Training Fully Connected Neural Networks is $\exists \mathbb{R}$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

$\exists X_1, \ldots, X_n \in \mathbb{R} : \Phi(X_1, \ldots, X_n)$
Training Fully Connected Neural Networks is \( \exists R \)-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber
Motivation

**Neural Networks**: The most successful tool in artificial intelligence.

AlphaGo vs. Lee Sedol, 2016

photorealistic image generation (StyleGAN, 2019)
Neural Networks

inputs

$x_1$

$x_2$
Neural Networks

Training Fully Connected Neural Networks is $\exists B$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber
Training Fully Connected Neural Networks is \( \exists R \)-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber
Neural Networks

Architecture: directed acyclic graph
(vertices = neurons)
Neural Networks

Architecture: directed acyclic graph
(vertices = neurons)

Weights: on edges
Neural Networks

Architecture: directed acyclic graph
(vertices = neurons)

Weights: on edges

Biases: on hidden neurons
Neural Networks

Architecture: directed acyclic graph (vertices = neurons)

Weights: on edges

Biases: on hidden neurons

Activation Function: $\max\{0, x\}$

ReLU : $\mathbb{R} \rightarrow \mathbb{R}$

$x \mapsto \max\{0, x\}$
Neural Networks

Architecture: directed acyclic graph (vertices = neurons)

Weights: on edges

Biases: on hidden neurons

Activation Function: \( \circ \) = ReLU

ReLU: \( \mathbb{R} \rightarrow \mathbb{R} \)
\( x \mapsto \max\{0, x\} \)
Neural Networks

Architecture: directed acyclic graph (vertices = neurons)

Weights: on edges

Biases: on hidden neurons

Activation Function: $\text{ReLU} = \max\{0, x\}$

ReLU : $\mathbb{R} \rightarrow \mathbb{R}$

$x \mapsto \max\{0, x\}$
Neural Networks

Architecture: directed acyclic graph (vertices = neurons)

Weights: on edges

Biases: on hidden neurons

Activation Function: \( \text{ReLU} = \max\{0, x\} \)

ReLU : \( \mathbb{R} \to \mathbb{R} \)

ReLU(3 \cdot 2 + 4 \cdot 1 + (−3)) = 7

ReLU(3 \cdot 2 + 4 \cdot 1 + (−3)) = 7

Training Fully Connected Neural Networks is \( \exists \mathbb{R} \)-Complete
Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber
Neural Networks

**Architecture:** directed acyclic graph (vertices = neurons)

**Weights:** on edges

**Biases:** on hidden neurons

**Activation Function:**
\[ \text{ReLU}(x) = \max\{0, x\} \]

ReLU : \( \mathbb{R} \rightarrow \mathbb{R} \)

ReLU : \( x \mapsto \max\{0, x\} \)

ReLU : \( \mathbb{R} \rightarrow \mathbb{R} \)

ReLU : \( x \mapsto \max\{0, x\} \)
Neural Networks

**Architecture:** directed acyclic graph (vertices = neurons)

**Weights:** on edges

**Biases:** on hidden neurons

**Activation Function:** $\text{ReLU} = \max\{0, x\}

Neural network realizes a function:

$$f(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}^2$$

weights + biases parametrize $f$

ReLU($3 \cdot 2 + 4 \cdot 1 + (-3)$) = 7

$7 \cdot 1 = 7$

$7 \cdot (-4) = -28$
Training Neural Networks

Question:
- The weights and biases $\Theta$ parametrize the function $f(\cdot, \Theta)$.
  ~ What are good values for $\Theta$?
Question:
- The weights and biases $\Theta$ parametrize the function $f(\cdot, \Theta)$.
  $\Rightarrow$ What are good values for $\Theta$?

Training Data:
List of points $(x_i; y_i) \in \mathbb{R}^2 \times \mathbb{R}^2$:
- $x_i \in \mathbb{R}^2$: input values
- $y_i \in \mathbb{R}^2$: labels = desired output values
Training Neural Networks

Question:
- The weights and biases $\Theta$ parametrize the function $f(\cdot, \Theta)$.
  $\rightsquigarrow$ What are good values for $\Theta$?

Training Data:
List of points $(x_i; y_i) \in \mathbb{R}^2 \times \mathbb{R}^2$:
- $x_i \in \mathbb{R}^2$: input values
- $y_i \in \mathbb{R}^2$: labels = desired output values

Optimize: Choose $\Theta$ such that overall fitting error is minimal.
For all $i$: $y_i \approx f(x_i, \Theta)$
Training Neural Networks

Question:
- The weights and biases $\Theta$ parametrize the function $f(\cdot, \Theta)$.
  $\rightsquigarrow$ What are good values for $\Theta$?

Training Data:
List of points $(x_i; y_i) \in \mathbb{R}^2 \times \mathbb{R}^2$:
- $x_i \in \mathbb{R}^2$: input values
- $y_i \in \mathbb{R}^2$: labels = desired output values

Optimize: Choose $\Theta$ such that overall fitting error is minimal.
For all $i$: $y_i \approx f(x_i, \Theta)$

Best case: $y_i = f(x_i, \Theta)$
Training Fully Connected Neural Networks is \( \exists \mathbb{R} \)-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

**Decision Problem**

**TRAIN-NN:**

**Input:**
- network architecture
- \( n \) data points \((x_i; y_i)\)

**Question:** Are there weights and biases \( \Theta \), such that:

\[
y_i = f(x_i, \Theta) \quad \forall i \in \{1, \ldots, n\}
\]
Decision Problem

**TRAIN-NN:**

**Input:**
- network architecture
- $n$ data points $(x_i; y_i)$

**Question:** Are there weights and biases $\Theta$, such that:

$$y_i = f(x_i, \Theta) \quad \forall i \in \{1, \ldots, n\}$$

No optimization, just a yes/no-question.
Training Fully Connected Neural Networks is $\exists \mathbb{R}$-Complete

Decision Problem

**TRAIN-NN:**

**Input:**
- network architecture
- $n$ data points $(x_i; y_i)$

**Question:** Are there weights and biases $\Theta$, such that:

$$y_i = f(x_i, \Theta) \quad \forall i \in \{1, \ldots, n\}$$

No optimization, just a yes/no-question.

A little more general:
- a *cost function* $\text{cost}(\cdot)$
- a *threshold* $\gamma$

$$\sum_{i=1}^{n} \text{cost}(y_i, f(x_i, \Theta)) \leq \gamma?$$
How hard can it be?

**NP-hard** in many settings:
- binary classification (Blum, Rivest 1992)
- sigmoid activation function (Jones 1997, ...)
- single hidden neuron with ReLU (Geol et al. 2020)
How hard can it be?

**NP-hard** in many settings:
- binary classification (Blum, Rivest 1992)
- sigmoid activation function (Jones 1997, . . .)
- single hidden neuron with ReLU (Geol et al. 2020)

**NP-membership** in simple settings:
- single output neuron, one ReLU layer (Arora et al. 2016)
- step activation functions (Khalife, Basu 2022)
How hard can it be?

∃R-complete for:

- one hidden layer, three outputs, identity activation function
  (Abrahamsen, Kleist, Miltzow 2021)
How hard can it be?

$\exists R$-complete for:

- one hidden layer, three outputs, identity activation function
  (Abrahamsen, Kleist, Miltzow 2021)

Their proof relies on particularly difficult to train network architectures.

$\leadsto$ This is not a practical setting.
Our Result

**Theorem:** Training neural networks is $\exists R$-complete, for
Our Result

Theorem: Training neural networks is $\exists R$-complete, for
- exactly one hidden layer,
Our Result

**Theorem:** Training neural networks is $\exists \mathbb{R}$-complete, for

- exactly one hidden layer,
- two inputs, two outputs,

in NP for single output.
Our Result

**Theorem:** Training neural networks is $\exists \mathbb{R}$-complete, for

- exactly one hidden layer,
- two inputs, two outputs,
- fully connected network architecture,

in NP for single output

often used (as a building block)
in practical architectures
Our Result

**Theorem:** Training neural networks is $\exists^R$-complete, for

- exactly one hidden layer,
- two inputs, two outputs,
- fully connected network architecture,
- only 13 different labels, in NP for single output
- often used (as a building block) in practical architectures
- common in classification tasks
Our Result

**Theorem:** Training neural networks is \( \exists \mathbb{R} \)-complete, for

- exactly one hidden layer,
- two inputs, two outputs,
- fully connected network architecture,
- only 13 different labels,
- (more or less) any training error \( \gamma \),

in NP for single output

- often used (as a building block) in practical architectures
- common in classification tasks

We prove \( \gamma = 0 \). Add inconsistent training data for \( \gamma > 0 \).
Our Result

**Theorem:** Training neural networks is $\exists R$-complete, for

- exactly one hidden layer,
- two inputs, two outputs,
- fully connected network architecture,
- only 13 different labels,
- (more or less) any training error $\gamma$,
- ReLU activation function,

in NP for single output

often used (as a building block)
in practical architectures

common in classification tasks

We prove $\gamma = 0$. Add inconsistent training data for $\gamma > 0$.

by far the most used in practice
Existential Theory of the Reals

**Definition: (ETR)**

**EXISTENTIAL THEORY OF THE REALS:**
All true sentences of the form

$$\exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n).$$

$$\varphi =$$ quantifier-free formula of polynomial equations and inequalities
Existential Theory of the Reals

**Definition:** (ETR)

**Existential Theory of the Reals:**
All true sentences of the form

$$\exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n).$$

$$\varphi =$$ quantifier-free formula of polynomial equations and inequalities

Solving systems of non-linear equations and inequalities.
Existential Theory of the Reals

**Definition:** (ETR)

**EXISTENTIAL THEORY OF THE REALS:**

All true sentences of the form

$$\exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n).$$

$\varphi = $ quantifier-free formula of polynomial equations and inequalities

**Example:**

$$\varphi(X, Y) :\equiv X^2 + Y^2 \leq 1$$

Solving systems of non-linear equations and inequalities.
Existential Theory of the Reals

Definition: (ETR)

**Existential Theory of the Reals:**

All true sentences of the form

\[ \exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n). \]

\( \varphi = \) quantifier-free formula of polynomial equations and inequalities

Example:

\[ \varphi(X, Y) \equiv X^2 + Y^2 \leq 1 \]
\[ \land \quad Y \geq 2X^2 - 1 \]

Solving systems of non-linear equations and inequalities.
Existential Theory of the Reals

**Definition: (ETR)**

**Existential Theory of the Reals:**

All true sentences of the form

\[ \exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n). \]

\( \varphi = \) quantifier-free formula of polynomial equations and inequalities

Solving systems of non-linear equations and inequalities.

**Example:**

\[ \varphi(X, Y) :\equiv X^2 + Y^2 \leq 1 \]
\[ \wedge \quad Y \geq 2X^2 - 1 \]

\( \exists X, Y \in \mathbb{R} : \varphi(X, Y) \) is true
Training Fully Connected Neural Networks is $\exists R$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

Complexity Class $\exists R$

Base Problem: ETR
Decide whether $\exists X \in \mathbb{R}^n : \varphi(X)$ is true.
Complexity Class \( \exists R \)

Base Problem: ETR
Decide whether \( \exists X \in \mathbb{R}^n : \varphi(X) \) is true.
Training Fully Connected Neural Networks is $\exists R$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

**Complexity Class $\exists R$**

Base Problem: ETR
Decide whether $\exists X \in \mathbb{R}^n : \varphi(X)$ is true.

The complexity class $\exists R$ contains all problems that reduce to ETR.
Training Fully Connected Neural Networks is $\exists \mathbb{R}$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

Complexity Class $\exists \mathbb{R}$

Base Problem: ETR
Decide whether $\exists X \in \mathbb{R}^n : \varphi(X)$ is true.

The complexity class $\exists \mathbb{R}$ contains all problems that reduce to ETR.

$\exists \mathbb{R}$-complete $\iff$ equivalent to ETR (under polynomial time transformations)
Complexity Class $\exists R$

Base Problem: ETR
Decide whether $\exists X \in \mathbb{R}^n : \phi(X)$ is true.

The complexity class $\exists R$ contains all problems that reduce to ETR.

$\exists R$-complete $\iff$ equivalent to ETR
(under polynomial time transformations)
Practical Implications
Practical Implications

Problems \textbf{in }P:\]

- Efficient algorithms in theory and practice.
Practical Implications

**NP-complete** problems:

- No efficient algorithms in theory. (assuming $\text{NP} \neq \text{P}$)
- Highly optimized off-the-shelf tools can solve large instance to optimality.
Practical Implications

$\exists R$-complete problems:

- Exponential time algorithms in theory. However, useless in practice.

- Gradient descent often works reasonably well. But: No guarantees on time and quality.
Practical Implications

**PSPACE-complete** problems:

- No general purpose tools.
- \( P = \text{NP} = \exists R = \text{PSPACE} \) is possible, but considered unlikely.
$\exists R$-Complete Problems

Art Gallery Problem

Recognition of Unit Disk Graphs

Packing

... and many more geometric problems
∃R-Membership

\[ \Leftrightarrow \text{TRAIN-NN is at most as difficult as ETR} \]

**Goal:** Express \text{TRAIN-NN} as an ETR formula.
Training Fully Connected Neural Networks is \( \exists \mathbb{R} \)-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

\( \exists \mathbb{R} \)-Membership

\( \rightarrow \text{TRAIN-NN is at most as difficult as ETR} \)

**Goal:** Express TRAIN-NN as an ETR formula.

\[ \exists w_1, \ldots, b_1, \ldots \in \mathbb{R} : y_1 = f(x_1, \Theta) \land \ldots \land y_n = f(x_n, \Theta) \]

weights biases

formula checking that training data is fit exactly
$\exists R$-Hardness

$\exists R$-Hardness is at least as difficult as ETR

Express ETR formula as an instance of $\text{TRAIN-NN}$.

**Step 1:** Simplify formula.

$\text{ETR} \sim \text{ETR-NN}$

**Step 2:** $\text{ETR-NN} \sim \text{TRAIN-NN}$
Training Fully Connected Neural Networks is $\exists R$-Complete
Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

$\exists R$-Hardness

$\leadsto$ TRAIN-NN is at least as difficult as ETR

Express ETR formula as an instance of TRAIN-NN.

**Step 1:** Simplify formula.

ETR $\leadsto$ ETR-NN

- **Values:** $\exists X, \ldots \in [-1, 1] : \varphi(X)$
- **Constraints:**
  
  \[
  X + Y = Z \\
  XY + X + Y = 0 \quad \text{(nonlinear)} \\
  X \geq 0 \\
  X = 1
  \]
**∃R-Hardness**

∃R-Hardness implies TRAIN-NN is at least as difficult as ETR

Express ETR formula as an instance of TRAIN-NN.

**Step 1:** Simplify formula.

ETR $\leadsto$ ETR-NN

- **Values:** $\exists X, \ldots \in [-1, 1] : \varphi(X)$
- **Constraints:**
  - $X + Y = Z$
  - $XY + X + Y = 0$ (nonlinear)
  - $X \geq 0$
  - $X = 1$

**Step 2:** ETR-NN $\leadsto$ TRAIN-NN

Geometric construction
Geometry I

**Recall**: Neural network realizes a function $f(\cdot, \Theta)$. How does it look like?
Geometry I

**Recall:** Neural network realizes a function \( f(\cdot, \Theta) \).

How does it look like?

\[
f(\cdot, \Theta) : \mathbb{R} \to \mathbb{R}
\]

\[
x \mapsto \text{ReLU}(w_1 x + b) \cdot w_2
\]
Geometry I

**Recall:** Neural network realizes a function $f(\cdot, \Theta)$.

How does is look like?

$$f(\cdot, \Theta) : \mathbb{R} \rightarrow \mathbb{R}$$

$$x \mapsto \text{ReLU}(x \cdot 0) \cdot 1$$
Geometry I

Recall: Neural network realizes a function $f(\cdot, \Theta)$.
How does is look like?

$$f(\cdot, \Theta) : \mathbb{R} \rightarrow \mathbb{R}$$

$$x \mapsto \text{ReLU}(1x + 0) \cdot 2$$
Geometry I

**Recall:** Neural network realizes a function $f(\cdot, \Theta)$. How does it look like?

$$f(\cdot, \Theta) : \mathbb{R} \rightarrow \mathbb{R}$$

$$x \mapsto \text{ReLU}(1x + 0) \cdot (-2)$$
Geometry I

Recall: Neural network realizes a function $f(\cdot, \Theta)$. How does is look like?

$$f(\cdot, \Theta) : \mathbb{R} \rightarrow \mathbb{R}$$

$$x \mapsto \text{ReLU}(1x - 1) \cdot (-2)$$
Geometry I

Recall: Neural network realizes a function $f(\cdot, \Theta)$. How does it look like?

$$f(\cdot, \Theta) : \mathbb{R} \rightarrow \mathbb{R}$$

$$x \mapsto \text{ReLU}(\frac{1}{2}x - 1) \cdot (-2)$$
Geometry I

Recall: Neural network realizes a function $f(\cdot, \Theta)$. How does it look like?

$f(\cdot, \Theta): \mathbb{R} \rightarrow \mathbb{R}$

$x \mapsto \text{ReLU}(\frac{1}{2}x - 1) \cdot (-2)$

Breakpoint is determined only by first weight and bias.

Second weight only for scaling.

$f(\cdot, \Theta)$ is continuous and piecewise linear.
Geometry II

Question: Two outputs?
Geometry II

**Question:** Two outputs?

Separate functions, one per output.

All functions have the same breakpoint!
Geometry III

Question: Two inputs?

\[ f(\cdot, \Theta) : \mathbb{R} \rightarrow \mathbb{R} \]

\[ x \mapsto \text{ReLU}(w_1 x + b) \cdot w_2 \]
Training Fully Connected Neural Networks is $\exists \mathbb{R}$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

Geometry III

$f(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}$

$x \mapsto \text{ReLU}(w_{1,1} x_1 + w_{1,2} x_2 + b) \cdot w_2$
Training Fully Connected Neural Networks is $\exists \mathbb{R}$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

Geometry III

$$f(\cdot, \Theta) : \mathbb{R}^2 \rightarrow \mathbb{R}$$

$$x \mapsto \text{ReLU}(w_{1,1}x_1 + w_{1,2}x_2 + b) \cdot w_2$$

$$w_{1,1}x_1 + w_{1,2}x_2 + b = 0$$
Training Fully Connected Neural Networks is $\exists \mathbb{R}$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

Geometry IV

$f_1(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}$

$f_2(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}$

$f(\cdot, \Theta) : \mathbb{R}^2 \to \mathbb{R}^2$

$x \mapsto (f_1(x, \Theta), f_2(x, \Theta))$
More hidden neurons:

- Each ReLU neuron contributes exactly one breakline.
- $f(\cdot, \Theta)$ is the sum of all individual continuous piecewise linear functions.
- Same breaklines in $f_1$ and $f_2$. 
Encoding ETR as a Neural Network

**Goal:** $\text{ETR-NN} \sim \text{TRAIN-NN}$
Encoding ETR as a Neural Network

**Goal:** $\text{ETR-NN} \sim \text{TRAIN-NN}$

**Given:** variables, constraints

**Find:** data points, integer $m$

**Such that:** formula true $\iff$ trainable with $m$ ReLUs
Encoding ETR as a Neural Network

**Goal:** $\text{ETR-NN} \rightsquigarrow \text{TRAIN-NN}$

| Given: variables constraints | Find: data points integer $m$ | Such that: formula true trainable with $m$ ReLUs |

![Graph](image)
Encoding ETR as a Neural Network

**Goal:** ETR-NN $\sim$ TRAIN-NN

<table>
<thead>
<tr>
<th><strong>Given:</strong></th>
<th>variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>constraints</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Find:</strong></th>
<th>data points</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer $m$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Such that:</strong></th>
<th>formula true</th>
</tr>
</thead>
<tbody>
<tr>
<td>trainable with $m$ ReLUs</td>
<td></td>
</tr>
</tbody>
</table>

**Recall:** #ReLUs = #breakpoints
Encoding ETR as a Neural Network

**Goal:** $ETR$-NN $\leadsto$ $TRAIN$-NN

**Given:** variables, constraints  
**Find:** data points, integer $m$  
**Such that:** formula true $\iff$ trainable with $m$ ReLUs

\[ y \]

not collinear $\leadsto$ at least one ReLU  
Possible with 1 ReLU.

**Recall:** $\#\text{ReLUs} = \#\text{breakpoints}$
Encoding ETR as a Neural Network

**Goal:** ETR-NN $\sim$ TRAIN-NN

<table>
<thead>
<tr>
<th><strong>Given:</strong></th>
<th>variables</th>
<th>constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Find:</strong></td>
<td>data points</td>
<td>integer $m$</td>
</tr>
<tr>
<td><strong>Such that:</strong></td>
<td>formula true</td>
<td>trainable with $m$ ReLUs</td>
</tr>
</tbody>
</table>

$y \sim$ not collinear $\Rightarrow$ at least one ReLU

Possible with 1 ReLU.
Possible with more ReLUs.

**Recall:** #ReLU = #breakpoints
Encoding Variables

**Task:** Encode a value $X \in [-1, 1]$. 
Encoding Variables

**Task:** Encode a value $X \in [-1, 1]$.

Fit with 4 ReLUs:
$\sim$ 4 breakpoints
Encoding Variables

**Task:** Encode a value $X \in [-1, 1]$.

Fit with 4 ReLUs:

$\sim 4$ breakpoints
Encoding Variables

**Task:** Encode a value $X \in [-1, 1]$. 

Fit with 4 ReLUs: $\sim 4$ breakpoints
Encoding Variables

**Task:** Encode a value $X \in [-1, 1]$.

Fit with 4 ReLUs:

$\sim 4$ breakpoints
Encoding Variables

Task: Encode a value $X \in [-1, 1]$.

Fit with 4 ReLUs: $\sim 4$ breakpoints
Encoding Variables

**Task:** Encode a value $X \in [-1, 1]$. 

Fit with 4 ReLUs: $\sim 4$ breakpoints
Encoding Variables

**Task:** Encode a value \( X \in [-1, 1] \).

**Idea:** The slope encodes the value.

Minimum slope is 1, we enforce a maximum slope of 3:

\( \sim \) Interpret slopes in \([1, 3]\) as values in \([-1, 1]\).
Encoding Variables

**Task:** Encode a value $X \in [-1, 1]$.

**Idea:** The slope encodes the value.
Minimum slope is 1, we enforce a maximum slope of 3:
$\mapsto$ Interpret slopes in $[1, 3]$ as values in $[-1, 1]$. 

Fit with 4 ReLUs: $\mapsto$ 4 breakpoints

**Levee**
ITA: argine
DE: Deich
Linear Constraints

**Question:** How to encode constraints involving $X$ and $Y$?
Linear Constraints

Question: How to encode constraints involving $X$ and $Y$?

- impossible in one dimension
Linear Constraints

Question: How to encode constraints involving $X$ and $Y$?

- impossible in one dimension
- levees intersect in two dimensions
Question: How to encode constraints involving $X$ and $Y$?

- impossible in one dimension
- levees intersect in two dimensions
- Add a data point in intersection to encode a linear constraint.
Nonlinear Constraint

**Task:** Encode a nonlinear relation.
Nonlinear Constraint

**Task:** Encode a nonlinear relation.

Fit with 5 ReLUs:

5 breakpoints
Training Fully Connected Neural Networks is $\exists \mathbb{R}$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

Nonlinear Constraint

**Task:** Encode a nonlinear relation.

Fit with 5 ReLUs:

$\sim$ 5 breakpoints
Nonlinear Constraint

**Task:** Encode a nonlinear relation.

Fit with 5 ReLUs:

Possible, but dimensions need to share one breakpoint.
Training Fully Connected Neural Networks is $\exists \mathbb{R}$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

Nonlinear Constraint

**Task:** Encode a nonlinear relation.

Fit with 5 ReLUs: $\sim$ 5 breakpoints

Possible, but dimensions need to share one breakpoint.
Nonlinear Constraint

**Task:** Encode a nonlinear relation.

Fit with 5 ReLUs: \( \sim 5 \) breakpoints

Possible, but dimensions need to share one breakpoint.

\[
\begin{align*}
3 &= \frac{3}{s_1} + \frac{3}{s_2} \\
&\sim 3
\end{align*}
\]
Nonlinear Constraint

**Task:** Encode a nonlinear relation.

Fit with 5 ReLUs: 5 breakpoints

Possible, but dimensions need to share one breakpoint.

\[
\begin{align*}
\frac{3}{s_1} + \frac{3}{s_2} &\sim 3 \\
\sim &\quad 3 = \frac{3}{s_1} + \frac{3}{s_2} \\
&\quad s_1 s_2 + s_1 + s_2 = 0
\end{align*}
\]
Nonlinear Constraint

**Task:** Encode a nonlinear relation.

Fit with 5 ReLUs:

\[ \sim \to 5 \text{ breakpoints} \]

Possible, but dimensions need to share one breakpoint.

\[ \frac{3}{s_1} + \frac{3}{s_2} \]

\[ \sim \to 3 = \frac{3}{s_1} + \frac{3}{s_2} \]

\[ \sim \to \quad s_1s_2 + s_1 + s_2 = 0 \]
Training Fully Connected Neural Networks is \( \exists \mathbb{R} \)-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber
Training Fully Connected Neural Networks is $\exists R$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber
Training Fully Connected Neural Networks is $\exists \mathbb{R}$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

Top Down View

$X_1 + X_2 = X_4$

$X_1 X_3 + X_1 + X_3 = 0$

more levees

double levee

$X_1$ $X_2$ $X_3$ $X_4$ ...

levees
Training Fully Connected Neural Networks is $\exists \mathbb{R}$-Complete

Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, Simon Weber

- Top Down View

There are weights and biases $\Theta$ exactly fitting all data points $\iff$ the ETR instance is true.

The diagram illustrates the equations:

1. $X_1 + X_2 = X_4$
2. $X_1X_3 + X_1 + X_3 = 0$

The points $=1$, $=2$, $=\ldots$ represent solutions to these equations.
Questions?
Questions?

Thank you!