Training Binarized Neural Networks using MIP and CP

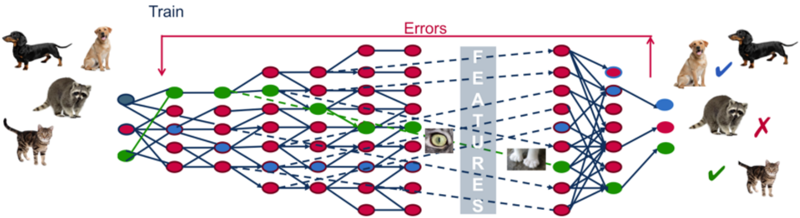
Rodrigo Toro Icarte León Illanes Margarita P. Castro
Andre A. Cire Sheila A. McIlraith J. Christopher Beck

UNIVERSITY OF TORONTO
VECTOR INSTITUTE

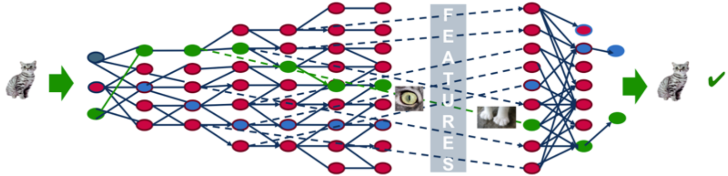
CP 2019
October 4

Binarized Neural Networks

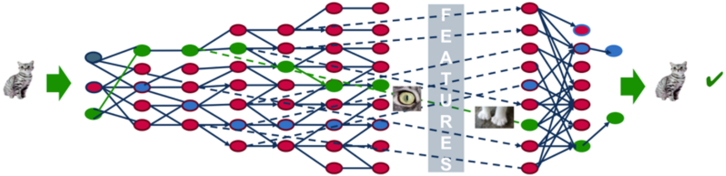
Deep learning



Deep learning



Deep learning



Binarized Neural Networks (BNNs)

- BNNs are NNs with binary weights and activations.
- Similar performance to standard deep learning.
- More efficient (w.r.t. energy and memory) at deploy time.

How to train BNNs

Supervised learning

Objective

Learn a function that maps inputs to outputs from examples.

Supervised learning

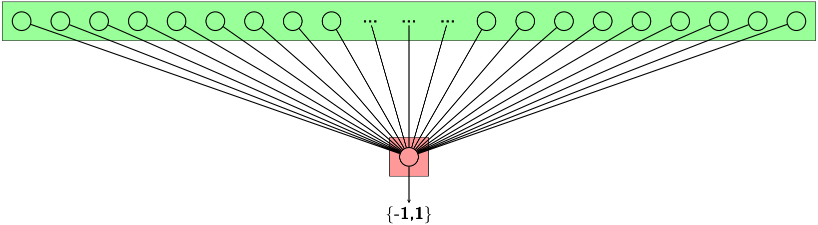
Objective

Learn a function that maps inputs to outputs from examples.

5 0 4 1 8 6 2 7 3 9

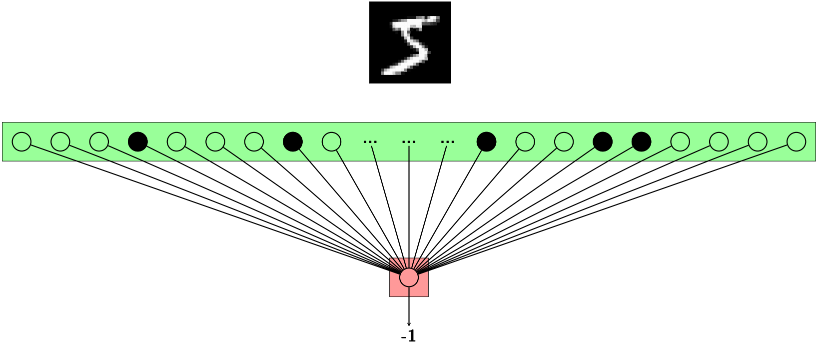
Example: Digit recognition.

The binarized perceptron



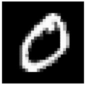
$$n = \begin{cases} +1 & \text{if } \sum_i w_i \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_i \in \{-1, 0, 1\}.$$

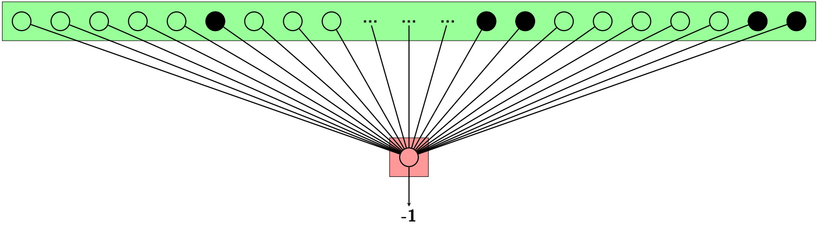
The binarized perceptron



$$n = \begin{cases} +1 & \text{if } \sum_i w_i \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_i \in \{-1, 0, 1\}.$$

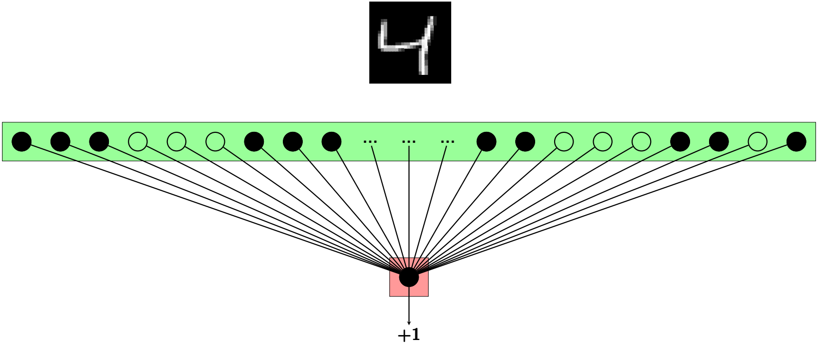
The binarized perceptron





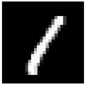
$$n = \begin{cases} +1 & \text{if } \sum_i w_i \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_i \in \{-1, 0, 1\}.$$

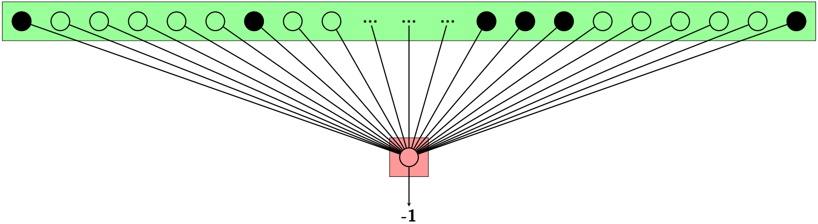
The binarized perceptron



$$n = \begin{cases} +1 & \text{if } \sum_i w_i \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_i \in \{-1, 0, 1\}.$$

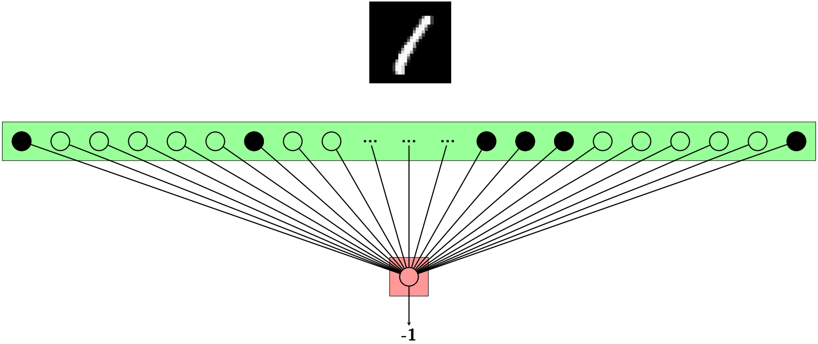
The binarized perceptron





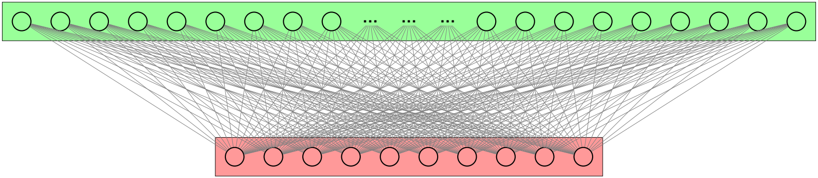
$$n = \begin{cases} +1 & \text{if } \sum_i w_i \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_i \in \{-1, 0, 1\}.$$

The binarized perceptron



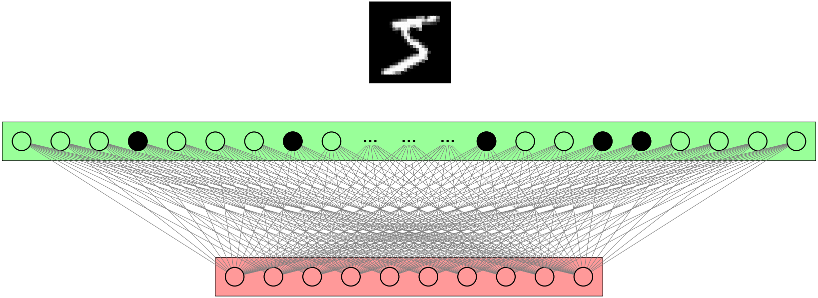
How can we use the perceptron for digit recognition?

The binarized perceptron



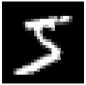
$$n_j = \begin{cases} +1 & \text{if } \sum_i w_{ij} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

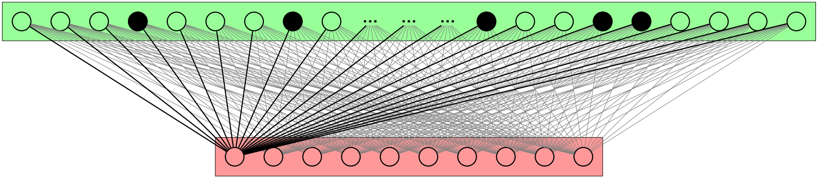
The binarized perceptron



$$n_j = \begin{cases} +1 & \text{if } \sum_i w_{ij} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

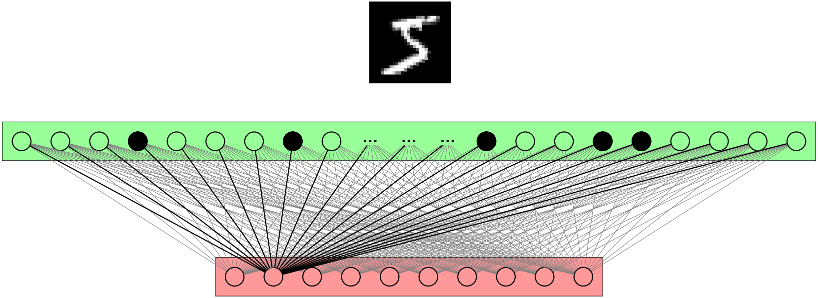
The binarized perceptron





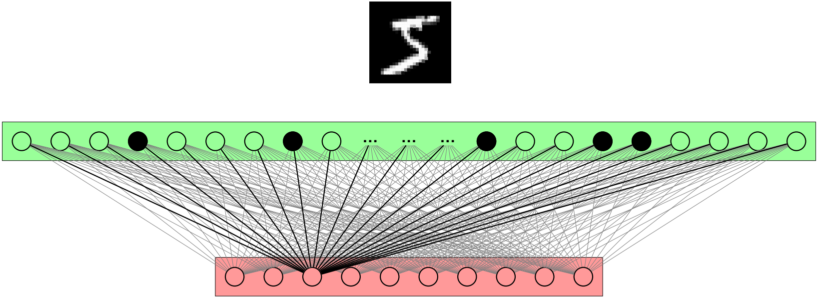
$$n_0 = \begin{cases} +1 & \text{if } \sum_i w_{i0} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

The binarized perceptron



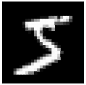
$$n_1 = \begin{cases} +1 & \text{if } \sum_i w_{i1} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

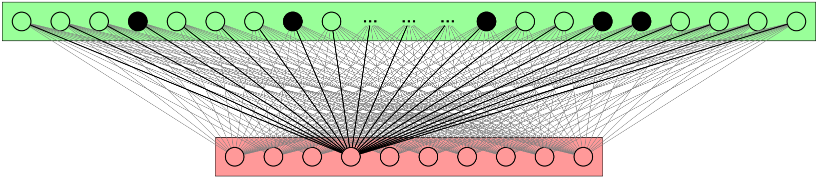
The binarized perceptron



$$n_2 = \begin{cases} +1 & \text{if } \sum_i w_{i2} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

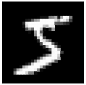
The binarized perceptron

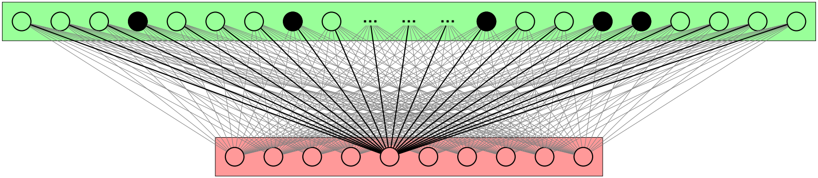




$$n_3 = \begin{cases} +1 & \text{if } \sum_i w_{i3} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

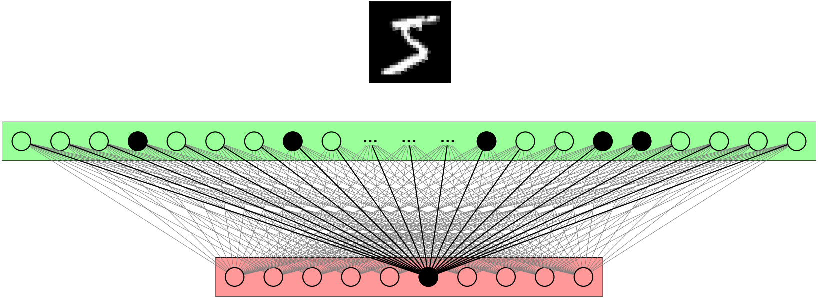
The binarized perceptron





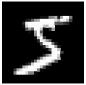
$$n_4 = \begin{cases} +1 & \text{if } \sum_i w_{i4} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

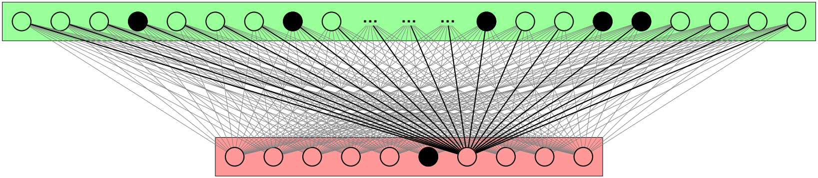
The binarized perceptron



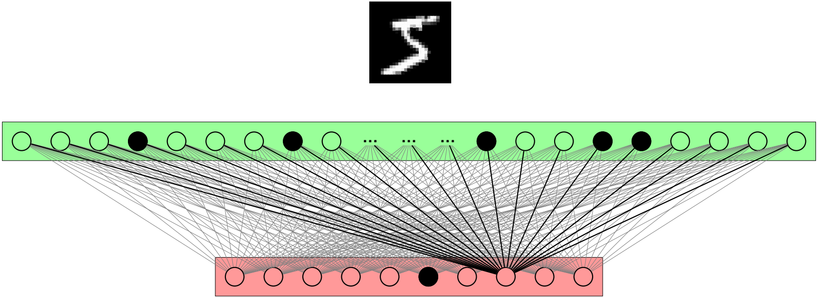
$$n_5 = \begin{cases} +1 & \text{if } \sum_i w_{i5} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

The binarized perceptron



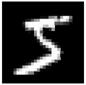


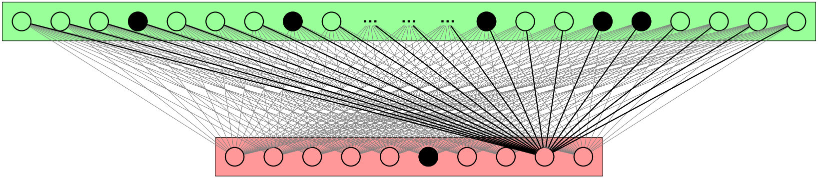
$$n_6 = \begin{cases} +1 & \text{if } \sum_i w_{i6} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$



$$n_7 = \begin{cases} +1 & \text{if } \sum_i w_{i7} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

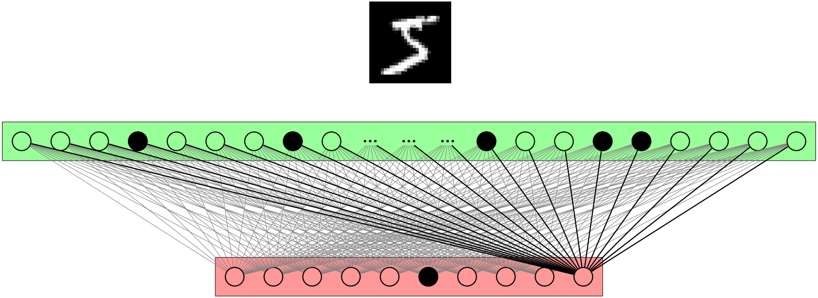
The binarized perceptron



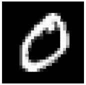


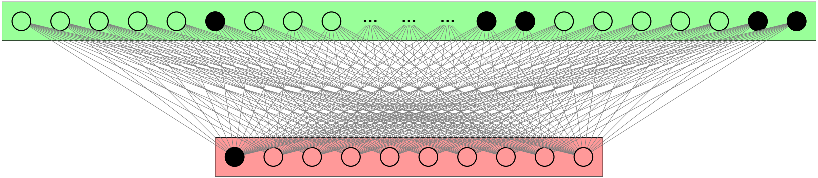
$$n_8 = \begin{cases} +1 & \text{if } \sum_i w_{i8} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

The binarized perceptron



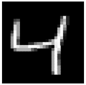
$$n_9 = \begin{cases} +1 & \text{if } \sum_i w_{i9} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

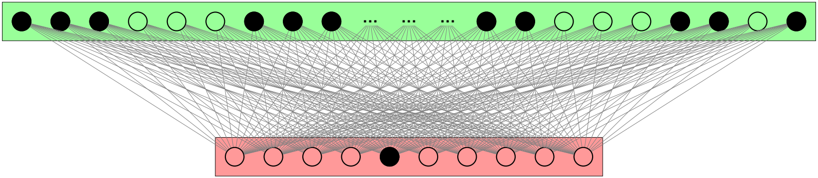




$$n_j = \begin{cases} +1 & \text{if } \sum_i w_{ij} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

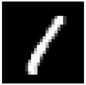
The binarized perceptron

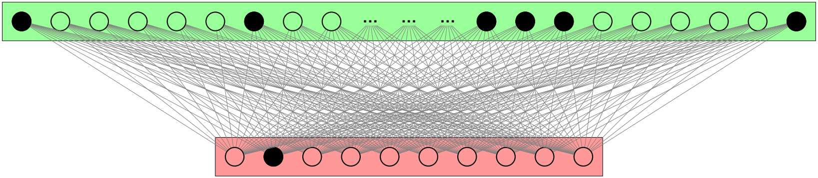




$$n_j = \begin{cases} +1 & \text{if } \sum_i w_{ij} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

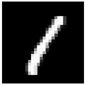
The binarized perceptron

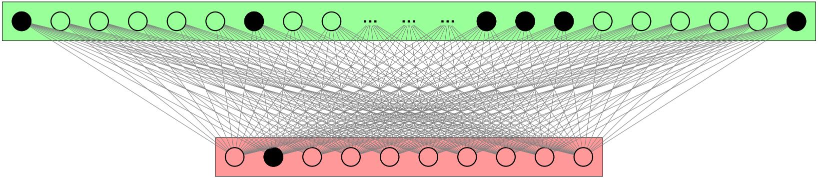




$$n_j = \begin{cases} +1 & \text{if } \sum_i w_{ij} \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}, \text{ where } w_{ij} \in \{-1, 0, 1\}.$$

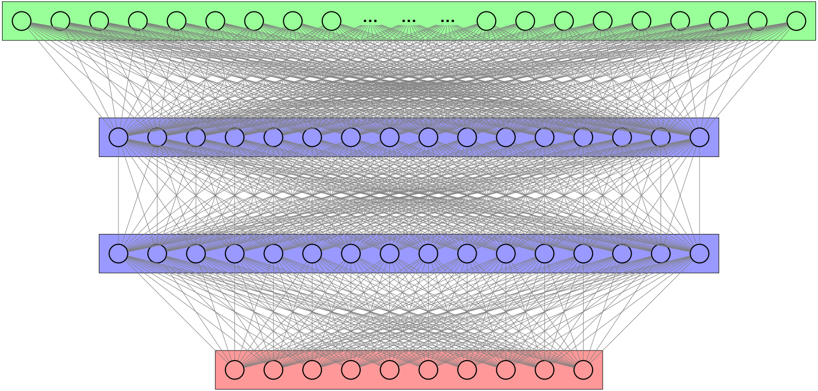
The binarized perceptron



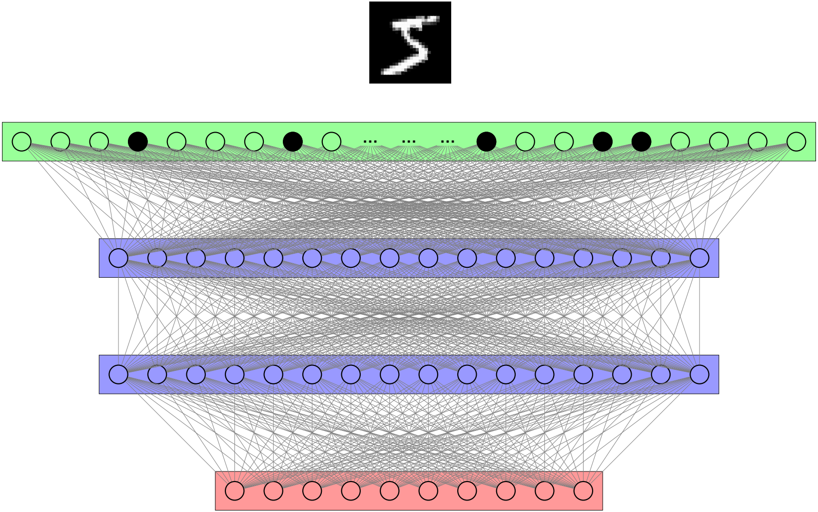


Problem: for most training sets, this problem is infeasible.

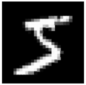
(Deep) binarized neural networks

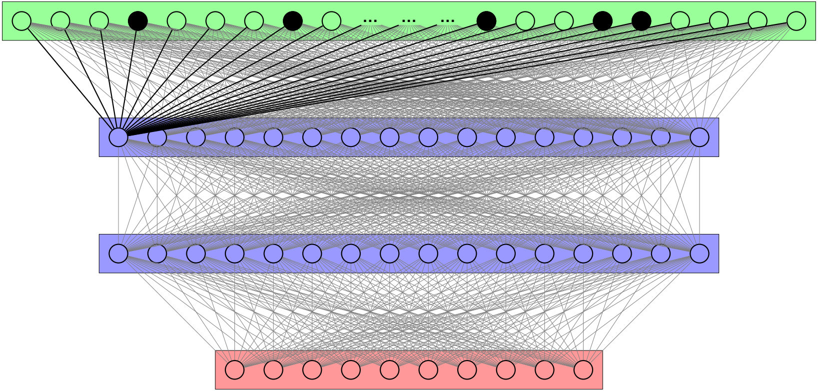


(Deep) binarized neural networks

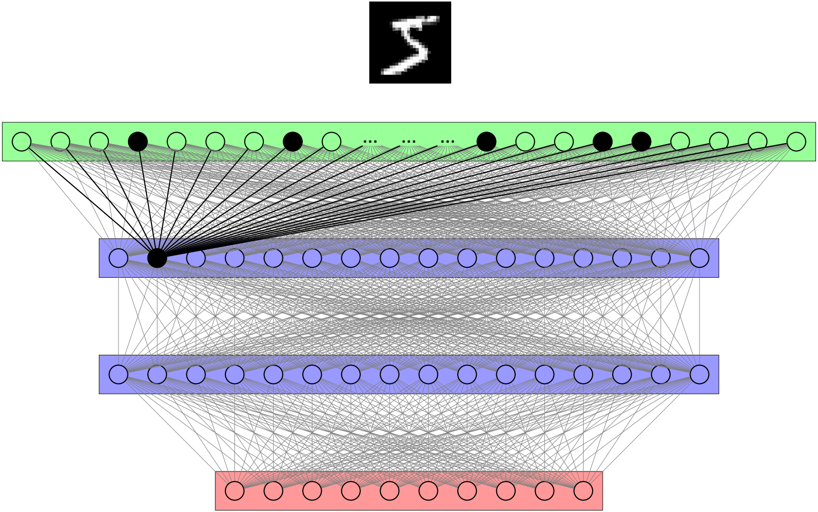


(Deep) binarized neural networks

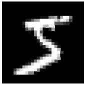


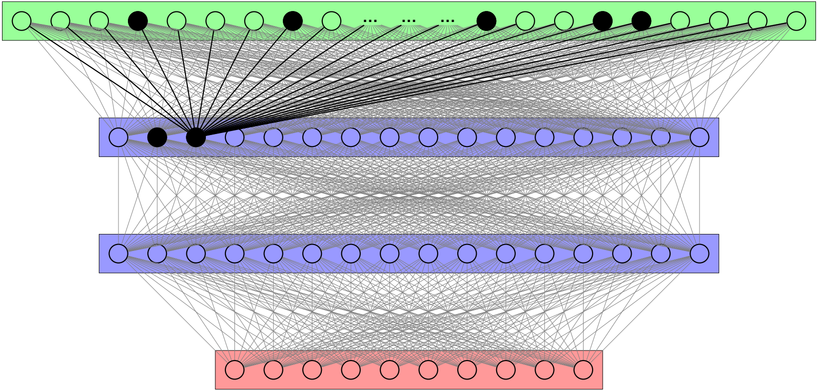


(Deep) binarized neural networks

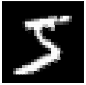


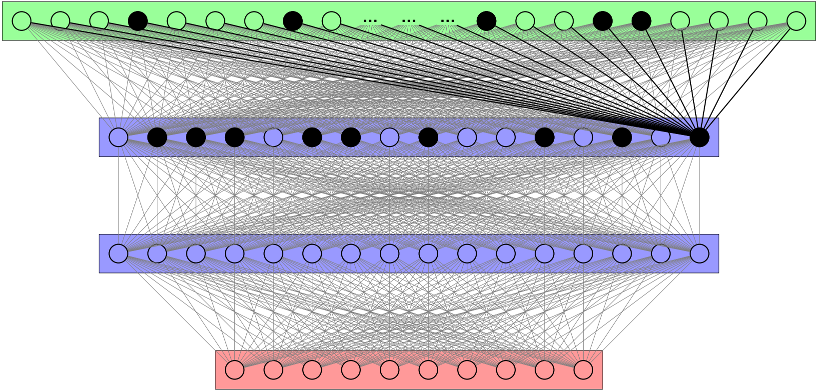
(Deep) binarized neural networks



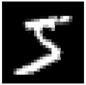


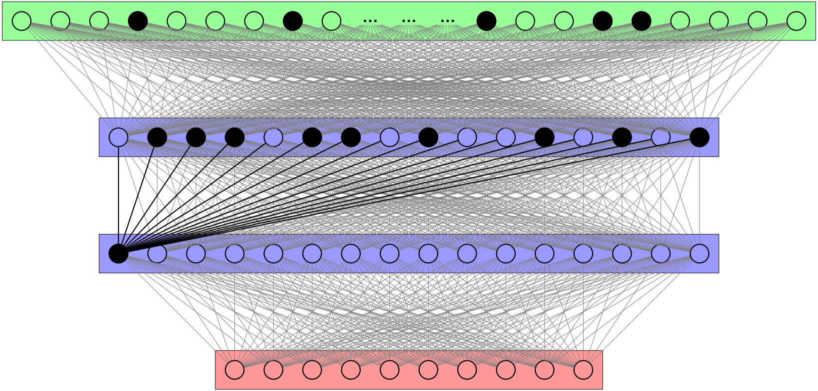
(Deep) binarized neural networks



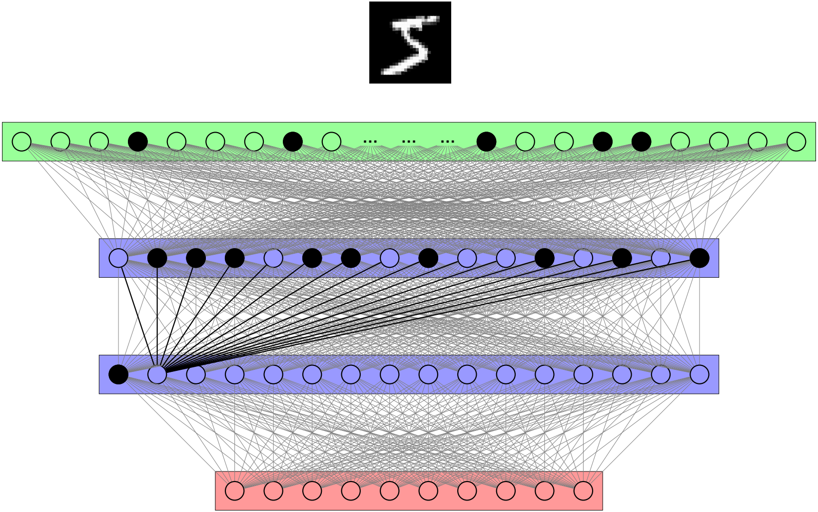


(Deep) binarized neural networks

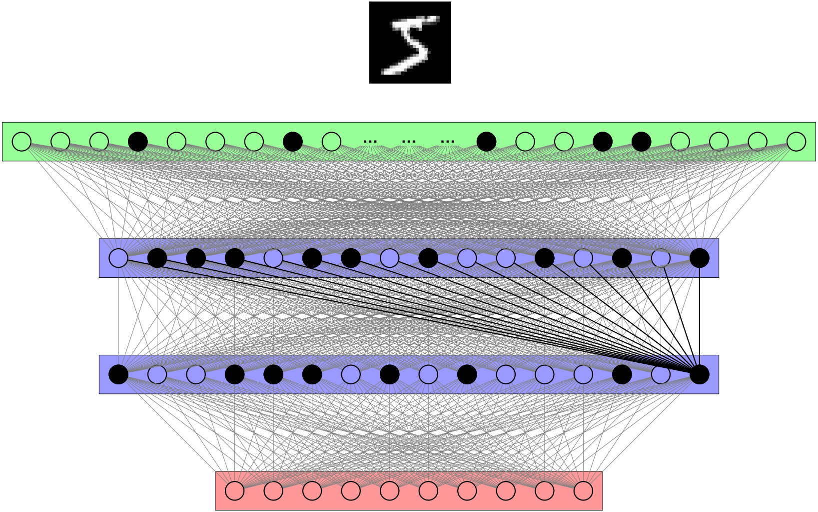




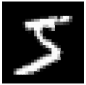
(Deep) binarized neural networks

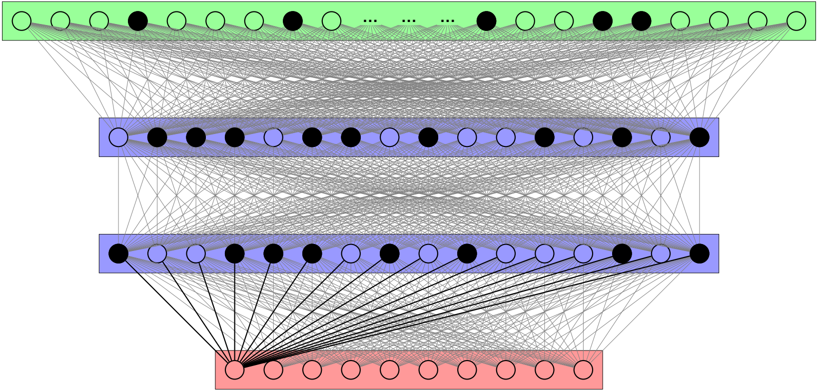


(Deep) binarized neural networks

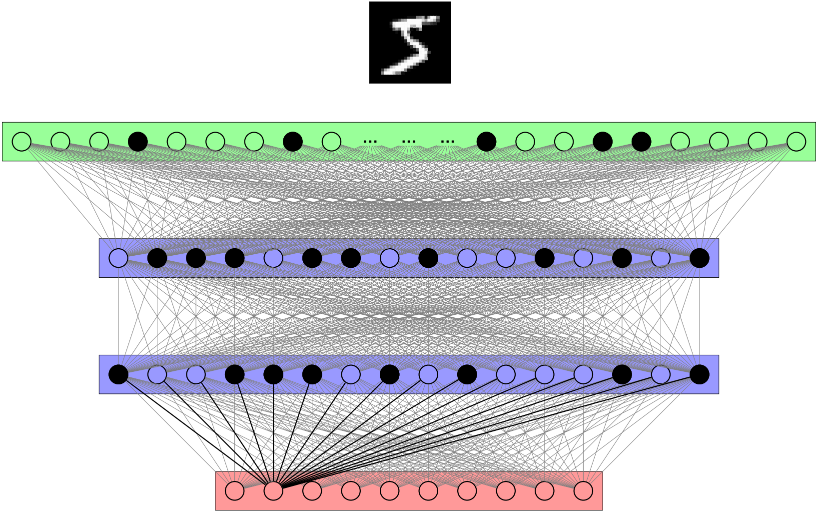


(Deep) binarized neural networks

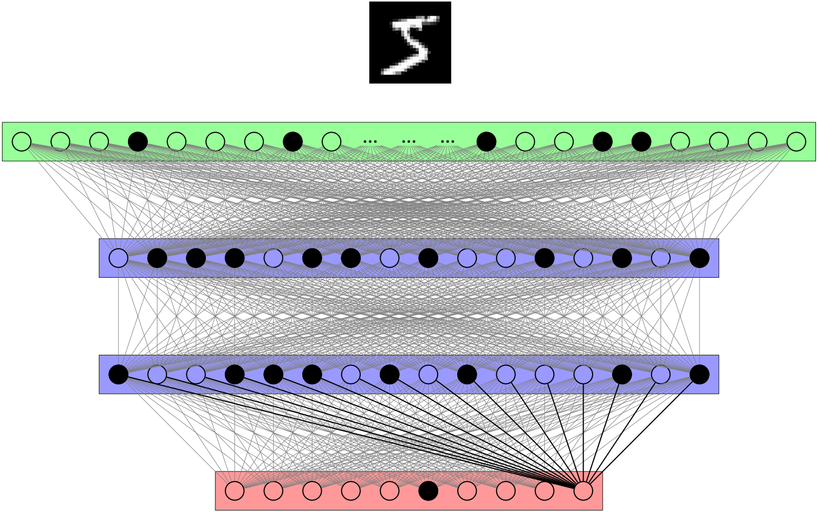




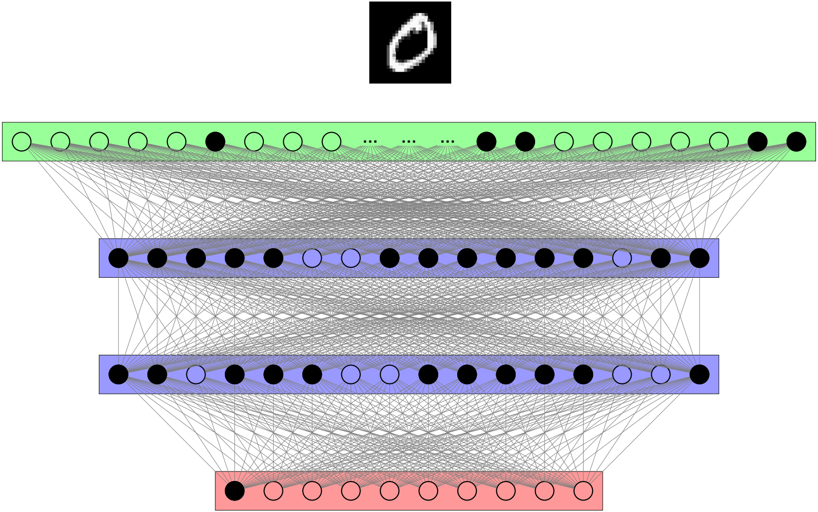
(Deep) binarized neural networks



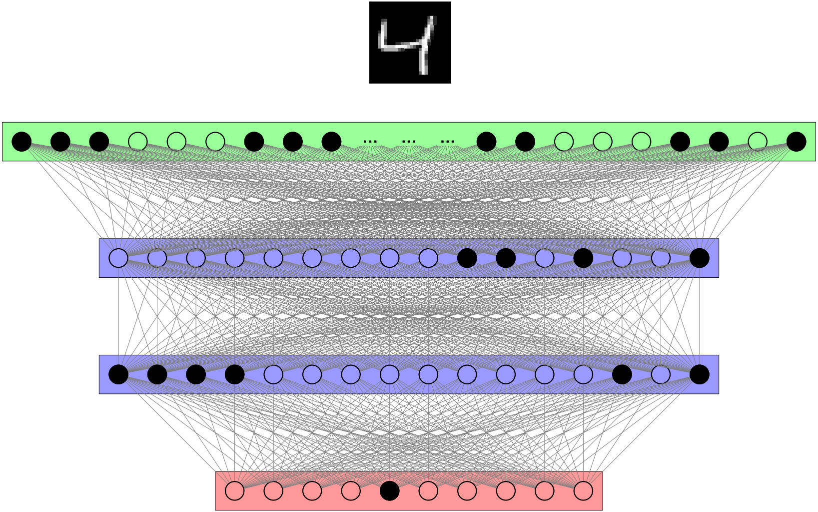
(Deep) binarized neural networks



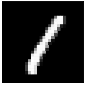
(Deep) binarized neural networks

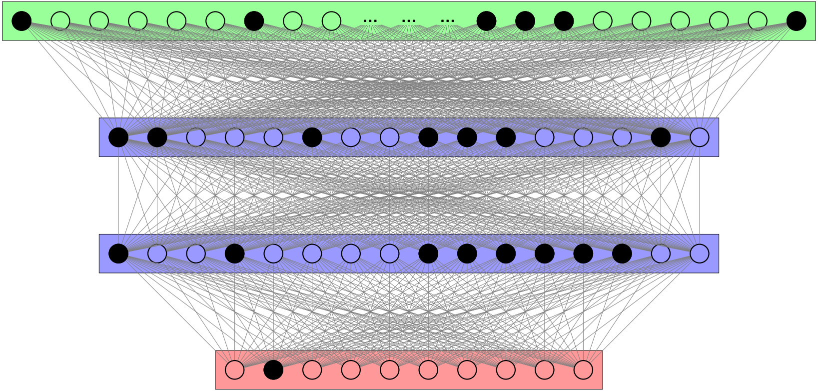


(Deep) binarized neural networks



(Deep) binarized neural networks





Formal definition

Any assignment to **W** defines a function $\mathcal{N}_{\mathbf{W}} : \mathbb{R}^{N_0} \rightarrow \{-1, 1\}^{N_L}$:

- $n_{0j} = x_j$ (first layer).
- $n_{\ell j} = 1$ if $\sum_i w_{i\ell j} \cdot n_{(\ell-1)i} \geq 0$; -1 otherwise.

UNIVERSITY OF TORONTO

Formal definition

Any assignment to **W** defines a function $\mathcal{N}_{\mathbf{W}} : \mathbb{R}^{N_0} \rightarrow \{-1, 1\}^{N_L}$:

- $n_{0j} = x_j$ (first layer).
- $n_{\ell j} = 1$ if $\sum_i w_{i\ell j} \cdot n_{(\ell-1)i} \geq 0$; -1 otherwise.

Training a BNN

Given $\mathcal{T} = \{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^T, \mathbf{y}^T)\}$, find **W** s.t. $\mathcal{N}_{\mathbf{W}}(\mathbf{x}^k) \approx \mathbf{y}^k \; \forall k$.

Formal definition

Any assignment to $\mathbf{W}$ defines a function $\mathcal{N}_{\mathbf{W}} : \mathbb{R}^{N_0} \rightarrow \{-1, 1\}^{N_L}$:

- $n_{0j} = x_j$ (first layer).
- $n_{\ell j} = 1$ if $\sum_i w_{i\ell j} \cdot n_{(\ell-1)i} \geq 0$; -1 otherwise.

Training a BNN

Given $\mathcal{T} = \{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^T, \mathbf{y}^T)\}$, find $\mathbf{W}$ s.t. $\mathcal{N}_{\mathbf{W}}(\mathbf{x}^k) \approx \mathbf{y}^k \; \forall k$.

How can we train a BNN?

Formal definition

Any assignment to $\mathbf{W}$ defines a function $\mathcal{N}_{\mathbf{W}} : \mathbb{R}^{N_0} \rightarrow \{-1, 1\}^{N_L}$:

- $n_{0j} = x_j$ (first layer).
- $n_{\ell j} = 1$ if $\sum_i w_{i\ell j} \cdot n_{(\ell-1)i} \geq 0$; -1 otherwise.

Training a BNN

Given $\mathcal{T} = \{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^T, \mathbf{y}^T)\}$, find $\mathbf{W}$ s.t. $\mathcal{N}_{\mathbf{W}}(\mathbf{x}^k) \approx \mathbf{y}^k \; \forall k$.

How can we train a BNN?

Use gradient descent!

How to train binarized neural networks

Wait! we cannot compute gradients over discrete weights.

How to train binarized neural networks

Wait! we cannot compute gradients over discrete weights.

- Train over continuous weights and activations.
- Binarize the weights and activations during the forward pass.
- Use continuous weights and activations in the backward pass.

How to train binarized neural networks

Wait! we cannot compute gradients over discrete weights.

- Train over continuous weights and activations.
- Binarize the weights and activations during the forward pass.
- Use continuous weights and activations in the backward pass.

[To me] It feels like an odd hack to GD... but it works in practice.

About this work

Main contributions

1. Show that training BNNs is a discrete optimization problem.

2. Propose a MIP, CP, and MIP/CP hybrid model to train BNNs.

3. Run an extensive experimental comparison.

Code: `https://bitbucket.org/RToroIcarte/bnn`

Related Work

SAT and MIP have been used for generating adversarial examples and verifying properties of BNNs:

- e.g. Fischetti et al. (2017), Tjeng et al. (2017), Khalil et al. (2018), Narodytska (2018), Cheng et al. (2018), among others.

Related Work

SAT and MIP have been used for generating adversarial examples and verifying properties of BNNs:

- e.g. Fischetti et al. (2017), Tjeng et al. (2017), Khalil et al. (2018), Narodytska (2018), Cheng et al. (2018), among others.

To the best of our knowledge, this is the first work that proposes model-based approaches to train BNNs.

Related Work

SAT and MIP have been used for generating adversarial examples and verifying properties of BNNs:

- e.g. Fischetti et al. (2017), Tjeng et al. (2017), Khalil et al. (2018), Narodytska (2018), Cheng et al. (2018), among others.

To the best of our knowledge, this is the first work that proposes model-based approaches to train BNNs... but why? 🤔

SAT and MIP have been used for generating adversarial examples and verifying properties of BNNs:

- e.g. Fischetti et al. (2017), Tjeng et al. (2017), Khalil et al. (2018), Narodytska (2018), Cheng et al. (2018), among others.

To the best of our knowledge, this is the first work that proposes model-based approaches to train BNNs... but why? 🤔

Model-based approaches can find provably optimal solutions

Related Work

SAT and MIP have been used for generating adversarial examples and verifying properties of BNNs:

- e.g. Fischetti et al. (2017), Tjeng et al. (2017), Khalil et al. (2018), Narodytska (2018), Cheng et al. (2018), among others.

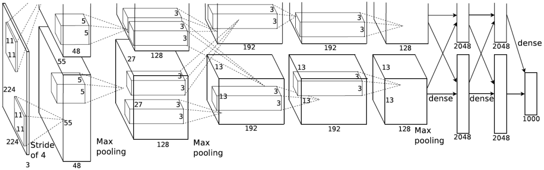
To the best of our knowledge, this is the first work that proposes model-based approaches to train BNNs... but why? 🤔

Model-based approaches can find provably optimal solutions, but they have two (fundamental) issues:

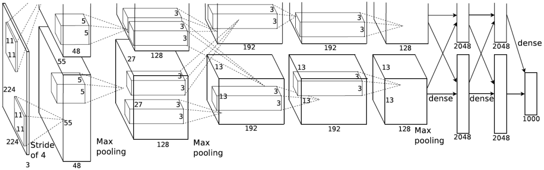
- Scalability.
- Overfitting.

Scalability

A rough estimation



AlexNet:

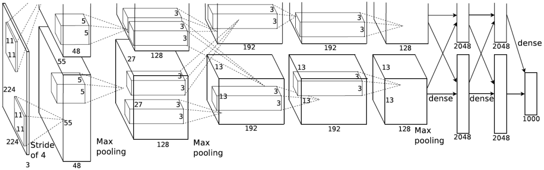


AlexNet:

- Neurons: 154K (input) – 707K (hidden) – 1K (output).
- Weights: 62.3 millions.
- ImageNet: 14 million examples.

How many discrete decision variables are needed?

A rough estimation



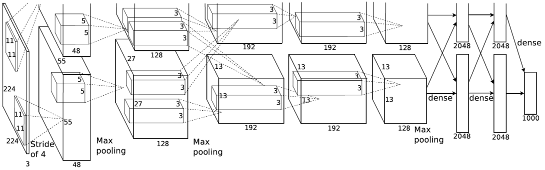
AlexNet:

- Neurons: 154K (input) – 707K (hidden) – 1K (output).
- Weights: 62.3 millions.
- ImageNet: 14 million examples.

How many discrete decision variables are needed?

62.3M (weights)

A rough estimation



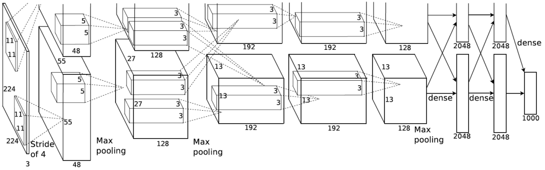
AlexNet:

- Neurons: 154K (input) – 707K (hidden) – 1K (output).
- Weights: 62.3 millions.
- ImageNet: 14 million examples.

How many discrete decision variables are needed?

62.3M (weights) $+$ 0.7M $\times$ 14M (hidden activations)

A rough estimation



AlexNet:

- Neurons: 154K (input) – 707K (hidden) – 1K (output).
- Weights: 62.3 millions.
- ImageNet: 14 million examples.

How many discrete decision variables are needed?

62.3M (weights) $+$ 0.7M $\times$ 14M (hidden activations)

$\approx 9.89 \cdot 10^{12}$ **decision variables!**

Fortunately, not all is about big data ;)

Few-shot learning

Few-shot learning

Challenge: learn a good classifier given limited training data.

Few-shot learning

Challenge: learn a good classifier given limited training data.

Why?

1. Humans learn with far less examples than deep networks.
2. Collecting large amounts of labeled data is expensive

... and sometimes impossible (e.g., healthcare).

Few-shot learning

Challenge: learn a good classifier given limited training data.

Why?

1. Humans learn with far less examples than deep networks.
2. Collecting large amounts of labeled data is expensive

... and sometimes impossible (e.g., healthcare).

E.g., let's say that we only have access to the following examples:

5 0 4 1 8 6 2 7 3 9

Few-shot learning

Challenge: learn a good classifier given limited training data.

Why?

1. Humans learn with far less examples than deep networks.
2. Collecting large amounts of labeled data is expensive

... and sometimes impossible (e.g., healthcare).

E.g., let's say that we only have access to the following examples:

5 0 4 1 8 6 2 7 3 9

We better classify them correctly!

Training BNNs: A feasibility perspective

Training BNNs: A feasibility perspective

Objective: Find a BNN that fits the data.

Problem definition

Given $\mathcal{T} = \{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^T, \mathbf{y}^T)\}$, find $\mathbf{W}$ s.t. $\mathcal{N}_{\mathbf{W}}(\mathbf{x}^k) = \mathbf{y}^k \; \forall k$.

...also known as 100% train performance.

Training BNNs: A feasibility perspective

Objective: Find a BNN that fits the data.

Problem definition

Given $\mathcal{T} = \{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^T, \mathbf{y}^T)\}$, find $\mathbf{W}$ s.t. $\mathcal{N}_{\mathbf{W}}(\mathbf{x}^k) = \mathbf{y}^k \; \forall k$.

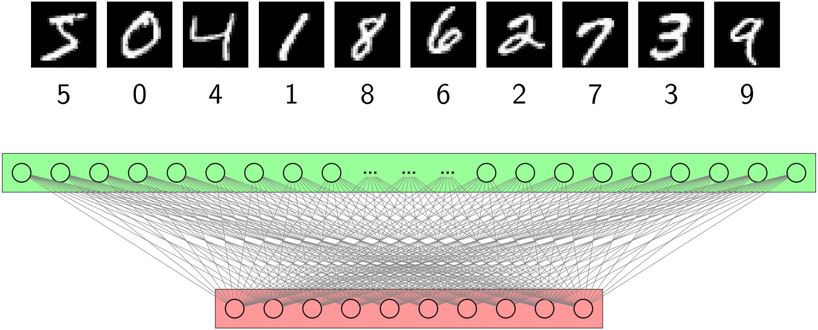
...also known as 100% train performance.

Let's start by formulating this problem as a MIP model.

MIP Model

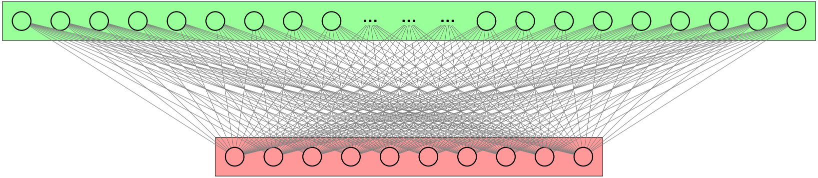
5 0 4 1 8 6 2 7 3 9

MIP Model



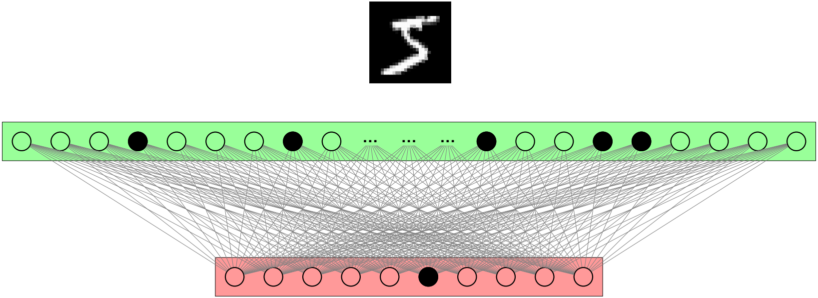
MIP Model

5 0 4 1 8 6 2 7 3 9



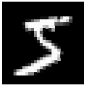
Goal: Find $\mathbf{W}$ such that $\mathcal{N}_{\mathbf{W}}(\mathbf{x}^k) = \mathbf{y}^k$ for all $k \in \{1 \dots T\}$.

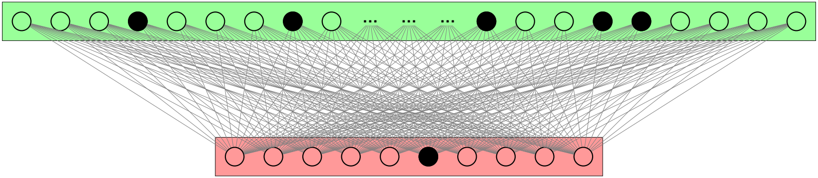
MIP Model



Goal: Find $\mathbf{W}$ such that $\mathcal{N}_{\mathbf{W}}(\mathbf{x}^k) = \mathbf{y}^k$ for all $k \in \{1 \ldots T\}$.

MIP Model

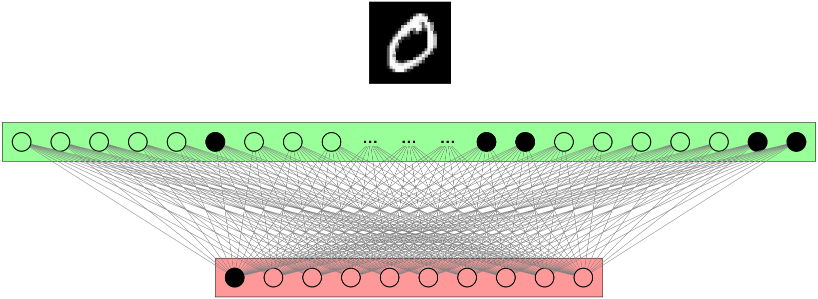




$$\sum_i w_{ij} \cdot x_i^0 \geq 0 \qquad\qquad j = 5$$

$$\sum_i w_{ij} \cdot x_i^0 < 0 \qquad\qquad j \neq 5$$

MIP Model



$$\sum_i w_{ij} \cdot x_i^1 \geq 0 \qquad\qquad j = 0$$

$$\sum_i w_{ij} \cdot x_i^1 < 0 \qquad\qquad j \neq 0$$

MIP Model

$$w_{ij} \in \{-1, 0, 1\} \qquad \forall i \in N_0, j \in N_L$$

$$\sum_{i=1}^{N_0} x_i^k \cdot w_{ij} \geq 0 \qquad \forall j \in N_L, k \in T : y_j^k = 1$$

$$\sum_{i=1}^{N_0} x_i^k \cdot w_{ij} \leq -\epsilon \qquad \forall j \in N_L, k \in T : y_j^k = -1$$

MIP Model

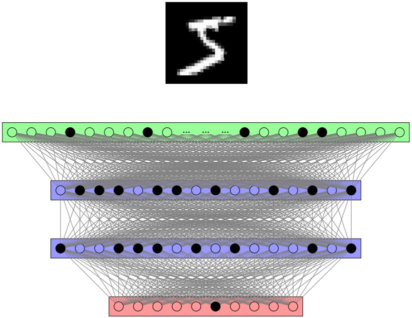
$$w_{ij} \in \{-1, 0, 1\} \qquad \forall i \in N_0, j \in N_L$$

$$\sum_{i=1}^{N_0} x_i^k \cdot w_{ij} \geq 0 \qquad \forall j \in N_L, k \in T : y_j^k = 1$$

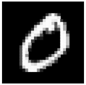
$$\sum_{i=1}^{N_0} x_i^k \cdot w_{ij} \leq -\epsilon \qquad \forall j \in N_L, k \in T : y_j^k = -1$$

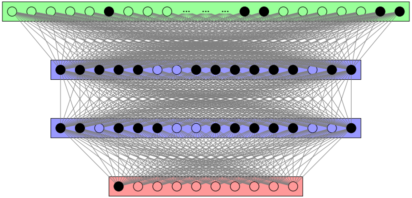
What if the BNN has hidden layers?

MIP Model



MIP Model





MIP Model

We need to model the neuron activations using extra variables:

- $u_{\ell j}^{k}$ is 1 if neuron j in layer ℓ is *active* given $x^{k} \in \mathcal{T}$ and 0 o/w.
- $2 \cdot u_{\ell j}^{k} - 1$ is the neuron's output.

MIP Model

We need to model the neuron activations using extra variables:

- $u^k_{\ell j}$ is 1 if neuron j in layer ℓ is *active* given $x^k \in \mathcal{T}$ and 0 o/w.
- $2 \cdot u^k_{\ell j} - 1$ is the neuron's output.
- $(u^k_{\ell j} = 1) \implies \sum_{i \in N_{\ell-1}} w_{i\ell j} \cdot (2 \cdot u^k_{(\ell-1)i} - 1) \geq 0$
- $(u^k_{\ell j} = 0) \implies \sum_{i \in N_{\ell-1}} w_{i\ell j} \cdot (2 \cdot u^k_{(\ell-1)i} - 1) \leq -\epsilon$

MIP Model

We need to model the neuron activations using extra variables:

- $u^k_{\ell j}$ is 1 if neuron j in layer ℓ is *active* given $x^k \in \mathcal{T}$ and 0 o/w.
- $2 \cdot u^k_{\ell j} - 1$ is the neuron's output.
- $(u^k_{\ell j} = 1) \implies \sum_{i \in N_{\ell-1}} w_{i\ell j} \cdot (2 \cdot u^k_{(\ell-1)i} - 1) \geq 0$
- $(u^k_{\ell j} = 0) \implies \sum_{i \in N_{\ell-1}} w_{i\ell j} \cdot (2 \cdot u^k_{(\ell-1)i} - 1) \leq -\epsilon$

We need to model $w \cdot n$ using extra variables:

- Add variable $c^k_{i\ell j}$ to represent $w_{i\ell j} \cdot (2 \cdot u^k_{(\ell-1)i} - 1)$

MIP Model

We need to model the neuron activations using extra variables:

- $u^k_{\ell j}$ is 1 if neuron j in layer ℓ is *active* given $x^k \in \mathcal{T}$ and 0 o/w.
- $2 \cdot u^k_{\ell j} - 1$ is the neuron's output.
- $(u^k_{\ell j} = 1) \implies \sum_{i \in N_{\ell-1}} w_{i\ell j} \cdot (2 \cdot u^k_{(\ell-1)i} - 1) \geq 0$
- $(u^k_{\ell j} = 0) \implies \sum_{i \in N_{\ell-1}} w_{i\ell j} \cdot (2 \cdot u^k_{(\ell-1)i} - 1) \leq -\epsilon$

We need to model $w \cdot n$ using extra variables:

- Add variable $c^k_{i\ell j}$ to represent $w_{i\ell j} \cdot (2 \cdot u^k_{(\ell-1)i} - 1)$
- $(u^k_{\ell j} = 1) \implies (c^k_{i\ell j} = w_{i\ell j})$
- $(u^k_{\ell j} = 0) \implies (c^k_{i\ell j} = -w_{i\ell j})$

MIP Model

$$
\begin{aligned}
&a^k_{\ell j} = \sum_{i \in N_{\ell-1}} c^k_{i\ell j} && \forall \ell \in \mathcal{L}, j \in N_\ell, k \in T \\
&a^k_{Lj} \geq 0 && \forall j \in N_L, k \in T : y^k_j = 1 \\
&a^k_{Lj} \leq -\epsilon && \forall j \in N_L, k \in T : y^t_j = -1 \\
&(u^k_{\ell j} = 1) \implies (a^k_{\ell j} \geq 0) && \forall \ell \in \mathcal{L}^{L-1}, j \in N_\ell, k \in T \\
&(u^k_{\ell j} = 0) \implies (a^k_{\ell j} \leq -\epsilon) && \forall \ell \in \mathcal{L}^{L-1}, j \in N_\ell, k \in T \\
&c^k_{i1j} = x^k_i \cdot w_{i1j} && \forall i \in N_0, j \in N_1, k \in T \\
&(u^k_{\ell j} = 1) \implies (c^k_{i\ell j} = w_{i\ell j}) && \forall \ell \in \mathcal{L}_2, i \in N_{\ell-1}, j \in N_\ell, k \in T \\
&(u^k_{\ell j} = 0) \implies (c^k_{i\ell j} = -w_{i\ell j}) && \forall \ell \in \mathcal{L}_2, i \in N_{\ell-1}, j \in N_\ell, k \in T \\
&w_{i\ell j} \in \{-1, 0, 1\} && \forall \ell \in \mathcal{L}, i \in N_{\ell-1}, j \in N_\ell \\
&u^k_{\ell j} \in \{0, 1\} && \forall \ell \in \mathcal{L}^{L-1}, j \in N_\ell, k \in T \\
&c^k_{i\ell j} \in \mathbb{R} && \forall \ell \in \mathcal{L}, i \in N_{\ell-1}, j \in N_\ell, k \in T
\end{aligned}
$$

MIP Model

"*This model is a nightmare for a MIP solver*"

—Andre A. Ciré

MIP Model

"*This model is a nightmare for a MIP solver*"
—Andre A. Ciré

1. It has (way too) many auxiliary variables:

$$u_{\ell j}^{k} \in \{0,1\} \qquad \forall \ell \in \mathcal{L}^{L-1}, j \in N_\ell, k \in T$$
$$c_{i\ell j}^{k} \in \mathbb{R} \qquad \forall \ell \in \mathcal{L}, i \in N_{\ell-1}, j \in N_\ell, k \in T$$

MIP Model

"This model is a nightmare for a MIP solver"

—Andre A. Ciré

1. It has (way too) many auxiliary variables:

$$u^k_{\ell j} \in \{0,1\} \qquad \forall \ell \in \mathcal{L}^{L-1}, j \in N_\ell, k \in T$$
$$c^k_{i\ell j} \in \mathbb{R} \qquad \forall \ell \in \mathcal{L}, i \in N_{\ell-1}, j \in N_\ell, k \in T$$

2. Everywhere I look, I see an implication constraint:

$$(u^k_{\ell j} = 1) \implies (a^k_{\ell j} \geq 0) \qquad \forall \ell \in \mathcal{L}^{L-1}, j \in N_\ell, k \in T$$
$$(u^k_{\ell j} = 0) \implies (a^k_{\ell j} \leq -\epsilon) \qquad \forall \ell \in \mathcal{L}^{L-1}, j \in N_\ell, k \in T$$
$$(u^k_{\ell j} = 1) \implies (c^k_{i\ell j} = w_{i\ell j}) \qquad \forall \ell \in \mathcal{L}_2, i \in N_{\ell-1}, j \in N_\ell, k \in T$$
$$(u^k_{\ell j} = 0) \implies (c^k_{i\ell j} = -w_{i\ell j}) \qquad \forall \ell \in \mathcal{L}_2, i \in N_{\ell-1}, j \in N_\ell, k \in T$$

CP model

CP model

We do not need auxiliary variables for this problem:

CP model

We do not need auxiliary variables for this problem:

$$
\begin{aligned}
&n_{Lj}^{k} = y_j^{k} && \forall j \in N_L, k \in T \\
&w_{i\ell j} \in \{-1, 0, 1\} && \forall \ell \in \mathcal{L}, i \in N_{\ell-1}, j \in N_\ell
\end{aligned}
$$

CP model

We do not need auxiliary variables for this problem:

$$n_{Lj}^k = y_j^k \qquad \forall j \in N_L, k \in T$$
$$w_{i\ell j} \in \{-1, 0, 1\} \qquad \forall \ell \in \mathcal{L}, i \in N_{\ell-1}, j \in N_\ell$$

where $n_{\ell j}^k$ is a CP expression recursively defined as follows:

$$n_{0j}^k = x_j^k \qquad \forall j \in N_0, k \in T$$
$$n_{\ell j}^k = 2\left(\texttt{scal\_prod}(\mathbf{w}_{\ell j}, \mathbf{n}_{\ell-1}^k) \geq 0\right) - 1 \quad \forall \ell \in \mathcal{L} \setminus \{L\}, j \in N_\ell, k \in T$$

UNIVERSITY OF TORONTO

A feasibility experiment

A feasibility experiment

Approaches:

- `GD`: Standard gradient-based approach.
- `MIP`: MIP model solved by Gurobi 8.1
- `CP`: CP model solved by CP Optimizer 12.8

A feasibility experiment

Approaches:

- `GD`: Standard gradient-based approach.
- `MIP`: MIP model solved by Gurobi 8.1
- `CP`: CP model solved by CP Optimizer 12.8

Problem instances:

- A 100 small training sets sampled from MNIST.
- Each training set has from 1 to 10 examples per class.
- Zero, one, and two hidden layers with 16 neurons.

A feasibility experiment

Approaches:

- `GD`: Standard gradient-based approach.
- `MIP`: MIP model solved by Gurobi 8.1
- `CP`: CP model solved by CP Optimizer 12.8

Problem instances:

- A 100 small training sets sampled from MNIST.
- Each training set has from 1 to 10 examples per class.
- Zero, one, and two hidden layers with 16 neurons.

Question:

- Which approach solves more instances given a 2h time limit?

A feasibility experiment

	No hidden layers			One hidden layer			Two hidden layers		
$\lvert\mathcal{T}\rvert$	MIP	GD	CP	MIP	GD	CP	MIP	GD	CP
10	**10**	**10**	**10**	**10**	9.6	**10**	9	9.2	**10**
20	**10**	**10**	**10**	7	5.6	**10**	0	8.4	**10**
30	**10**	**10**	**10**	0	0.4	**9**	0	5.2	**10**
40	**10**	**10**	**10**	0	0	**8**	0	6.2	**10**
50	**10**	**10**	**10**	0	0	**8**	0	4.2	**10**
60	**10**	**10**	**10**	0	0	**7**	0	2.2	**10**
70	**10**	**10**	**10**	0	0	**3**	0	0	**10**
80	**10**	**10**	**10**	0	0	**3**	0	0	**10**
90	**10**	**10**	8	0	0	**1**	0	0	**8**
100	**10**	**10**	8	0	0	0	0	0	**6**

Overfitting

Overfitting

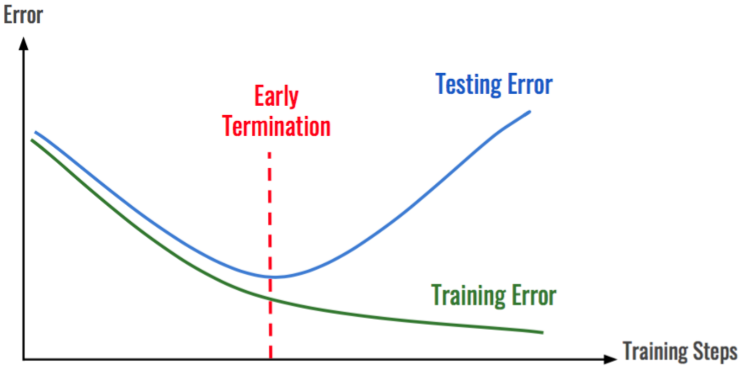
Memorizing is not learning!

The real goal is to find weights that generalize (small testing error).

Overfitting

Memorizing is not learning!

The real goal is to find weights that generalize (small testing error).



small training error $\neq$ small testing error

Towards finding solutions that generalize

5 0 4 1 8 6 2 7 3 9

We better classify them correctly!

Towards finding solutions that generalize

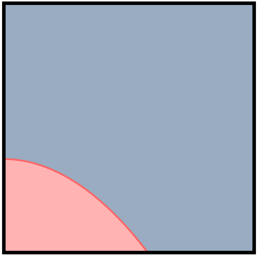
5 0 4 1 8 6 2 7 3 9

We better classify them correctly!

Towards finding solutions that generalize

5 0 4 1 8 6 2 7 3 9

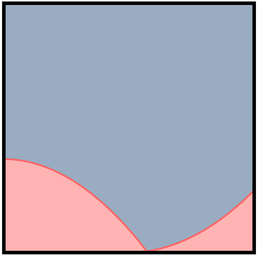
We better classify them correctly!



Towards finding solutions that generalize

5 0 4 1 8 6 2 7 3 9

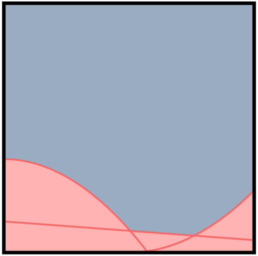
We better classify them correctly!



Towards finding solutions that generalize

5 0 4 1 8 6 2 7 3 9

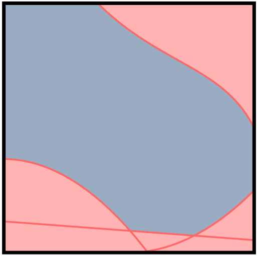
We better classify them correctly!



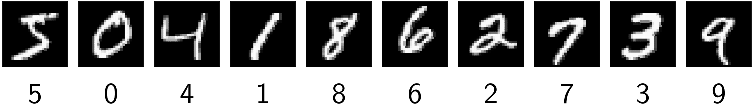
Towards finding solutions that generalize

5 0 4 1 8 6 2 7 3 9

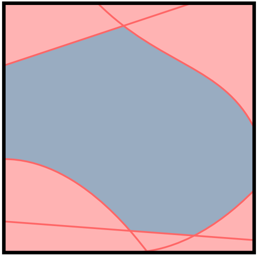
We better classify them correctly!



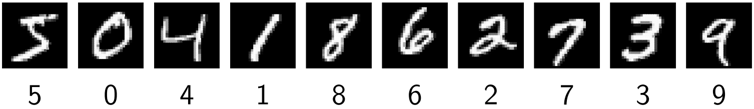
Towards finding solutions that generalize



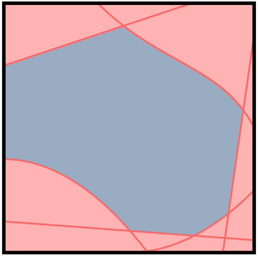
We better classify them correctly!



Towards finding solutions that generalize



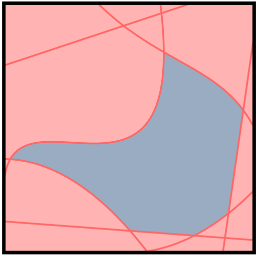
We better classify them correctly!



Towards finding solutions that generalize

5 0 4 1 8 6 2 7 3 9

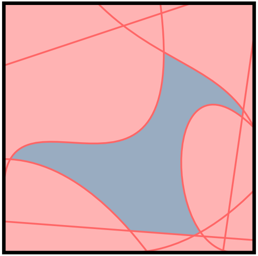
We better classify them correctly!



Towards finding solutions that generalize

5	0	4	1	8	6	2	7	3	9

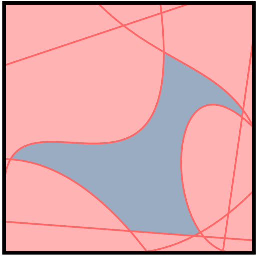
We better classify them correctly!



Towards finding solutions that generalize

5 0 4 1 8 6 2 7 3 9

We better classify them correctly!



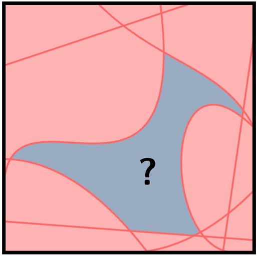
While most solutions overfit

... some generalize.

Towards finding solutions that generalize

5 0 4 1 8 6 2 7 3 9

We better classify them correctly!



While most solutions overfit

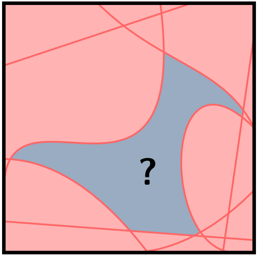
... some generalize.

How can we identify them?

Towards finding solutions that generalize

5 0 4 1 8 6 2 7 3 9

We better classify them correctly!



While most solutions overfit

... some generalize.

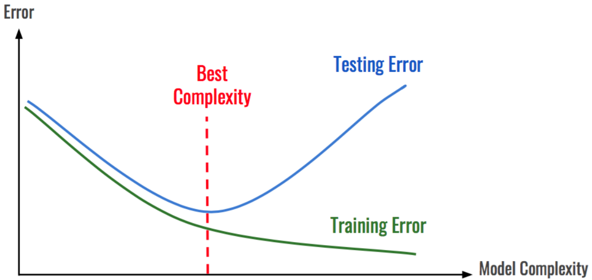
How can we identify them?

Two principles: simplicity & robustness.

The simplicity principle

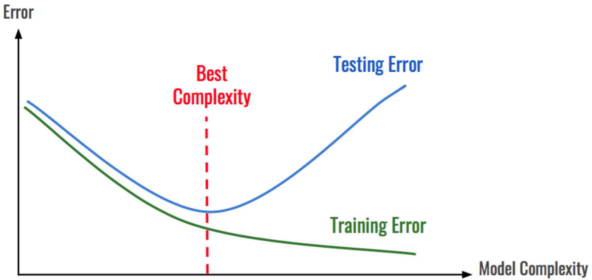
The simplicity principle

Occam's razor: prefer the simplest BNN that fits the data.



The simplicity principle

Occam's razor: prefer the simplest BNN that fits the data.



$$\min_{\mathbf{W}} \left\{ \sum_{w \in \mathbf{W}} |w| : \mathcal{N}_{\mathbf{W}}(\mathbf{x}) = \mathbf{y}, \forall (\mathbf{x}, \mathbf{y}) \in \mathcal{T},\ w \in \{-1, 0, 1\}, \forall w \in \mathbf{W} \right\}.$$

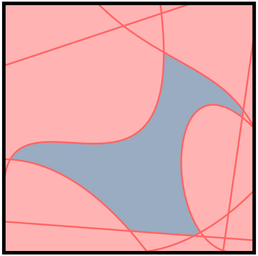
(min-weight)

The robustness principle

The robustness principle

Prefer robust solutions

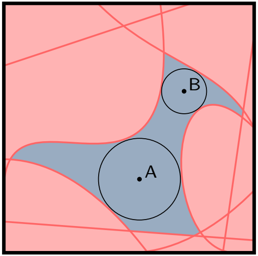
BNNs that fit the data under small perturbations to their weights.



The robustness principle

Prefer robust solutions

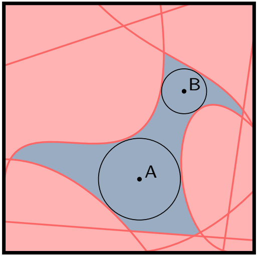
BNNs that fit the data under small perturbations to their weights.



The robustness principle

Prefer robust solutions

BNNs that fit the data under small perturbations to their weights.



$$\max_{\mathbf{W}} \quad \sum_{\ell \in \mathcal{L}, j \in N_\ell} \min\{|a_{\ell j}(\mathbf{x})| : (\mathbf{x}, \mathbf{y}) \in \mathcal{T}\}$$

$$\text{s.t.} \quad \mathcal{N}_{\mathbf{W}}(\mathbf{x}) = \mathbf{y} \qquad \forall (\mathbf{x}, \mathbf{y}) \in \mathcal{T}$$

$$w \in \{-1, 0, 1\} \qquad \forall w \in \mathbf{W}$$

(max-margin)

An optimality experiment

An optimality experiment

Approaches:

- $\texttt{CP}_w$ and $\texttt{CP}_m$: min-weight and max-margin CP models.
- $\texttt{MIP}_w$ and $\texttt{MIP}_m$: min-weight and max-margin MIP models.

An optimality experiment

Approaches:

- $\texttt{CP}_w$ and $\texttt{CP}_m$: min-weight and max-margin CP models.
- $\texttt{MIP}_w$ and $\texttt{MIP}_m$: min-weight and max-margin MIP models.

Problem instances:

- Same 100 instances using 0, 1, or 2 hidden layers.

An optimality experiment

Approaches:

- $\texttt{CP}_w$ and $\texttt{CP}_m$: min-weight and max-margin CP models.
- $\texttt{MIP}_w$ and $\texttt{MIP}_m$: min-weight and max-margin MIP models.

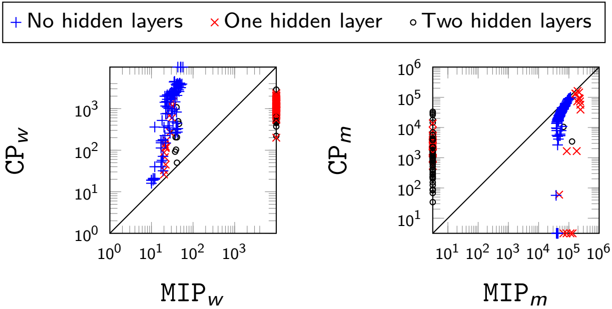
Problem instances:

- Same 100 instances using 0, 1, or 2 hidden layers.

Question:

- Will MIP or CP find better solutions given a 2h time limit?

An optimality experiment



(a) Min-weight optimization (b) Max-margin optimization

CP/MIP hybrid methods

Idea: use CP to find feasible solutions and MIP to optimize them.

CP/MIP hybrid methods

Idea: use CP to find feasible solutions and MIP to optimize them.

Option 1: model `HW`

Use the CP solution as a warm-start for MIP.

CP/MIP hybrid methods

Idea: use CP to find feasible solutions and MIP to optimize them.

Option 1: model `HW`

Use the CP solution as a warm-start for MIP.

Option 2: model `HA`

Use the CP solution to fix the activations of all neurons in the MIP model and search only over the weights.

CP/MIP hybrid methods

"This model is a nightmare for a MIP solver"

—Andre A. Ciré

1. It has (way too) many auxiliary variables:

$$u^k_{\ell j} \in \{0,1\} \qquad \forall \ell \in \mathcal{L}^{L-1}, j \in N_\ell, k \in T$$

$$c^k_{i\ell j} \in \mathbb{R} \qquad \forall \ell \in \mathcal{L}, i \in N_{\ell-1}, j \in N_\ell, k \in T$$

2. Everywhere I look, I see an implication constraint:

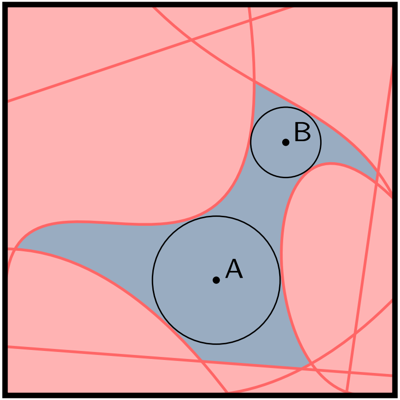
$$(u^k_{\ell j} = 1) \implies (a^k_{\ell j} \geq 0) \qquad \forall \ell \in \mathcal{L}^{L-1}, j \in N_\ell, k \in T$$

$$(u^k_{\ell j} = 0) \implies (a^k_{\ell j} \leq -\epsilon) \qquad \forall \ell \in \mathcal{L}^{L-1}, j \in N_\ell, k \in T$$

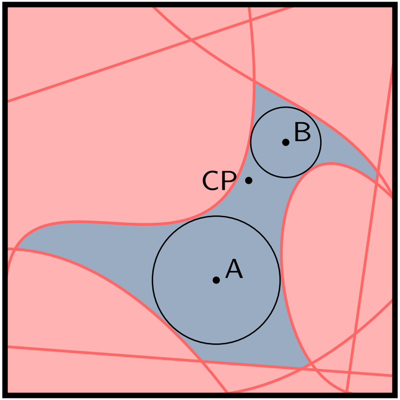
$$(u^k_{\ell j} = 1) \implies (c^k_{i\ell j} = w_{i\ell j}) \qquad \forall \ell \in \mathcal{L}_2, i \in N_{\ell-1}, j \in N_\ell, k \in T$$

$$(u^k_{\ell j} = 0) \implies (c^k_{i\ell j} = -w_{i\ell j}) \qquad \forall \ell \in \mathcal{L}_2, i \in N_{\ell-1}, j \in N_\ell, k \in T$$

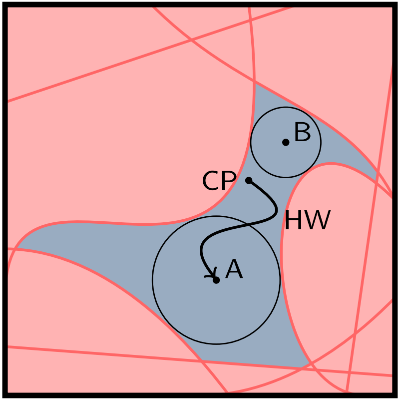
CP/MIP hybrid methods



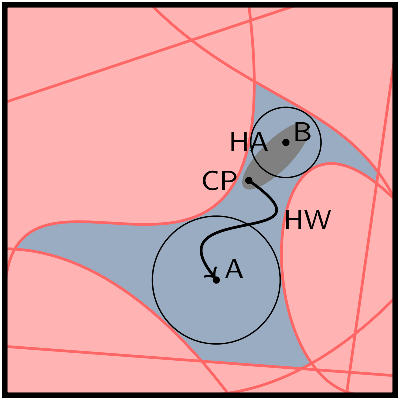
CP/MIP hybrid methods



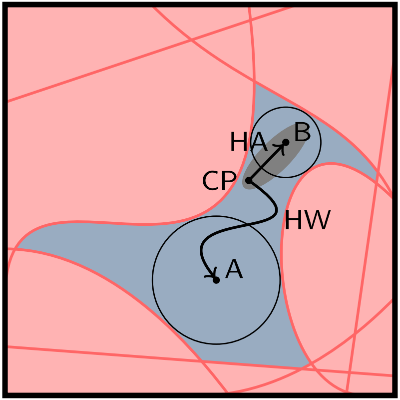
CP/MIP hybrid methods



CP/MIP hybrid methods



CP/MIP hybrid methods



Results

A generalization experiment

Approaches:

- $\texttt{CP}_w$, $\texttt{CP}_m$, $\texttt{MIP}_w$, and $\texttt{MIP}_m$ as before.
- $\texttt{HW}_w$ and $\texttt{HW}_m$: min-weight and max-margin warm-start CP/MIP.
- $\texttt{HA}_w$ and $\texttt{HA}_m$: min-weight and max-margin fixed-activation CP/MIP.
- $\texttt{GD}_b$ and $\texttt{GD}_t$: Two versions of gradient descent.

A generalization experiment

Approaches:

- $\mathtt{CP}_w$, $\mathtt{CP}_m$, $\mathtt{MIP}_w$, and $\mathtt{MIP}_m$ as before.
- $\mathtt{HW}_w$ and $\mathtt{HW}_m$: min-weight and max-margin warm-start CP/MIP.
- $\mathtt{HA}_w$ and $\mathtt{HA}_m$: min-weight and max-margin fixed-activation CP/MIP.
- $\mathtt{GD}_b$ and $\mathtt{GD}_t$: Two versions of gradient descent.

Problem instances:

- Same 100 instances using 0, 1, or 2 hidden layers.

A generalization experiment

Approaches:

- $\texttt{CP}_w$, $\texttt{CP}_m$, $\texttt{MIP}_w$, and $\texttt{MIP}_m$ as before.
- $\texttt{HW}_w$ and $\texttt{HW}_m$: min-weight and max-margin warm-start CP/MIP.
- $\texttt{HA}_w$ and $\texttt{HA}_m$: min-weight and max-margin fixed-activation CP/MIP.
- $\texttt{GD}_b$ and $\texttt{GD}_t$: Two versions of gradient descent.

Problem instances:

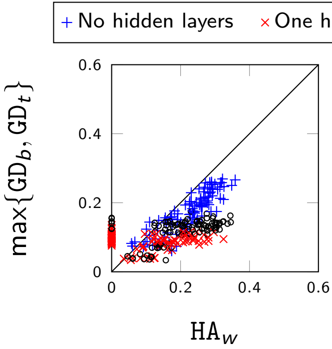
- Same 100 instances using 0, 1, or 2 hidden layers.

Question:

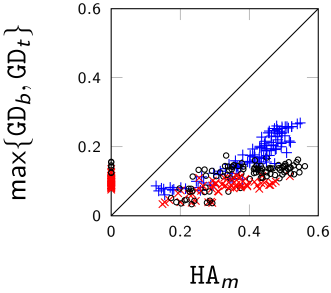
- Which model finds solutions that generalize better within 2h?

A generalization experiment

Test performance



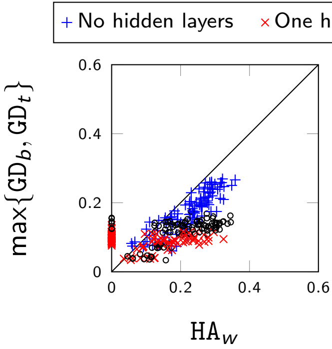
(c) Min-weight optimization



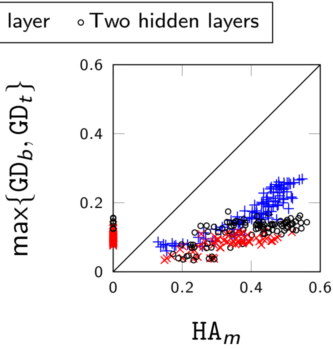
(d) Max-margin optimization

A generalization experiment

Test performance



(e) Min-weight optimization



(f) Max-margin optimization

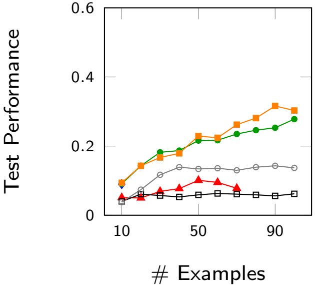
$\mathtt{HA}_m$ outperforms $\max\{\mathtt{GD}_b, \mathtt{GD}_t\}$ in **253 out of 300** experiments!

A generalization experiment

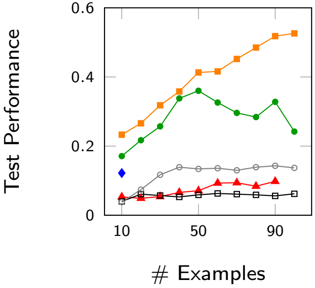
$\mathtt{HA}_m$ outperforms alternatives by a **large margin**!

A generalization experiment

$\mathtt{HA}_m$ outperforms alternatives by a **large margin**!



(i) 2 HL, min-weight



(j) 2 HL, max-margin

Concluding remarks

Concluding remarks

Summary:

- Training BNNs is a discrete optimization problem.
- We can train BNNs using MIP and CP, but:
  - Use small datasets.
  - Optimize some proxy for generalizability.
- Our $\mathtt{HA}_m$ model either outperformed GD or timed out.

Concluding remarks

Summary:

- Training BNNs is a discrete optimization problem.
- We can train BNNs using MIP and CP, but:
  - Use small datasets.
  - Optimize some proxy for generalizability.
- Our $\mathtt{HA}_m$ model either outperformed GD or timed out.

Open questions:

- How far can model-based approaches scale?
- What other proxies for generalization are worth studying?
- Are there meaningful ways to combine GD with MIP and CP?

Concluding remarks

Summary:

- Training BNNs is a discrete optimization problem.
- We can train BNNs using MIP and CP, but:
  - Use small datasets.
  - Optimize some proxy for generalizability.
- Our $\texttt{HA}_m$ model either outperformed GD or timed out.

Open questions:

- How far can model-based approaches scale?
- What other proxies for generalization are worth studying?
- Are there meaningful ways to combine GD with MIP and CP?

Thanks!