Delirious Representations Enhancing Predictive Systems with Flexible Numeric and Symbolic Domain Integration

Improving Algorithms Using Gradients

Bernhard Gstrein, Armin Biere

Optimization Algorithms

- You want to synthesize a circuit
- You have a good idea of your system's architecture
- In order to make it optimal, you set its parameters
  - Number (int, float)
  - LUT entries
  - Shut parts of your system on or off

Search Difficult

- Complex systems render search difficult
- Solution: Local search + heuristics
  - Are those heuristics really good?
  - Inflexible (heuristics might not work anymore if you change too many things)
  - Have to handle huge search space

Example for Difficult Search: Lookup-Table

- Consider lookup-tables
- 2-LUT: $2^{2^2} = 16$ possible truth tables
- 4-LUT: $2^{2^4} = 65536$ possible truth tables
- Imagine you have lots of LUTs in your system

x	y	out
0	0	a
0	1	b
1	0	c
1	1	d

Our Proposal

- We propose enhancement of algorithms
- Idea: Itentify differentiable parts and optimize them using gradients
- The gradients give a good direction of where to go in parameter space

- System with parameters $f(c, x)$
- Metric g
- $x \leftarrow x - \eta \frac{\partial g(f(c,x))}{\partial x}$

Making Logic Differentiable

- Boolean values 0, 1, and operations: NOT, AND, and OR
- Make differentiable
  - Express boolean values in the range [0, 1]
  - NOT(x) becomes $1 - x$
  - AND(x, y) becomes xy
  - OR(x, y) becomes $x + y - xy$
- We can chain arithmetic NOT, AND, and OR arbitrarily many times and still stay in the range [0, 1]

Example: Lookup-Table

x	y	out
0	0	a
0	1	b
1	0	c
1	1	d

$$(\overline{x} \wedge \overline{y} \wedge a) \vee (\overline{x} \wedge y \wedge b) \vee (x \wedge \overline{y} \wedge c) \vee (x \wedge y \wedge d)$$

Making Lookup-Tables Differentiable

x	y	out
0	0	a
0	1	b
1	0	c
1	1	d

$$(\overline{x} \wedge \overline{y} \wedge a) \vee (\overline{x} \wedge y \wedge b) \vee (x \wedge \overline{y} \wedge c) \vee (x \wedge y \wedge d)$$

$$\begin{aligned} \text{OR}(\text{OR}(\text{OR}(\text{AND}(\text{AND}(\text{NOT}(x), \text{NOT}(y)), a), \\ \text{AND}(\text{AND}(\text{NOT}(x), y), b)), \\ \text{AND}(\text{AND}(x, \text{NOT}(y)), c)), \text{AND}(\text{AND}(x, y), d)) \end{aligned}$$

Enhanced Optimization Algorithm

Algorithm Original: Optimize $g(f(c, x))$

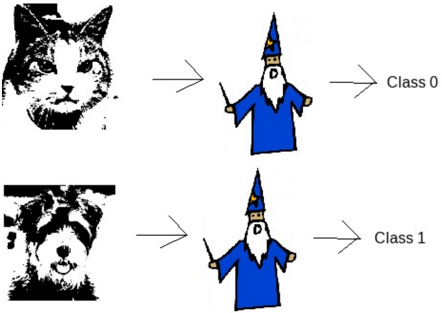
for iter **do**
  Optimize c
  Optimize x
  Do other things
end for

Algorithm Our Proposal: Optimize $g(f(c, x))$

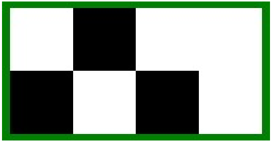
Initialize $x \in [0, 1]$ randomly
for iter **do**
  Optimize c
  $x \leftarrow x - \eta \frac{\partial g(f(c,x))}{\partial x}$
  $\text{clip}(x, [0, 1])$
  Do other things
end for
Round x

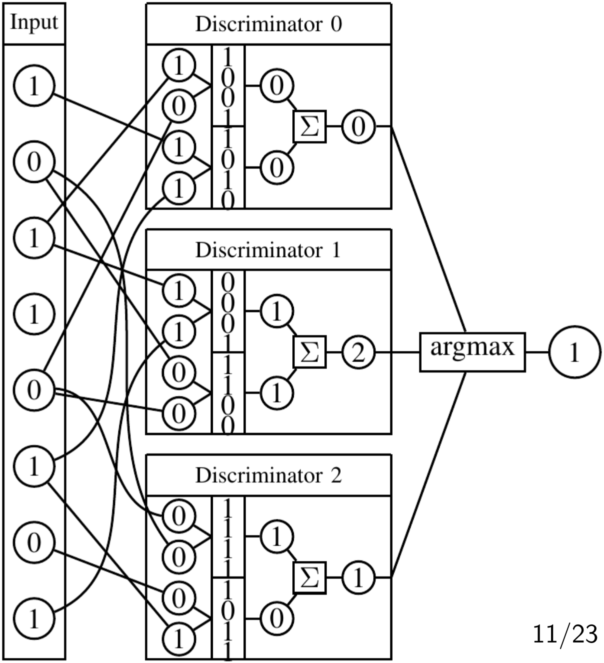
The WiSARD classification system

- **Wi**lkie, **S**tonham, and **A**leksander's **R**ecognition **D**evice [1]
- Image classification system developed in the 1980s
  - Input: Black-and-white image
  - Output: Class *k*
- Interesting because
  - Is a circuit
  - Is based on lookup-tables
  - Comes with symbolic learning algorithm



WiSARD Inference

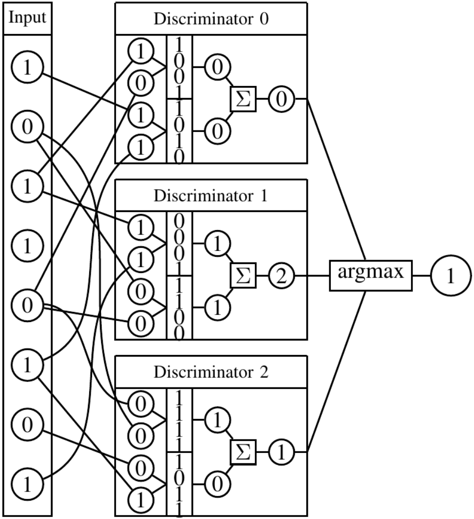


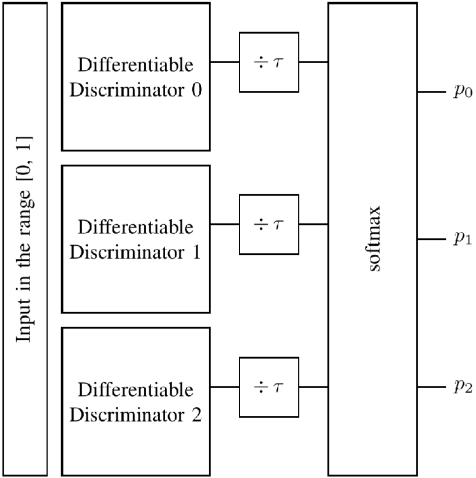


WiSARD Symbolic Training Algorithm

- LUT entries that are indexed by the training set are set to 1
  - Memorize patterns from the training set
- For details, we refer to our paper
- Performs well
  - Has been used for industrial deployment in the 1980s
  - Still getting attention nowadays [2]

Differentiable WiSARD





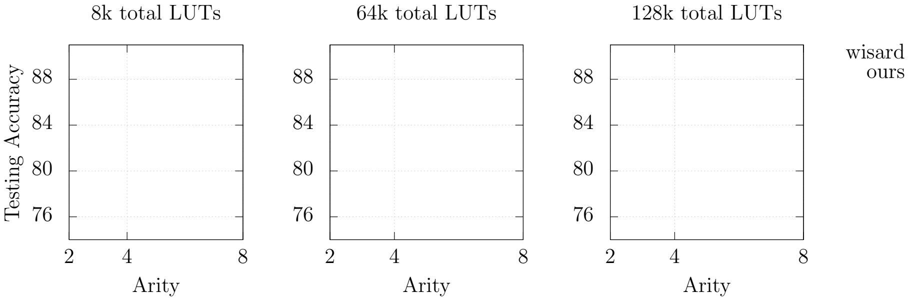
Training WiSARD using our scheme

Algorithm WiSARD training using gradients

Initialize c: k discriminators, n LUTs per discriminator, random LUT connections
Initialize params: random LUT parameters
for each image and label **do**
  Forward pass image
  Loss $\leftarrow$ Difference between actual label and prediction
  params $\leftarrow$ params $- \eta \frac{\partial L(f(c,\text{params}))}{\partial \text{params}}$
  Clip LUT parameters to range [0, 1]
end for
Round params

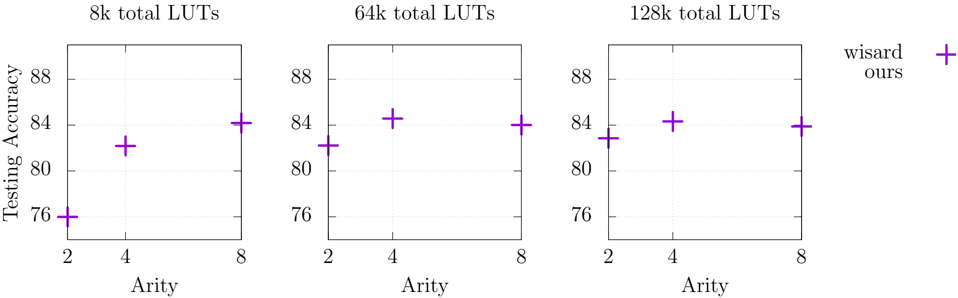
Results: Cybersecurity Dataset

- 593 input features, 2 classes
- Baseline neural network: 86.87%



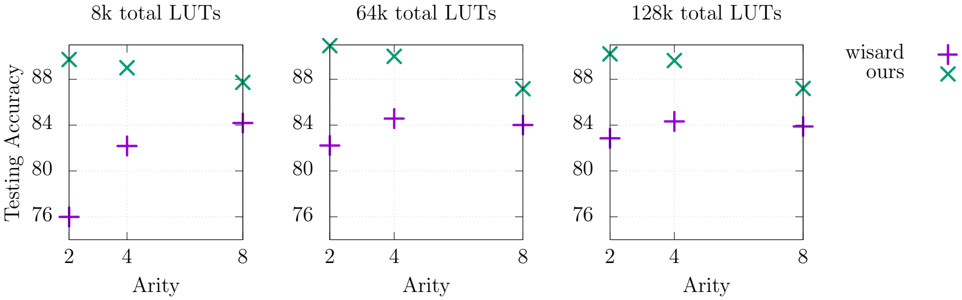
Results: Cybersecurity Dataset

- 593 input features, 2 classes
- Baseline neural network: 86.87%



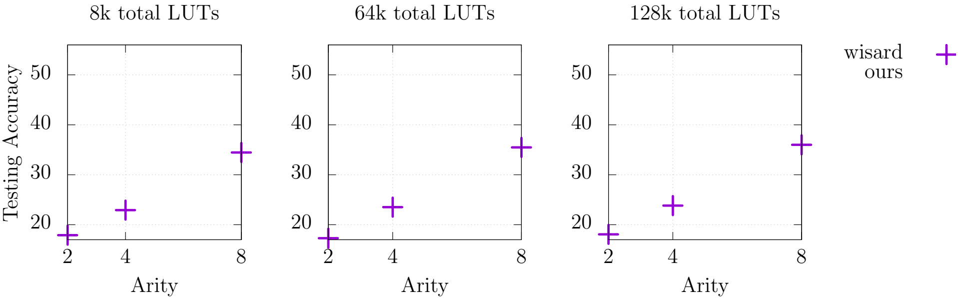
Results: Cybersecurity Dataset

- 593 input features, 2 classes
- Baseline neural network: 86.87%



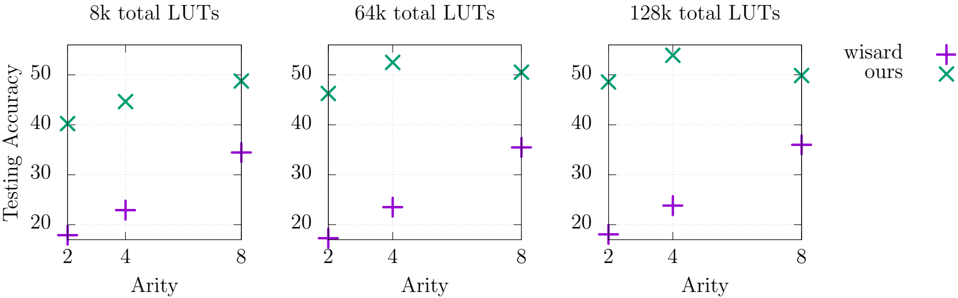
Results: CIFAR-10 Dataset

- 32×32 color images, 10 classes
- $32 \times 32 \times 3 \times 4 = 12288$ input features
- Baseline neural network: 61.25%



Results: CIFAR-10 Dataset

- 32×32 color images, 10 classes
- $32 \times 32 \times 3 \times 4 = 12288$ input features
- Baseline neural network: 61.25%



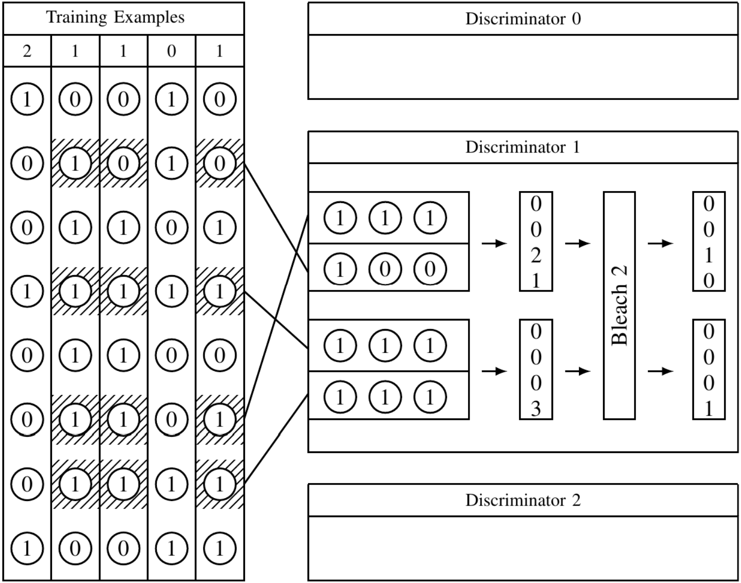
Summary

- We propose improving algorithms using gradients
  - Identify components that can be made differentiable
  - For those components, let gradients do the search
- We have seen example where gradients vastly outperform purely symbolic algorithm
- Next step: Apply this method somewhere else

Tseiting Encoding for Lookup-Tables

$$
\begin{aligned}
v_0 &= \text{NOT}(x), \\
v_1 &= \text{NOT}(y), \\
v_2 &= \text{AND}(\text{AND}(v_0, v_1), a), \\
v_3 &= \text{AND}(\text{AND}(v_0, y), b), \\
v_4 &= \text{AND}(\text{AND}(x, v_1), c), \\
v_5 &= \text{OR}(\text{OR}(v_2, v_3), v_4), \\
\text{LUT2}(x, y) &= \text{OR}(v_5, \text{AND}(\text{AND}(x, y), d)). \qquad (1)
\end{aligned}
$$

WiSARD Symbolic Training Algorithm



References I

[1] I. Aleksander, W. Thomas, and P. Bowden, "WISARD · a radical step forward in image recognition," *Sensor review*, vol. 4, no. 3, pp. 120–124, 1984.

[2] Z. Susskind *et al.*, "Weightless neural networks for efficient edge inference," in *Proceedings of the international conference on parallel architectures and compilation techniques*, 2022, pp. 279–290.