Aalto-yliopisto
Perustieteiden korkeakoulu

Reformulating neural networks as mathematical progamming problems

Joonatan Linkola

16.5.2023

Advisor: *Nikita Belyak*

Supervisor: *Fabricio Oliveira*

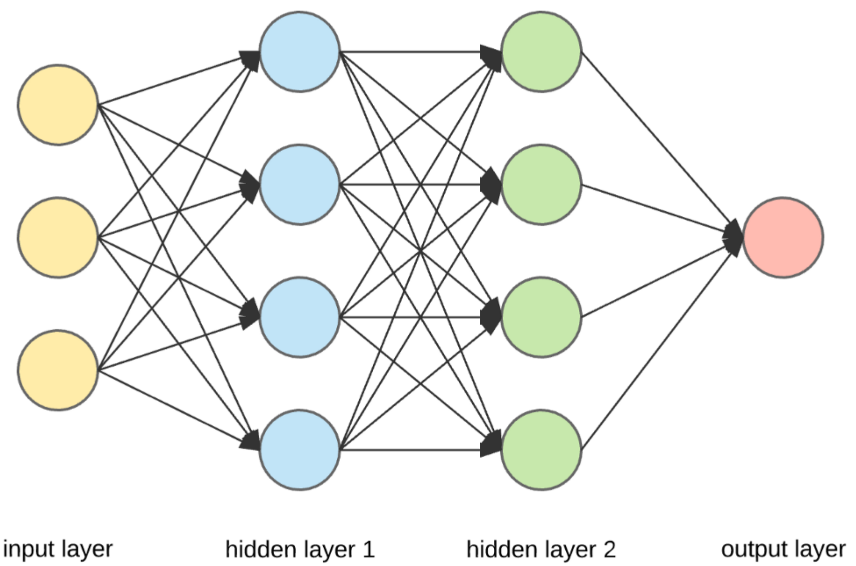
Systeemianalyysin laboratorio

Background

- Neural networks (NN) are popular and versatile machine learning (ML) paradigms

Background

- Neural networks (NN) are popular and versatile machine learning (ML) paradigms



Background

- Neural networks (NN) are popular and versatile machine learning (ML) paradigms
- NNs can be difficult for optimization

Background

- Neural networks (NN) are popular and versatile machine learning (ML) paradigms
- NNs can be difficult for optimization
- Surrogate models – a ReLU NN can be represented as a 0-1 Mixed integer linear program (0-1 MILP)

Background

- Neural networks (NN) are popular and versatile machine learning (ML) paradigms
- NNs can be difficult for optimization
- Surrogate models – a ReLU NN can be represented as a 0-1 Mixed integer linear program (0-1 MILP)
    - $f(x) = \mathrm{ReLU}(x) := \max\{0, x\}$

Background

- Neural networks (NN) are popular and versatile machine learning (ML) paradigms
- NNs can be difficult for optimization
- Surrogate models – a ReLU NN can be represented as a 0-1 Mixed integer linear program (0-1 MILP)
  - $f(x) = \mathrm{ReLU}(x) := \max\{0, x\}$
- The 0-1 MILP surrogate model is versatile

Background

- Neural networks (NN) are popular and versatile machine learning (ML) paradigms
- NNs can be difficult for optimization
- Surrogate models – a ReLU NN can be represented as a 0-1 Mixed integer linear program (0-1 MILP)
    - $f(x) = \mathrm{ReLU}(x) := \max\{0, x\}$
- The 0-1 MILP surrogate model is versatile
    - Objective function for optimization

Background

- Neural networks (NN) are popular and versatile machine learning (ML) paradigms
- NNs can be difficult for optimization
- Surrogate models – a ReLU NN can be represented as a 0-1 Mixed integer linear program (0-1 MILP)
  - $f(x) = \mathrm{ReLU}(x) := \max\{0, x\}$
- The 0-1 MILP surrogate model is versatile
  - Objective function for optimization
  - Additional constraints

Aims

- Develop a model which represents a trained ReLU NN as a 0-1 MILP

Aims

- Develop a model which represents a trained ReLU NN as a 0-1 MILP
- Case study – digit image classification problems

Aims

- Develop a model which represents a trained ReLU NN as a 0-1 MILP
- Case study – digit image classification problems
    - Feature visualization
    - Adversarial images
    - Performance analysis

Some technical notes

- Julia programming language
  - Namely Flux and JuMP libraries
- Gurobi Optimizer for 0-1 MILP optimization problems
- The code was run on a MacBook Pro with M2-processor

Training the ReLU network for digit image classification problems

- MNIST dataset: 28×28 pixel, 60 000 train, 10 000 test
- Network shape: input layer, 2 hidden layers, output layer
  - Nodes at each layer: 784 → 32 → 16 → 10
- Loss function: cross entropy
- Optimizer: ADAM(0.01) (gradient descent)
- 50 training cycles – 93,31% accuracy

0-1 MILP formulation for ReLU networks (Grimstad and Andersson, 2019)

Input layer:

$$L_{\text{in}} \leq x^0 \leq U_{\text{in}}$$

Hidden ReLU layers:

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

$$x^k = 0 \ \vee \ s^k = 0$$

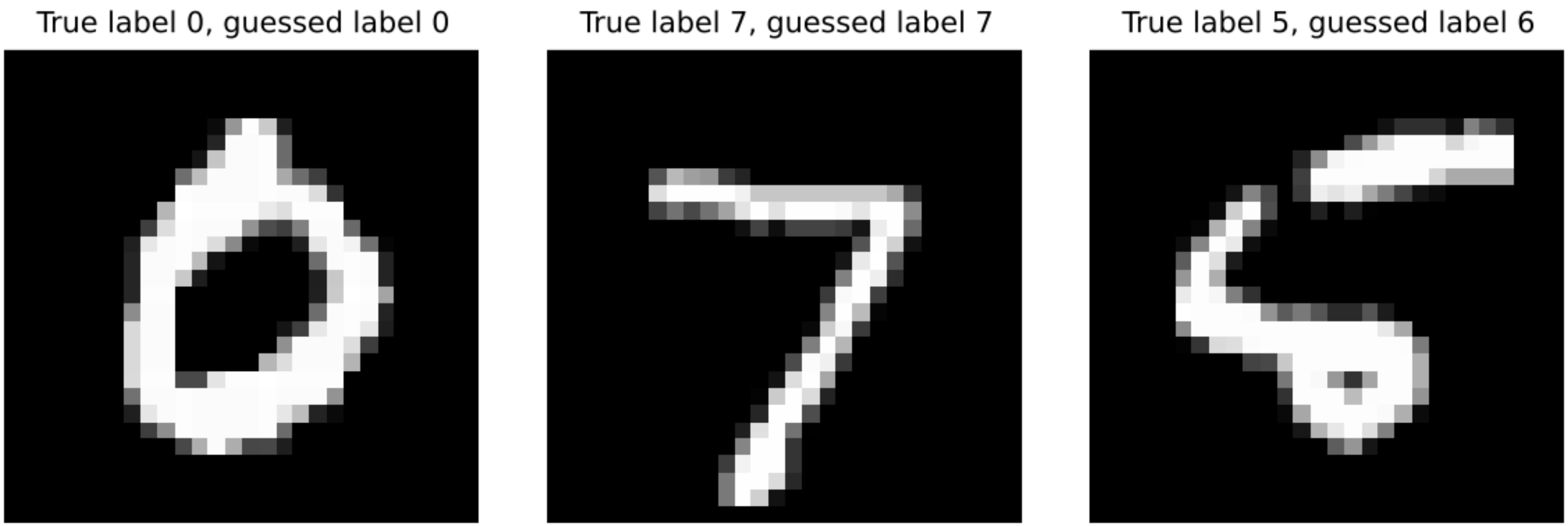
Output layer:

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

$$L_{\text{out}} \leq x^K \leq U_{\text{out}}$$

$\boldsymbol{W}$: weights at each layer
$\boldsymbol{b}$: biases at each layer
$\boldsymbol{x}$: node value at each layer
$\boldsymbol{k} = \{0, \ldots, K\}$: node index

Feature visualization using the surrogate

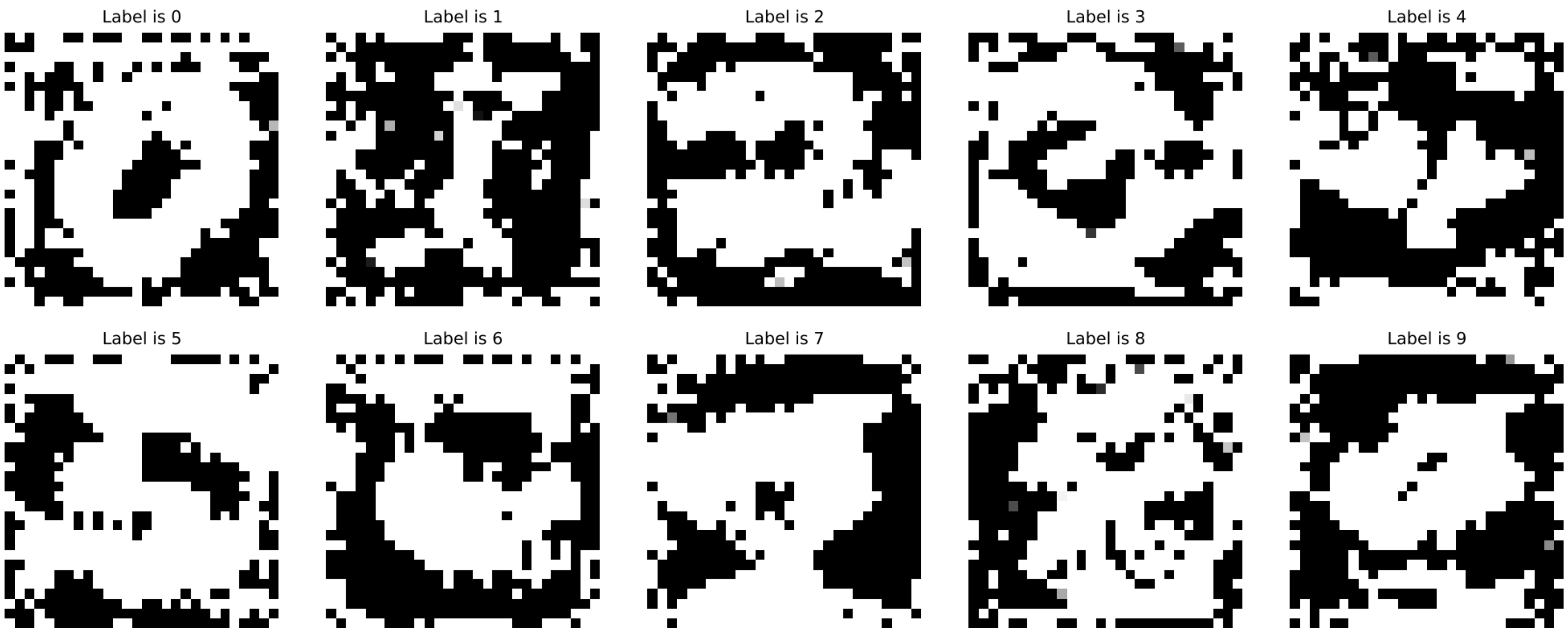


Optimizing the output value

- An additional objective function
    - Maximize an output node corresponding to a digit

Optimizing the output value

- An additional objective function
  - Maximize an output node corresponding to a digit

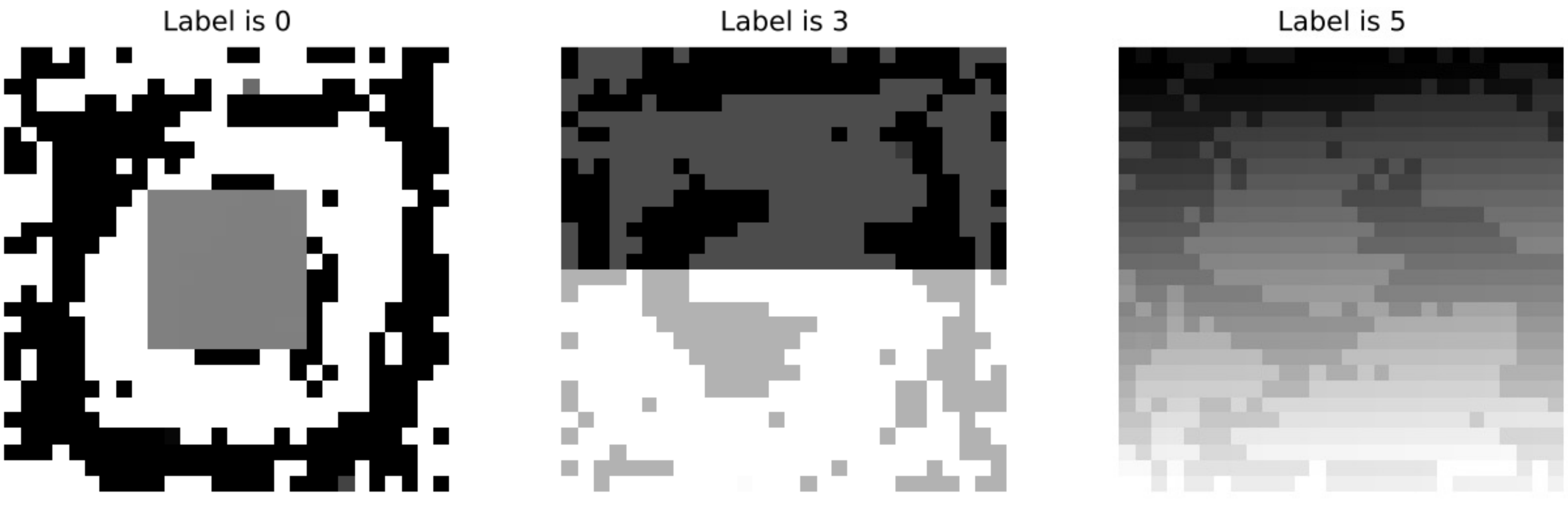


Optimizing the output value with additional constraints

- An additional objective function and constraints
    - Maximize an output node with constraints added

Optimizing the output value with additional constraints

- An additional objective function and constraints
    - Maximize an output node with constraints added



Adversarial images

- Specialized input images with the purpose of confusing the NN

Adversarial images

- Specialized input images with the purpose of confusing the NN
- Images look “normal” to the human eye but cause misclassification

Adversarial images

- Specialized input images with the purpose of confusing the NN
- Images look “normal” to the human eye but cause misclassification
    - Added noise, a few key pixel changes, etc.

Adversarial images

- Specialized input images with the purpose of confusing the NN
- Images look “normal” to the human eye but cause misclassification
    - Added noise, a few key pixel changes, etc.
- Here, the changes to the images are optimally minimal

Building the adversarial examples

- We impose that an image of digit $\tilde{d}$ must be misclassified as $d = (\tilde{d} + 5) \bmod 10$

Building the adversarial examples

- We impose that an image of digit $\tilde{d}$ must be misclassified as $d = (\tilde{d} + 5) \bmod 10$
  - 0 misclassified as 5, 1 as 6, etc.

Building the adversarial examples

- We impose that an image of digit $\tilde{d}$ must be misclassified as $d = (\tilde{d} + 5) \bmod 10$
    - 0 misclassified as 5, 1 as 6, etc.
- Value in the output node corresponding to the digit d must be at least 20% higher than in other nodes

Building the adversarial examples

- We impose that an image of digit $\tilde{d}$ must be misclassified as $d = (\tilde{d} + 5) \bmod 10$
  - 0 misclassified as 5, 1 as 6, etc.
- Value in the output node corresponding to the digit d must be at least 20% higher than in other nodes
- We minimize both L1-norm and L2-norm distances between the original image and the adversarial image

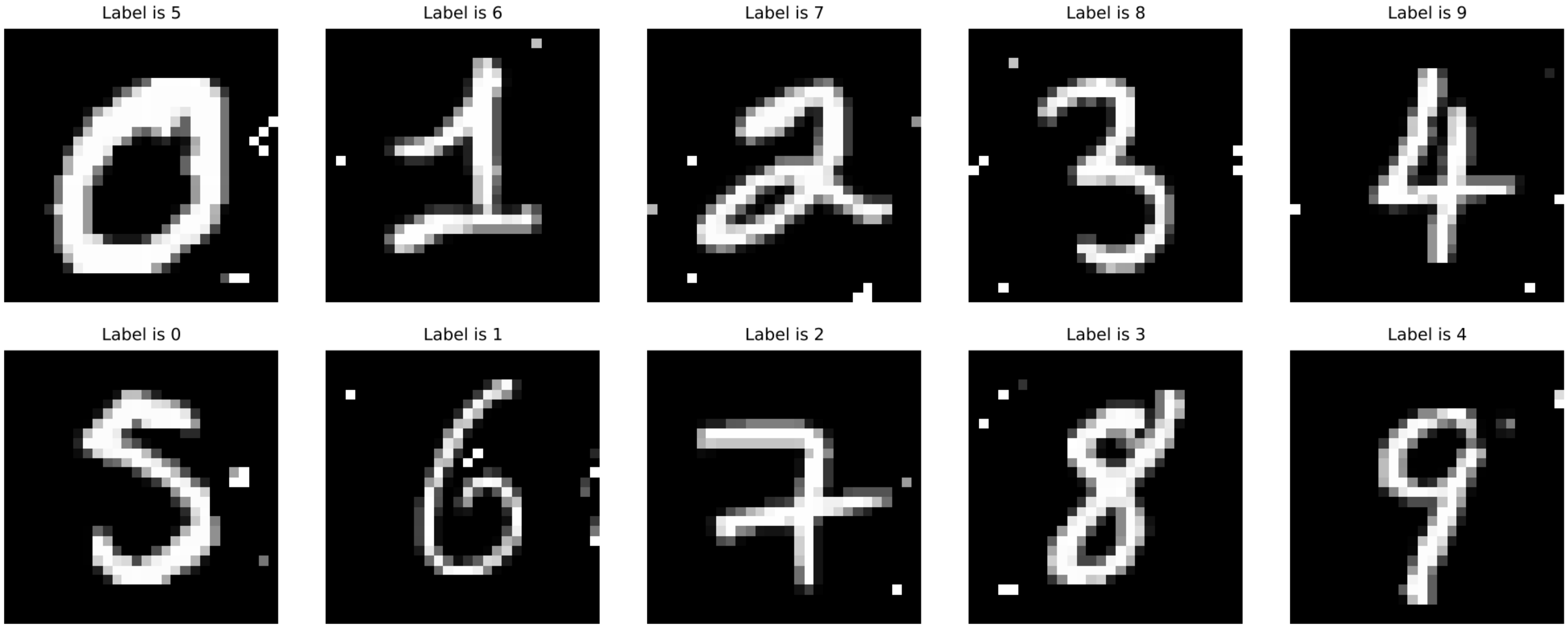
Building the adversarial examples

- We impose that an image of digit $\tilde{d}$ must be misclassified as $d = (\tilde{d} + 5) \bmod 10$
  - 0 misclassified as 5, 1 as 6, etc.
- Value in the output node corresponding to the digit d must be at least 20% higher than in other nodes
- We minimize both L1-norm and L2-norm distances between the original image and the adversarial image
  - L^p-norm: $$x = (x_1, x_2, \ldots, x_n), \quad ||x||_p = \left(\sum_{1}^{n} x_n^p\right)^{\frac{1}{p}}$$

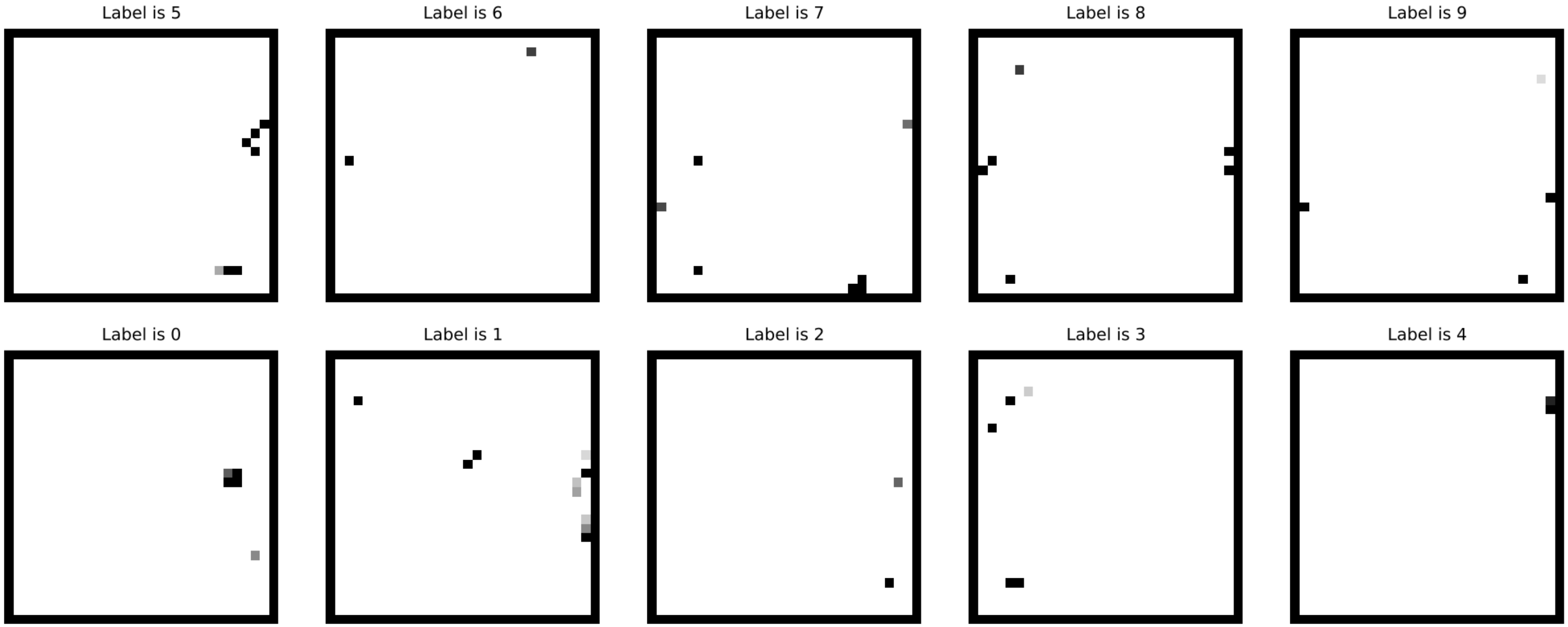
Building adversarial examples

- 1st additional constraint: $\boldsymbol{x_d^K \geq 1.2\, x_j^K, \quad j \in \{0, \ldots, 9\} \backslash \{d\}}$
- Additional variables $\boldsymbol{d_j \geq 0, \quad j \in 1, \ldots, 784}$
- 2nd additional constraint: $\boldsymbol{|x_j^0 - \tilde{x}_j^0| \leq d_j, \quad j \in \{1, \ldots, 784\}}$
- Objective functions:
  - L1-norm $\quad \mathbf{Min.} \displaystyle\sum_{j=1}^{784} \boldsymbol{d_j}$
  - L2-norm $\quad \mathbf{Min.} \displaystyle\sum_{j=1}^{784} \boldsymbol{d_j^2}$

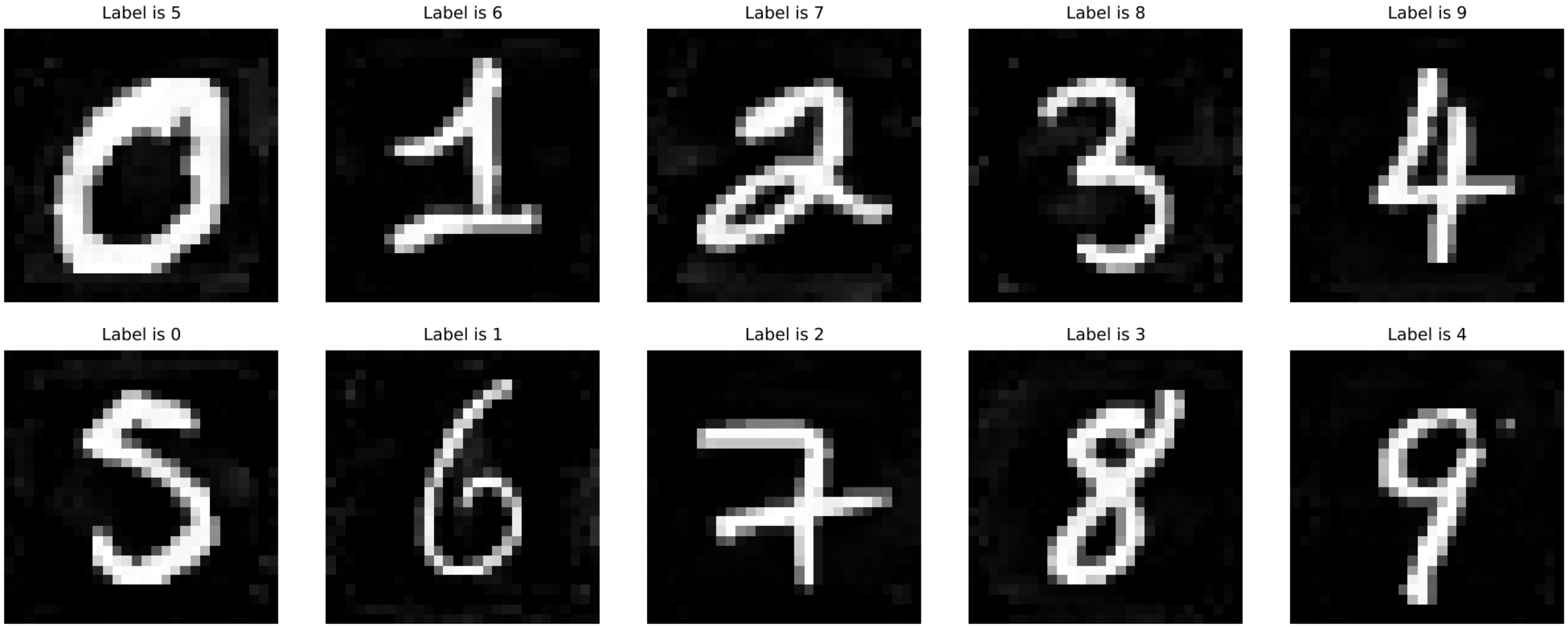
Adversarial images with L1-norm



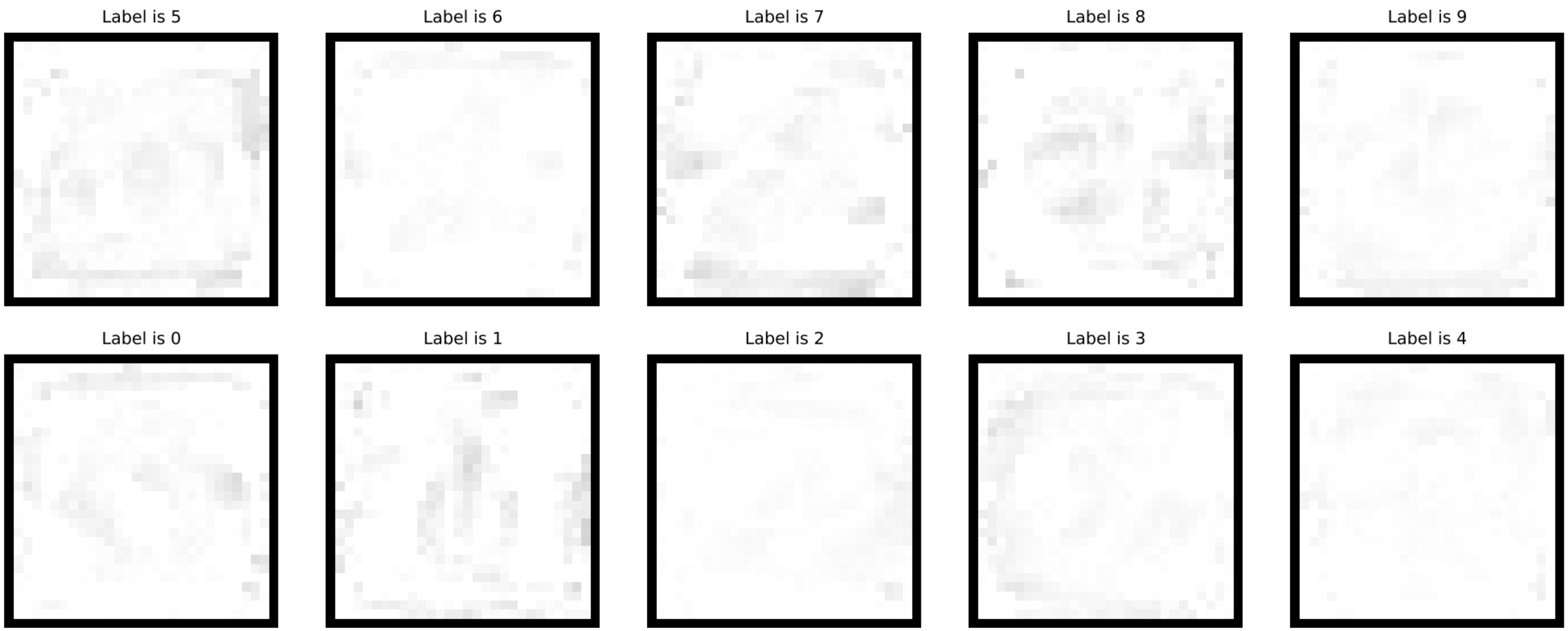
Adversarial images with L1-norm (pixel changes)



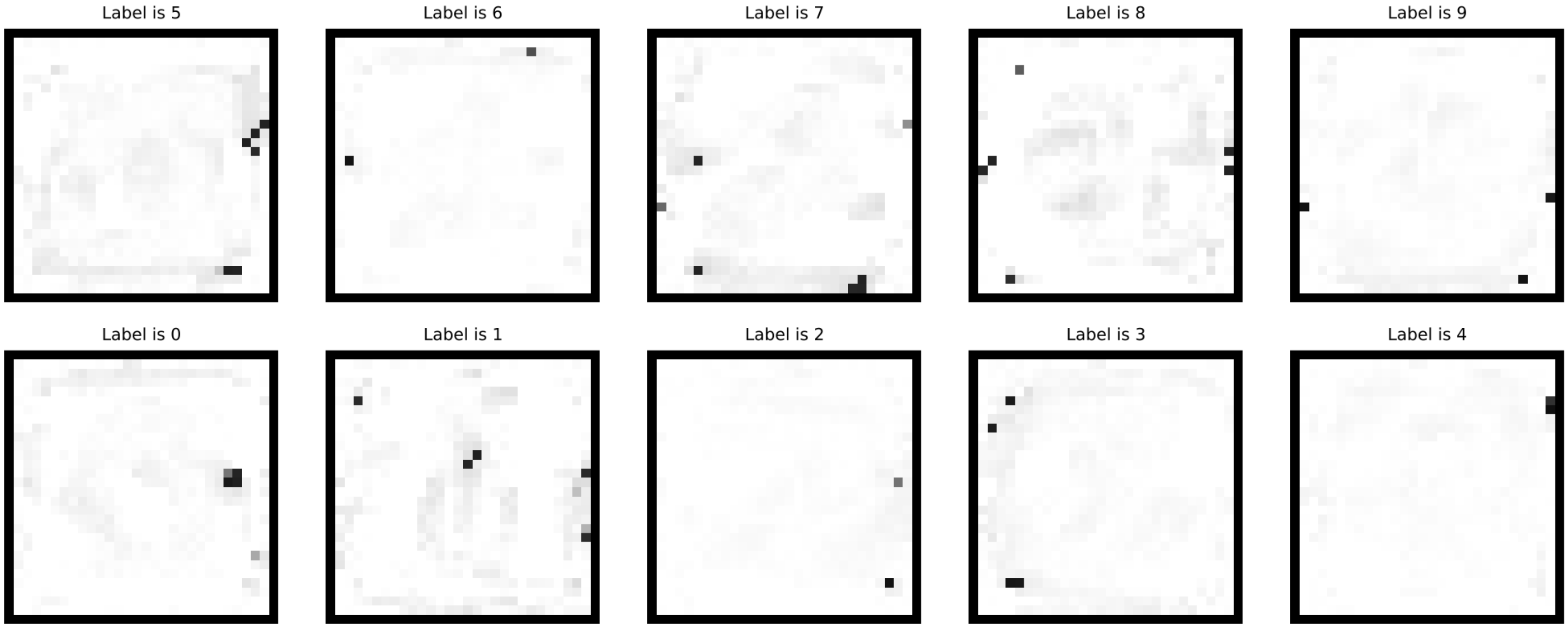
Adversarial images with L2-norm



Adversarial images with L2-norm (pixel changes)



Adversarial images (pixel changes compared)



Performance of 100 optimization cases for each application *

	Variables	Constraints	Avg. time (s)	Min. time (s)	Max. time (s)
Feature visualization	958	25760	0,0047	0,0042	0,0067
Optimal input*	958	25760	1,32	0,51	5,72
L1-norm	1742	27338	6,31	0,32	43,12
L2-norm	1742	27338	107,12	8,13	784,62

* only 10 different optimization cases available