering*, 2: 431–452, 2001.

Z. Jia, J. Thomas, T. Warszawski, M. Gao, M. Zaharia, and A. Aiken. Optimizing DNN computation with relaxed graph substitutions. *Proceedings of Machine Learning and Systems*, 1:27–39, 2019.

Julia. https://julialang.org/. 2023.

A. Malek and R.S. Beidokhti. Numerical solution for high order differential equations using a hybrid neural network–optimization method. *Applied Mathematics and Computation*, 183(1):260–271, 2006.

ML_as_MO. https://github.com/gamma-opt/ML_as_MO.

V. Nair and G.E. Hinton. Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

D. Parbat and M. Chakraborty. A python based support vector regression model for prediction of COVID19 cases in India. *Chaos, Solitons Fractals*, 138:109942, 2020.

T. Simpson, F. Mistree, J. Korte, and T. Mauery. Comparison of response surface and kriging models for multidisciplinary design optimization. *7th AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization*, page 4755, 1998.

A. Sobester, A. Forrester, and A. Keane. Engineering design via surrogate modelling: a practical guide. *8th symposium on multidisciplinary analysis and optimization*, 2008.

G. Sun, G. Li, Z. Gong, G. He, and Q. Li. Radial basis functional model for multi-objective sheet metal forming optimization. *Engineering Optimization*, 43 (12):1351–1366, 2011.

Z. Zhao, D. Dua, and S. Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.