Formal Analysis of Binarized Deep Neural Networks

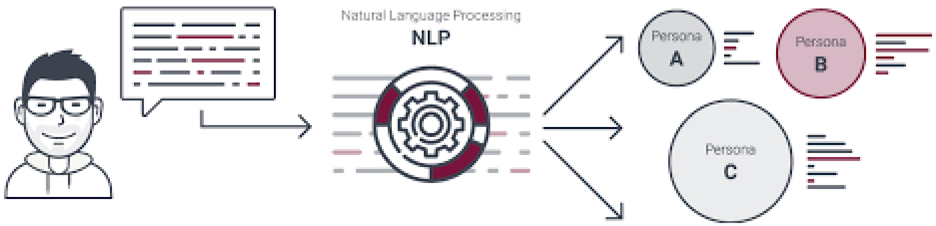
Nina Narodytska

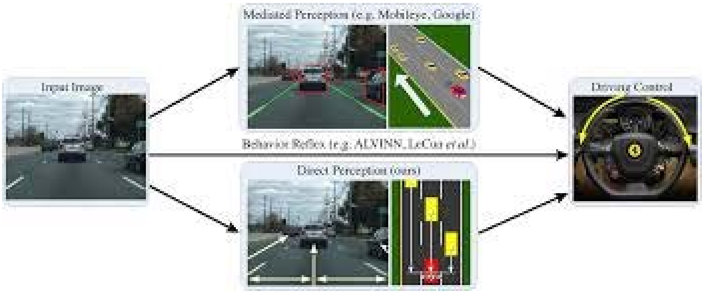
Outline

1. Motivation
2. Adversarial attacks on Neural Networks
3. Verification of Neural Networks
4. Few observations on properties/networks

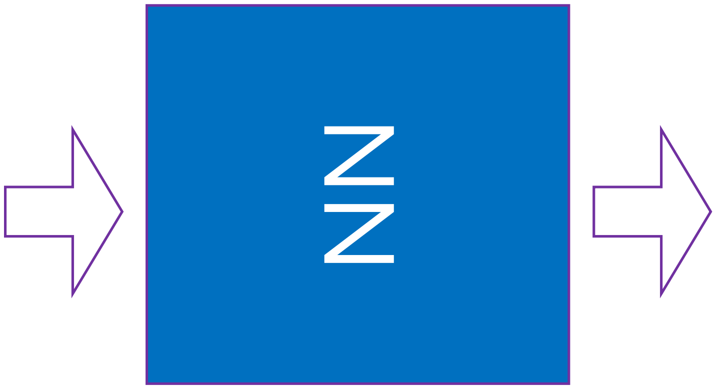
Motivation

Machine Learning





Vulnerability of NN



Function

Vulnerability of NN

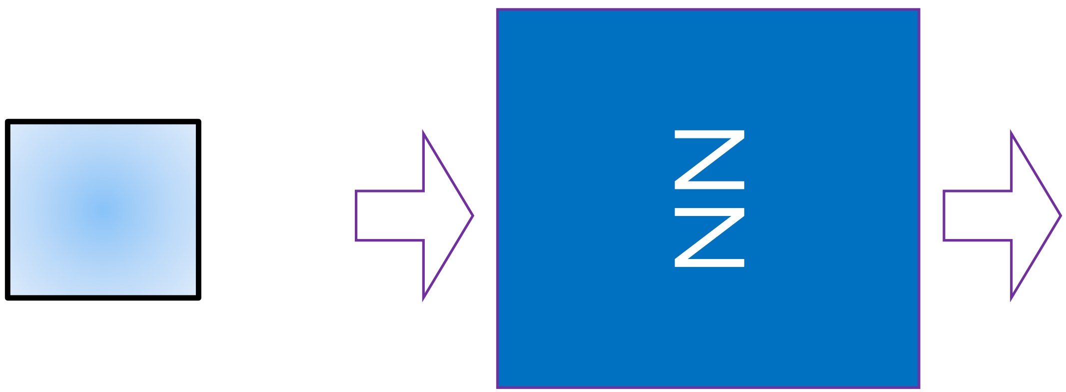
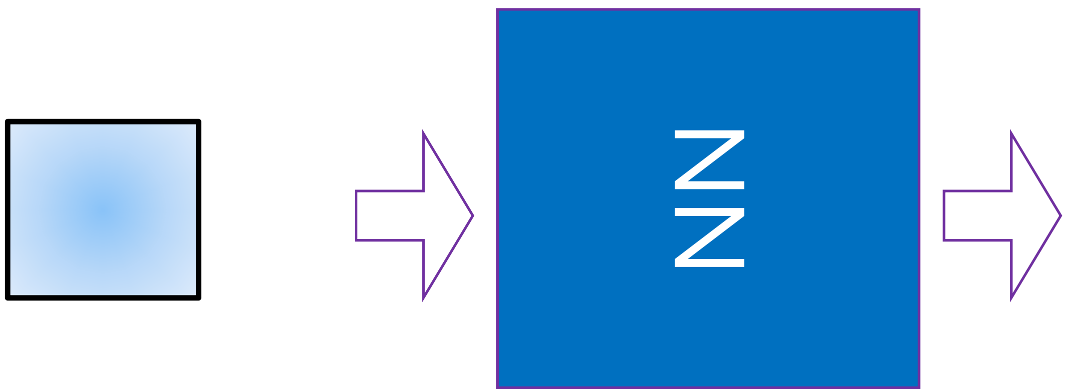


Image	Function

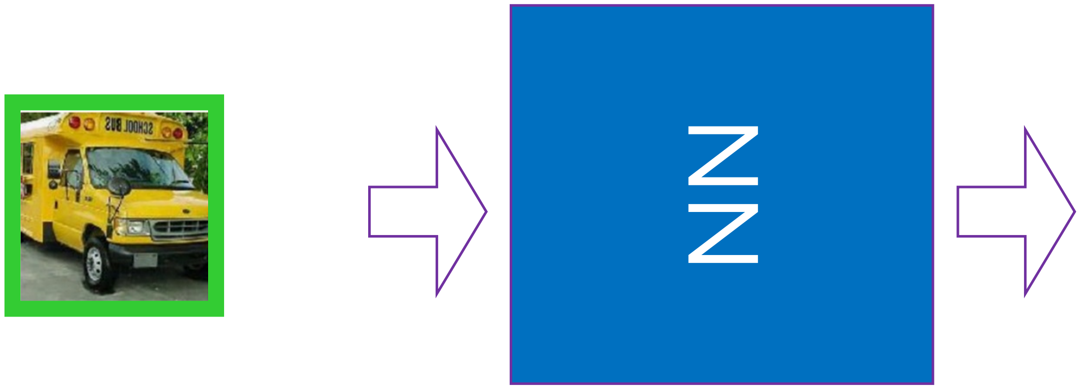
Vulnerability of NN



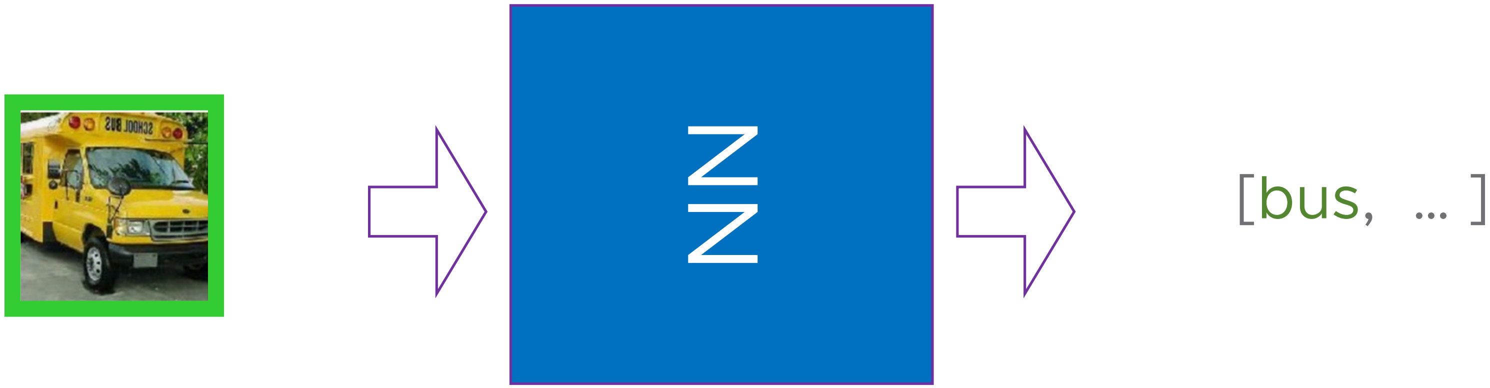
[
Class C (0.6),
Class B (0.2),
Class A (0.2)
]

Image	Function	Output

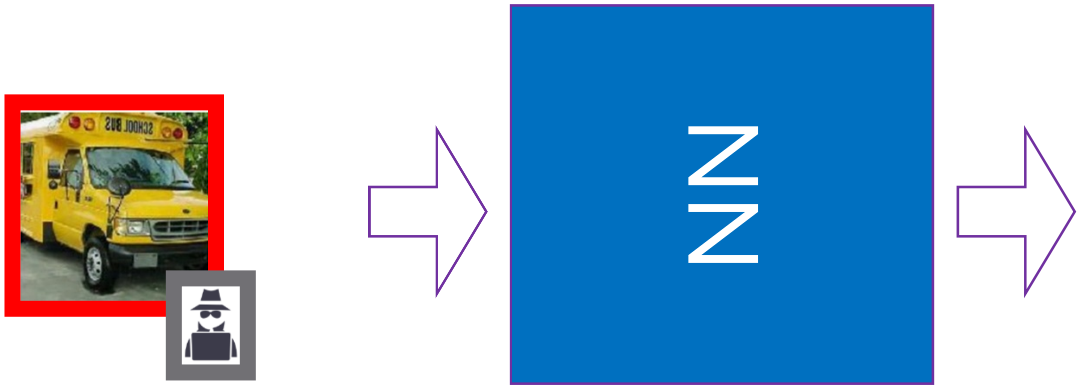
Vulnerability of NN



Vulnerability of NN



Vulnerability of NN



Vulnerability of NN

Adversarial attacks

[Szegedy *et al.*] *Intriguing properties of neural networks*

Untargeted adversarial examples

Given an input (X, C), an input $X' = X + P$ is an untargeted adversarial example iff NN misclassifies X' and P is small according to some metric.

Untargeted adversarial examples

Given an input $(\mathbf{X}, \mathbf{C})$, an input $X' = X + P$ is an untargeted adversarial example iff NN misclassifies X' and P is small according to some metric.

Untargeted adversarial examples

Given an input $(\mathbf{X},\ \mathbf{C})$, an input $\mathbf{X'}= \mathbf{X} + \mathbf{P}$ is an untargeted adversarial example iff NN misclassifies X' and P is small according to some metric.

Untargeted adversarial examples

Given an input $(\mathbf{X},\ \mathbf{C})$, an input $\mathbf{X'} = \mathbf{X} + \mathbf{P}$ is an untargeted adversarial example iff *NN misclassifies* X' and P is small according to some metric.

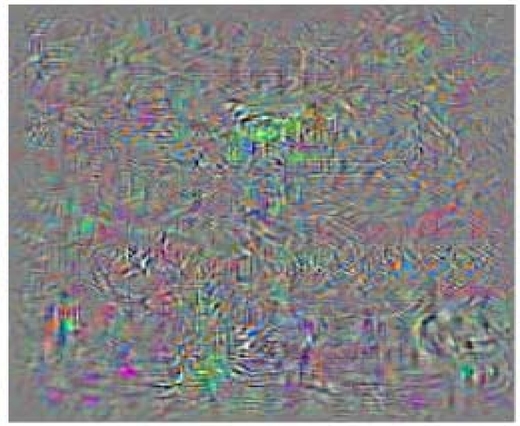
Untargeted adversarial examples

Original image

1.Bus
2. …

Untargeted adversarial examples

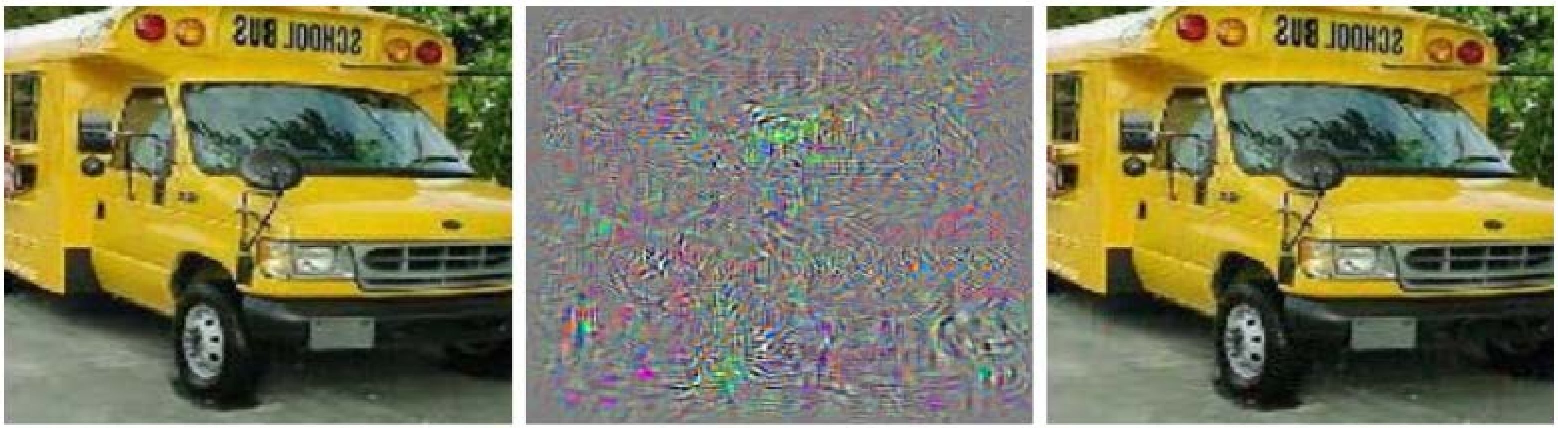
Original image + **P**erturbation



1.Bus
2. …

Untargeted adversarial examples

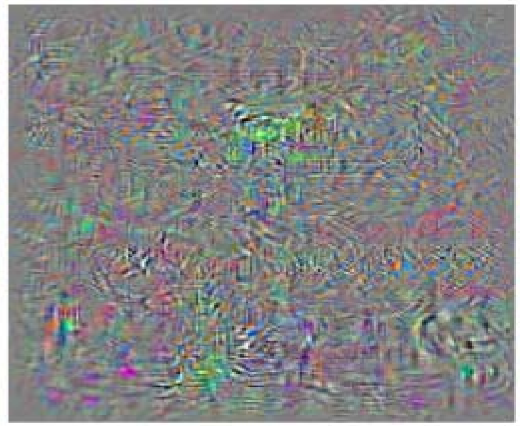
Original image + Perturbation = Perturbed image



1.Bus
2. …

Untargeted adversarial examples

Original image + **P**erturbation = Perturbed image


1.Bus
2. …

1. Ostrich
2. Bus

Targeted adversarial examples

Given a input (X, C) and a target class T, an input $X' = X + P$ is an targeted adversarial example iff the top prediction is T and P is small according to some metric.

Targeted adversarial examples

Given a input $(\mathbf{X}, \mathbf{C})$ and a target class $\mathbf{T}$, an input $X' = X + P$ is an targeted adversarial example iff the top prediction is T and P is small according to some metric.

Targeted adversarial examples

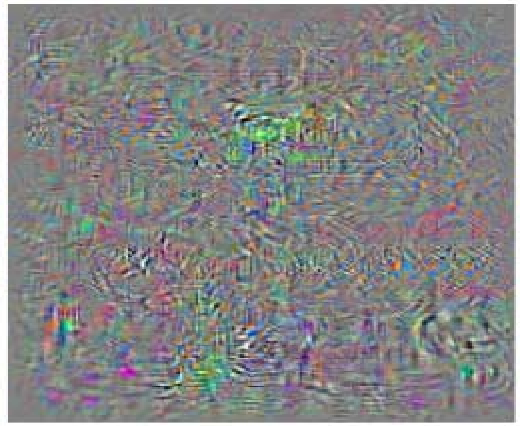
Given a input **(X, C)** and a target class **T**, an input **X'=X + P** is an targeted adversarial example iff the top prediction is T and P is small according to some metric.

Targeted adversarial examples

Given a input **(X, C)** and a target class **T**, an input **X'=X + P** is an targeted adversarial example iff *the top prediction is T* and P is small according to some metric.

Targeted adversarial examples

Original image + Perturbation = Perturbed image

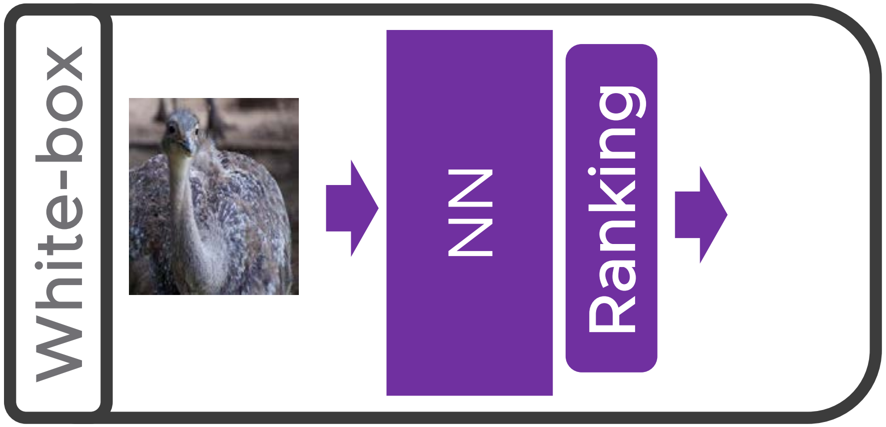


1.Bus
2. …

1. Building
2. Bus

Target: Building

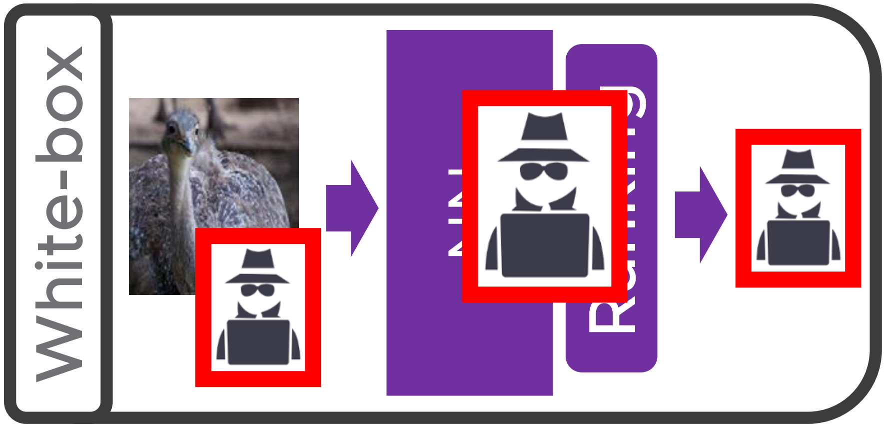
White-box vs Black-box Attacks



[Goodfellow *et al.*, Szegedy *et al.*]

[Papernot *et al.*, 2016a, 2016b]

White-box vs Black-box Attacks

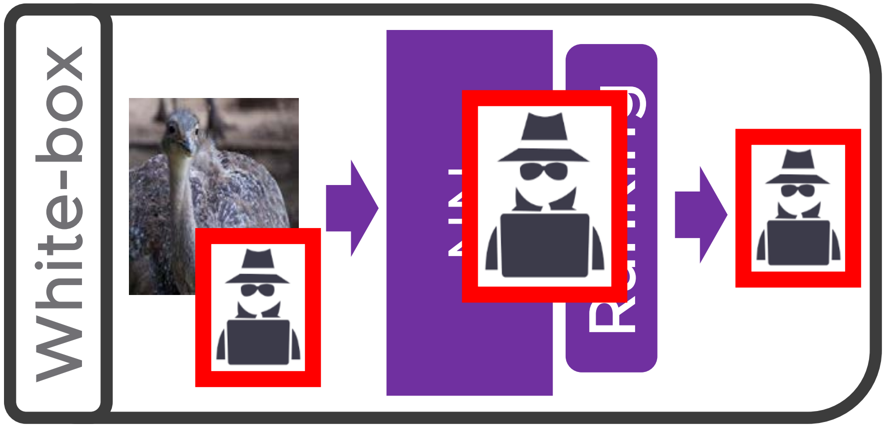


[Goodfellow *et al.*, Szegedy *et al.*]

[Papernot *et al.*, 2016a, 2016b]

Gradient-based methods that generate adversarial images by perturbing the gradients of the loss function w.r.t. the input image

White-box vs Black-box Attacks



[Goodfellow *et al.*, Szegedy *et al.*]

[Papernot *et al.*, 2016a, 2016b]

Gradient-based methods that generate adversarial images by perturbing the gradients of the loss function w.r.t. the input image

- More realistic and applicable model
- Challenging because of weak adversaries: no knowledge of the network architecture
- Previous attacks require 'transferability' assumption on adversarial examples
- GAN based attacks

Are NNs reliable to use in safety-critical application?

Verification of NN

Verification of NN

Verification of Neural Networks

Verification of NN

- Pulina and Tacchella 2010.
  An Abstraction-Refinement Approach to Verification of Artificial Neural Networks.
- Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, D. Vytiniotis, Aditya Nori, and A. Criminisi.
  Measuring neural net robustness with constraints
- Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer.
  Reluplex: An efficient SMT solver for verifying deep neural networks.
- Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu.
  Safety verification of deep neural networks
- Svyatoslav Korneev, Nina Narodytska, Luca Pulina, Armando Tacchella, N. Bjorner, and M. Sagiv. ***Constrained image generation using binarized neural networks with decision procedures***.
- Nina Narodytska, Shiva Prasad Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh. ***Verifying properties of binarized deep neural networks***
- Chih-Hong Cheng, Georg Nuhrenberg, and Harald Ruess.
  Maximum resilience of artificial neural networks.
- Chih-Hong Cheng, Georg Nuhrenberg, and Harald Ruess.
  Verification of binarized neural networks.
- Rudiger Ehlers.
  Formal verification of piece-wise linear feed-forward neural networks.
- Matteo Fischetti and Jason Jo.
  Deep neural networks as 0-1 mixed integer linear programs: A feasibility study.
- Vincent Tjeng and Russ Tedrake.
  Verifying neural networks with mixed integer programming

Verification of NN

- Pulina and Tacchella 2010.
  An Abstraction-Refinement Approach to Verification of Artificial Neural Networks.
- Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, D. Vytiniotis, Aditya Nori, and A. Criminisi.
  Measuring neural net robustness with constraints
- Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer.
  Reluplex: An efficient SMT solver for verifying deep neural networks.
- Xiaov
  Safety
- Svyat ... ***nstrained***
  imag
- Nina ... ***fying***
  prop
- Chih-
  Maxi
- Chih-Hong Cheng, Georg Nuhrenberg, and Harald Ruess.
  Verification of binarized neural networks.
- Rudiger Ehlers.
  Formal verification of piece-wise linear feed-forward neural networks.
- Matteo Fischetti and Jason Jo.
  Deep neural networks as 0-1 mixed integer linear programs: A feasibility study.
- Vincent Tjeng and Russ Tedrake.
  Verifying neural networks with mixed integer programming

- Scalability (size of the network, dimensionality of perturbations)

Verification of NN

	Core Techniques	Workable Layer Types	Running Time on ACAS Xu	Computational Complexity	Applicable to State-of-the-art Networks?	Maximal No. of Layers in Tested DNNs
SHERLOCK	MILP + Local Search	ReLu	No experiment	NP w.r.t. neuron no.	No (~6845 neurons)	6
Reluplex	SMT + LP	ReLu	O(10^4)-O(10^6)	NP w.r.t. neuron no.	No (~ 300 neurons)	6
Planet	SAT + LP	ReLu, maxpooling	O(10^3)	NP w.r.t. neuron no.	No (~ 300 neurons)	6
MIP	MIP	ReLu, maxpooling	O(10^3)	NP w.r.t. neuron no.	No (~ 300 neurons)	6
BaB	MIP + BaB	ReLu, maxpooling	O(10^2)	NP w.r.t. neuron no.	No (~ 300 neurons)	6
DeepGO (this paper)	GO + Lipschitz Continuty	Layer with Lipschitz Continuty (Sigmod, Tanh, max-pooling, ReLu, etc)	O(10^2)	NP w.r.t. changed input dimensions	Yes (millions of neurons)	19

Figure 8: A high-level comparison with state-of-the-art methods: SHERLOCK [10], Reluplex [7], Planet [26], MIP [11, 9] and BaB [12].

IJCAI'18:
Reachability Analysis of Deep Neural Networks with Provable Guarantees
Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska

Verification of NN

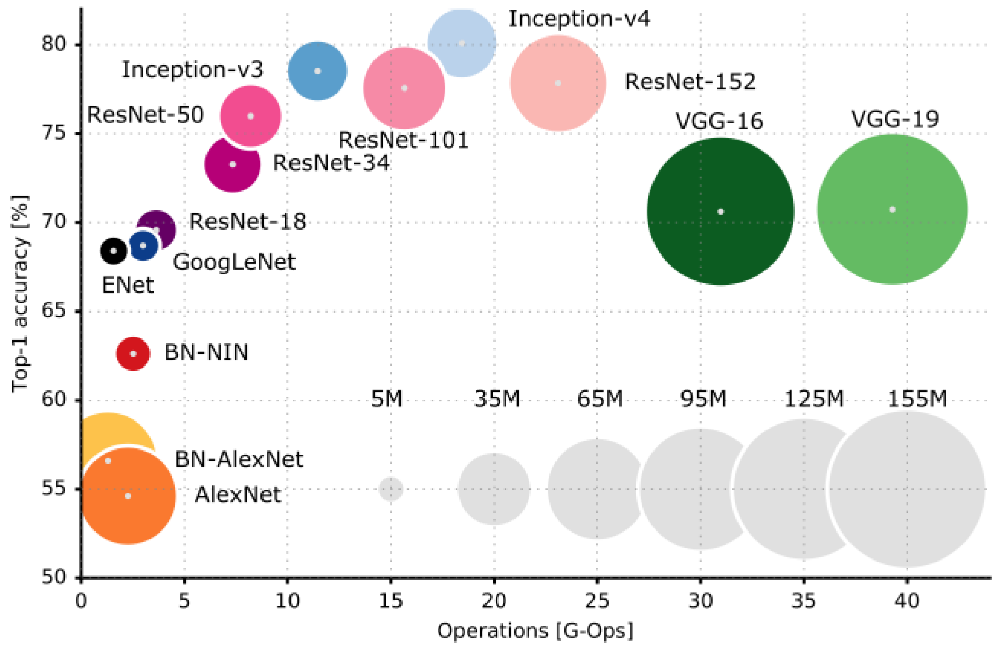
	Core Techniques	Workable Layer Types	Running Time on ACAS Xu	Computational Complexity	Applicable to State-of-the-art Networks?	Maximal No. of Layers in Tested DNNs
SHERLOCK	MILP + Local Search	ReLu	No experiment	NP w.r.t. neuron no	No (~6845 neurons)	6
Reluplex	SMT + LP	ReLu	O(10^4)-O(10^6)	NP w.r.t. neuron no	No (~ 300 neurons)	6
Planet	SAT + LP	ReLu, maxpooling	O(10^3)	NP w.r.t. neuron no	No (~ 300 neurons)	6
MIP	MIP	ReLu, maxpooling	O(10^3)	NP w.r.t. neuron no	No (~ 300 neurons)	6
BaB	MIP + BaB	ReLu, maxpooling	O(10^2)	NP w.r.t. neuron no	No (~ 300 neurons)	6
DeepGO (this paper)	GO + Lipschitz Continuity	Layer with Lipschitz Continuty (Sigmod, Tanh, max-pooling, ReLu, etc)	O(10^2)	NP w.r.t. changed input dimensions	Yes (millions of neurons)	19

Figure 8: A high-level comparison with state-of-the-art methods: SHERLOCK [10], Reluplex [7], Planet [26], MIP [11, 9] and BaB [12].

IJCAI'18:
Reachability Analysis of Deep Neural Networks with Provable Guarantees
Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska

Neural Networks

Neural Networks



[Alfredo Canziani, Adam Paszke, Eugenio Culurciello
An Analysis of Deep Neural Network Models for Practical Applications]

Binarized Neural Networks

Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1
Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, Yoshua Bengio

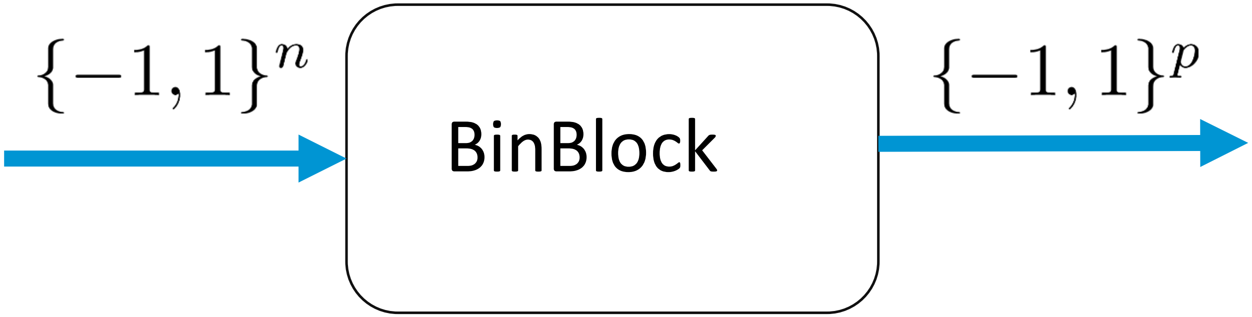
Why Binarized Neural Networks

- special class of NN, where most parameters are binary {-1,1}
- allows fast binary matrix multiplication (7x speed up on a GPU).
- produces smaller size models as most parameters are binary

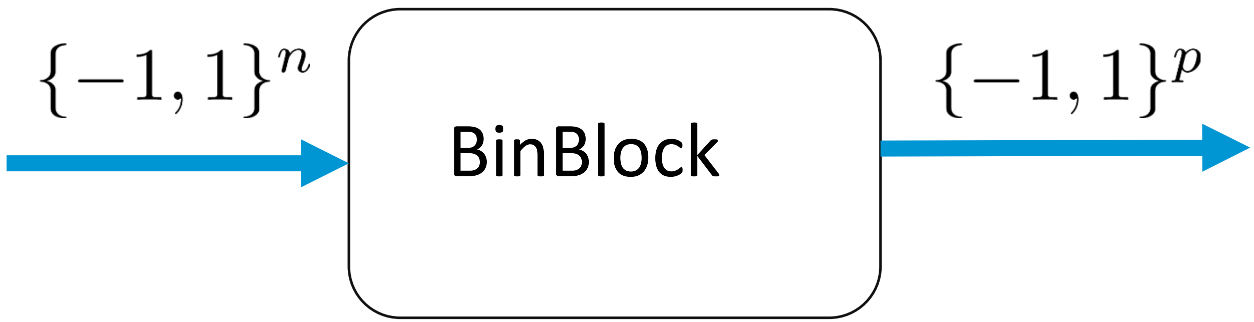
Binarized neural networks I Hubara, M Courbariaux, D Soudry, R El-Yaniv, Y Bengio Advances in Neural Information Processing Systems, 4107-4115	634 *	2016
Binaryconnect: Training deep neural networks with binary weights during propagations M Courbariaux, Y Bengio, JP David Advances in neural information processing systems, 3123-3131	483	2015

vmware

Binarized Building Block

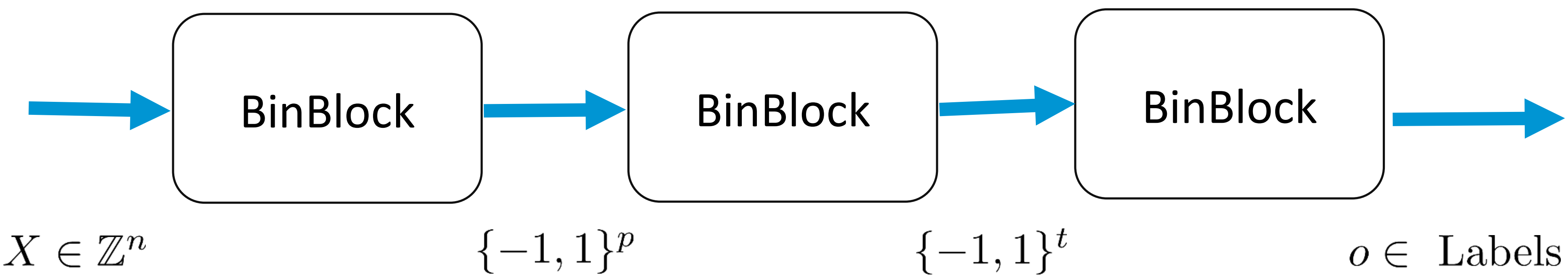


Binarized Building Block



A block can be encoded as SAT

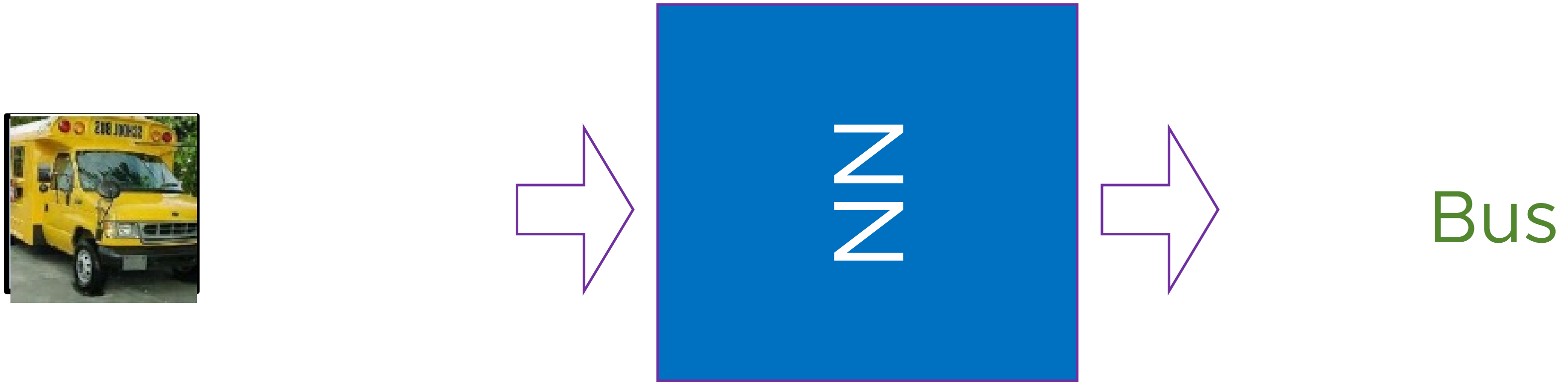
Binarized Building Block

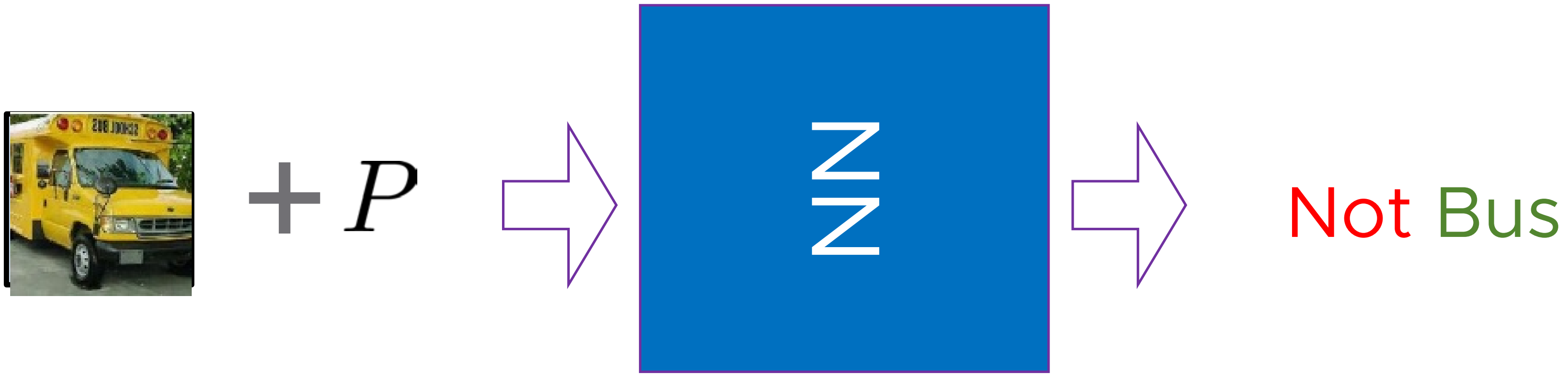


SAT-based approach to adversarial examples

Verifying Properties of Binarized Deep Neural Networks

N.Narodytska, with S. Kasiviswanathan, L. Ryzhyk, M. Sagiv, T. Walsh


NN
Bus


+ P
NN
Not Bus

Boolean encoding

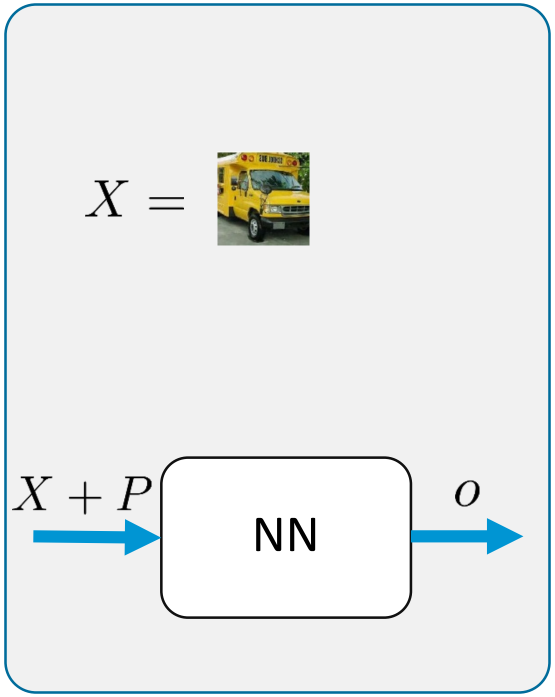
Network function

- Adversarial goal
- Constraints on perturbation

Step 1

Step 2

Boolean encoding

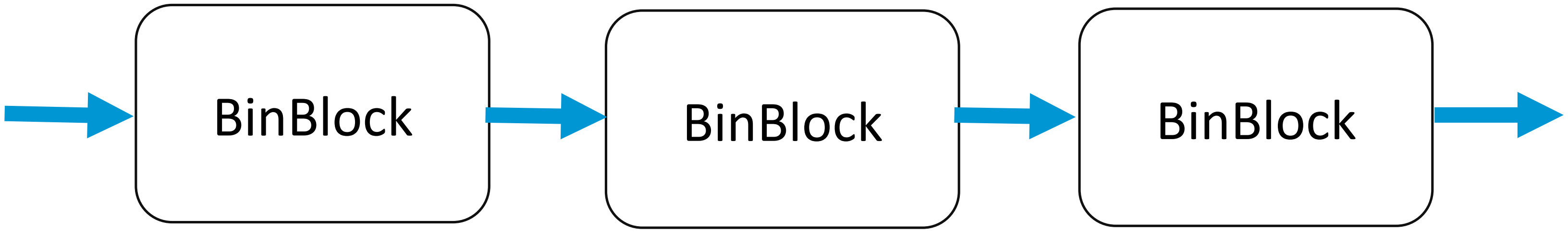


- Adversarial goal
- Constraints on perturbation

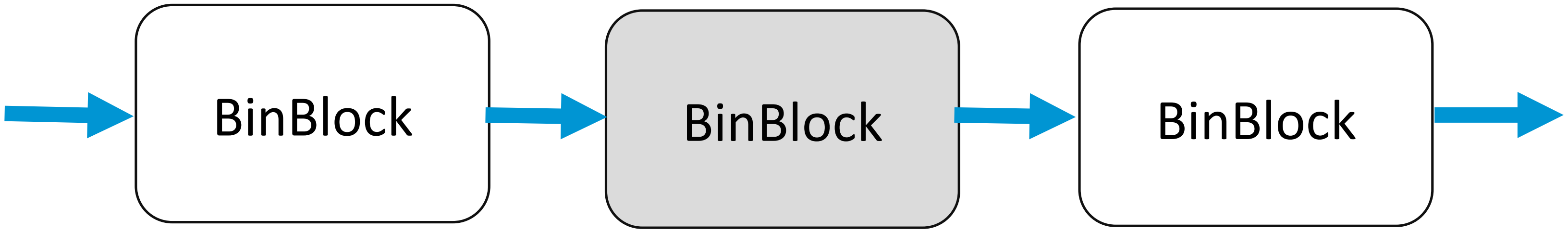
Step 1

Step 2

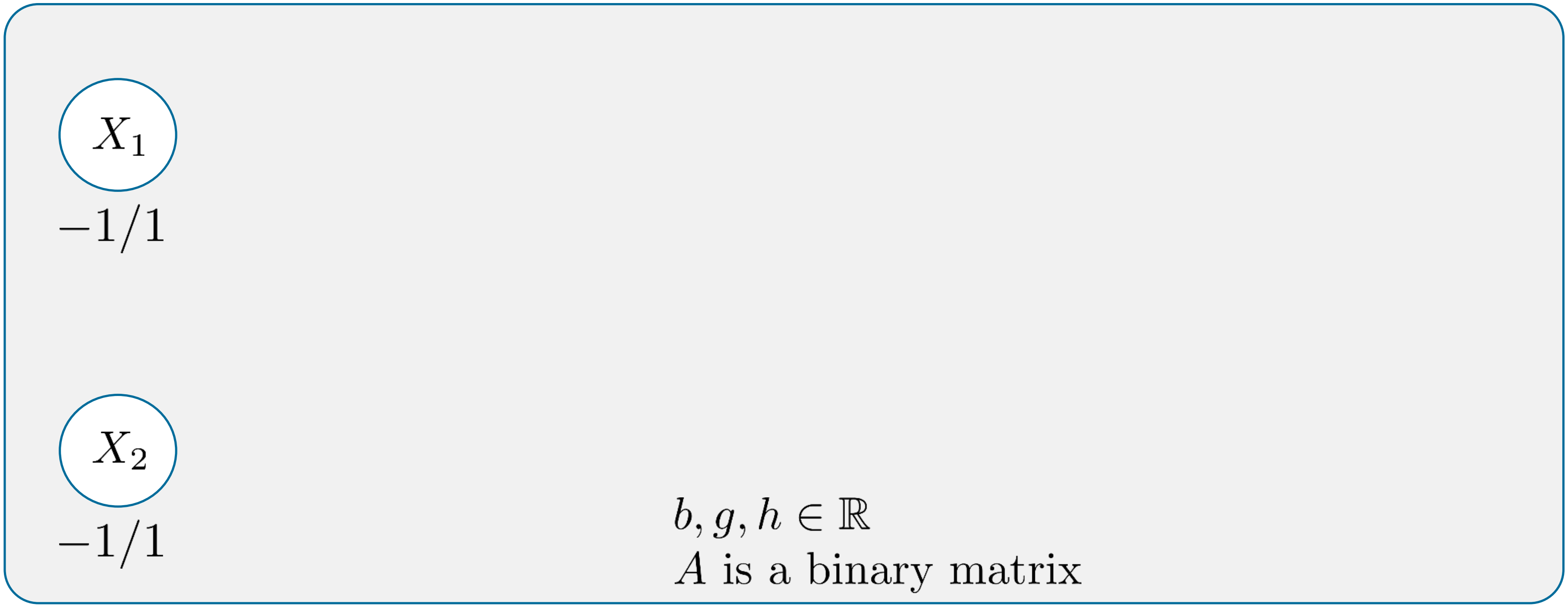
Block-wise BNN encoding



Block-wise BNN encoding

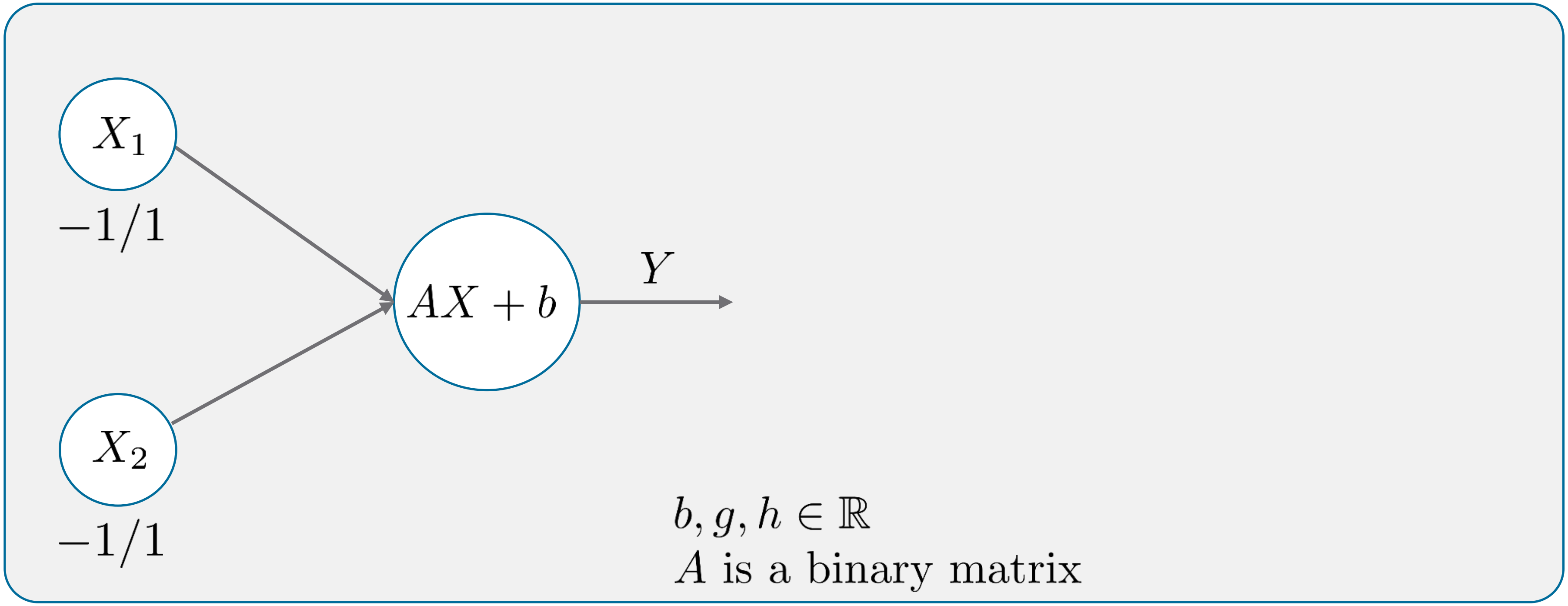


Block-wise BNN encoding



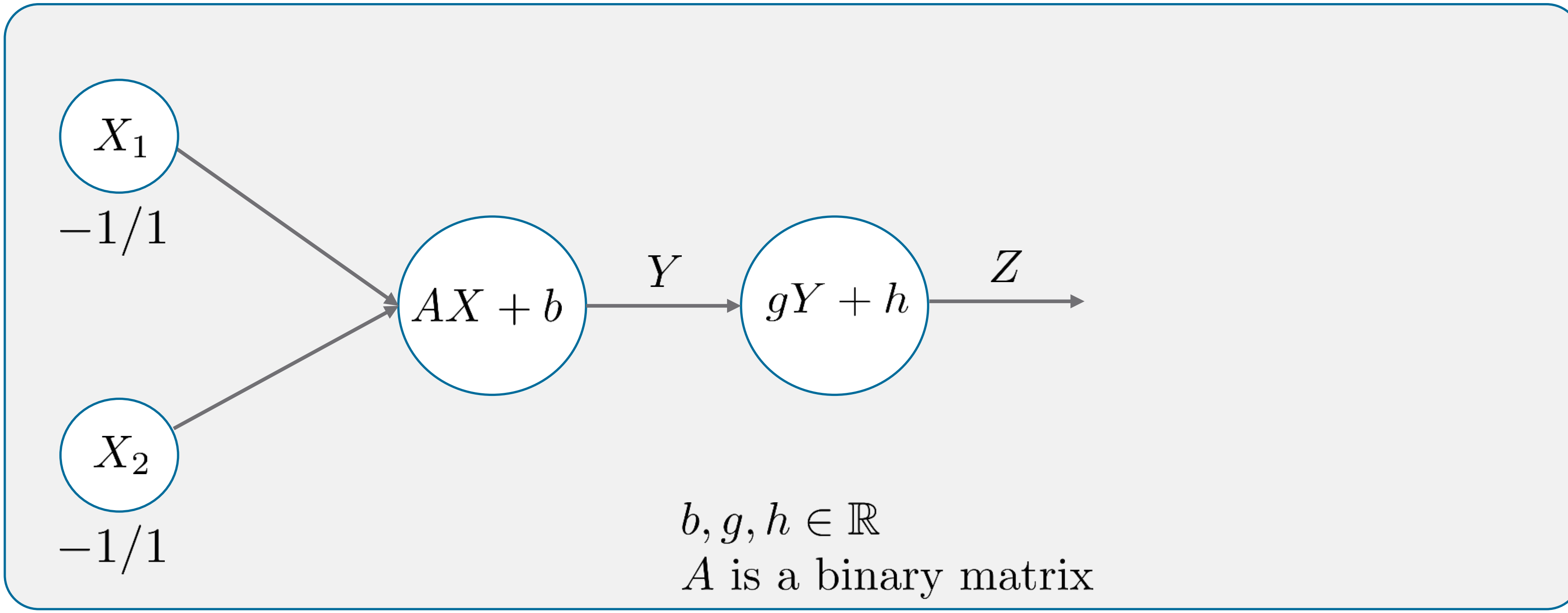
Block

Block-wise BNN encoding



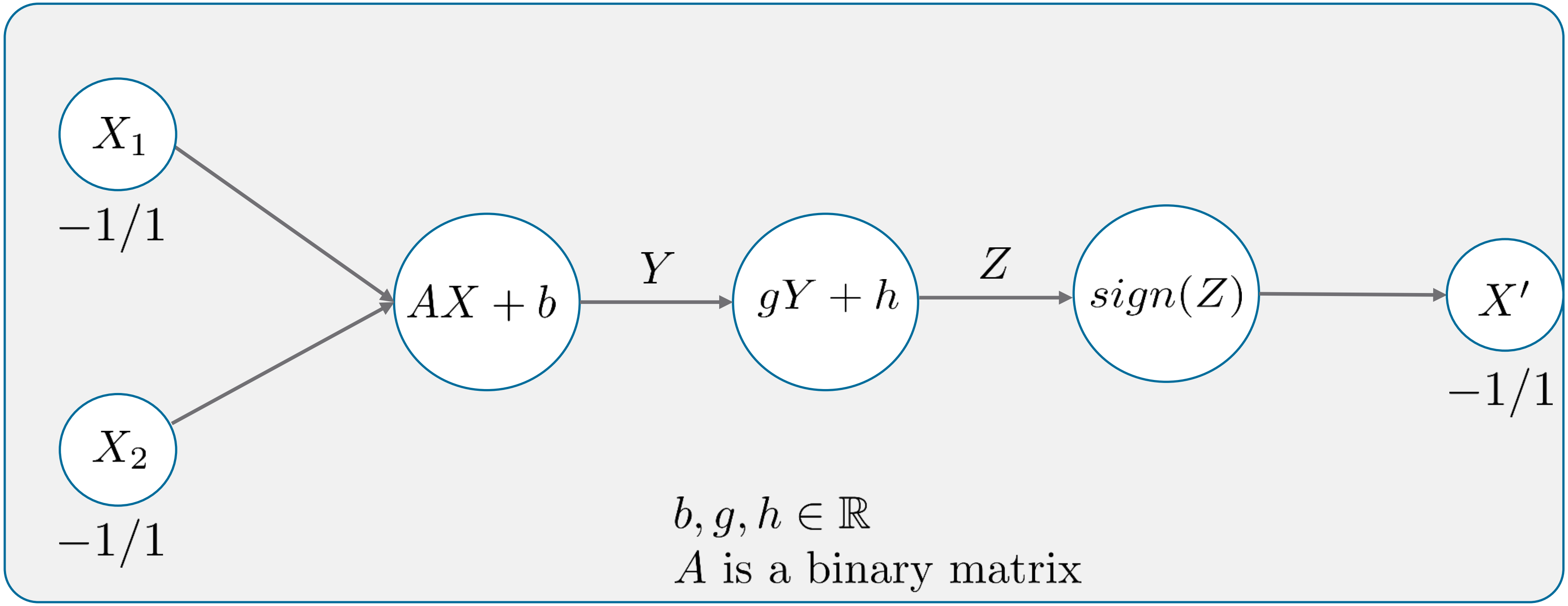
Block

Block-wise BNN encoding



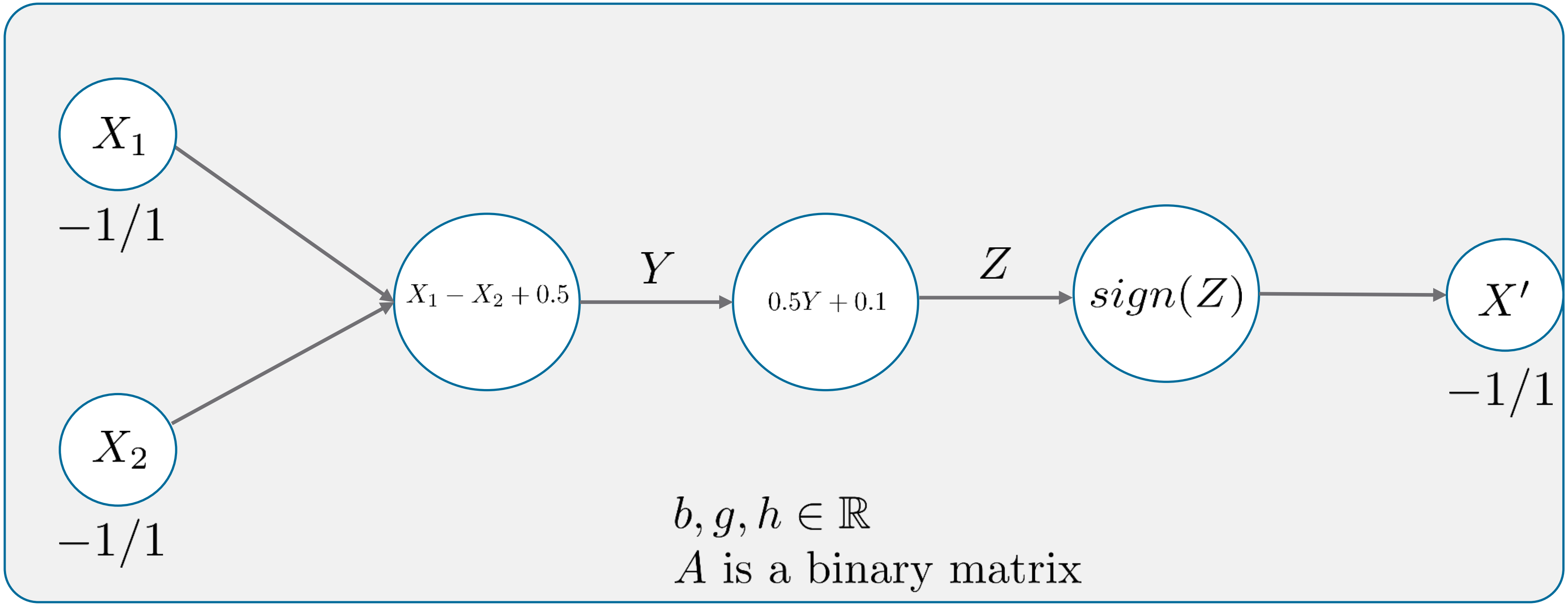
Block

Block-wise BNN encoding



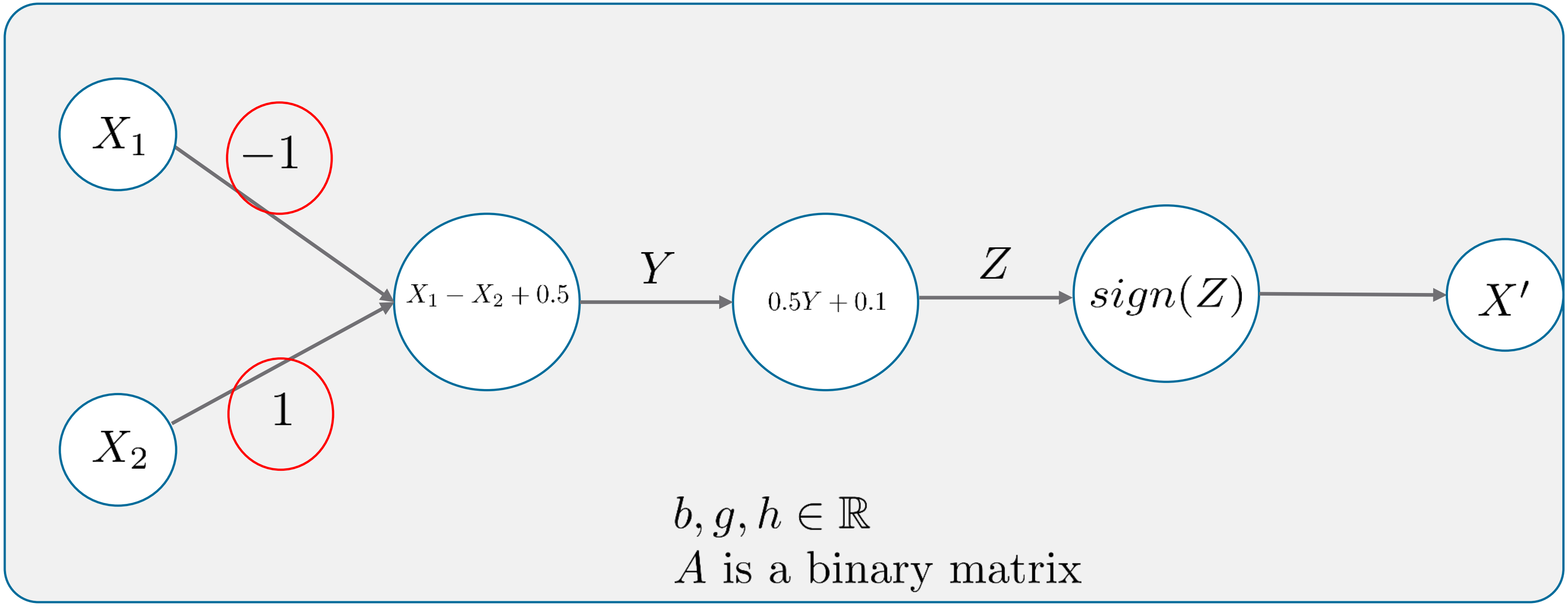
Block

Block-wise BNN encoding



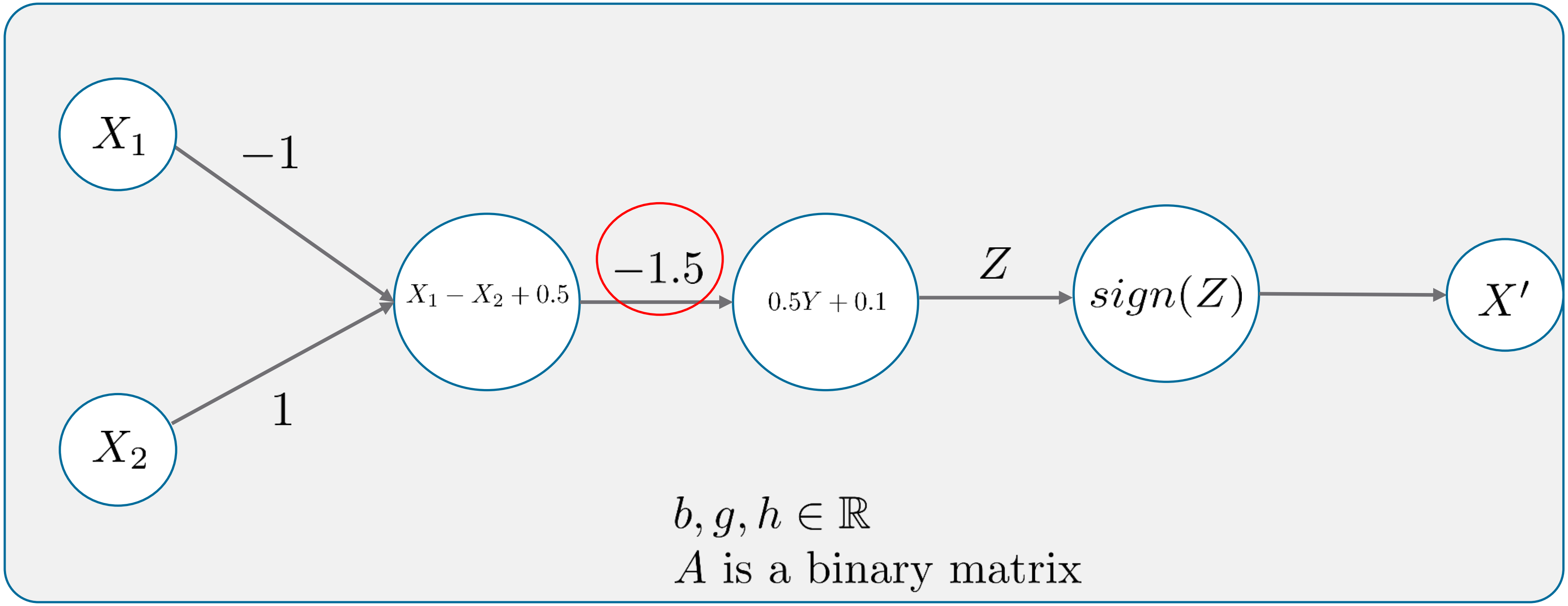
Block

Block-wise BNN encoding



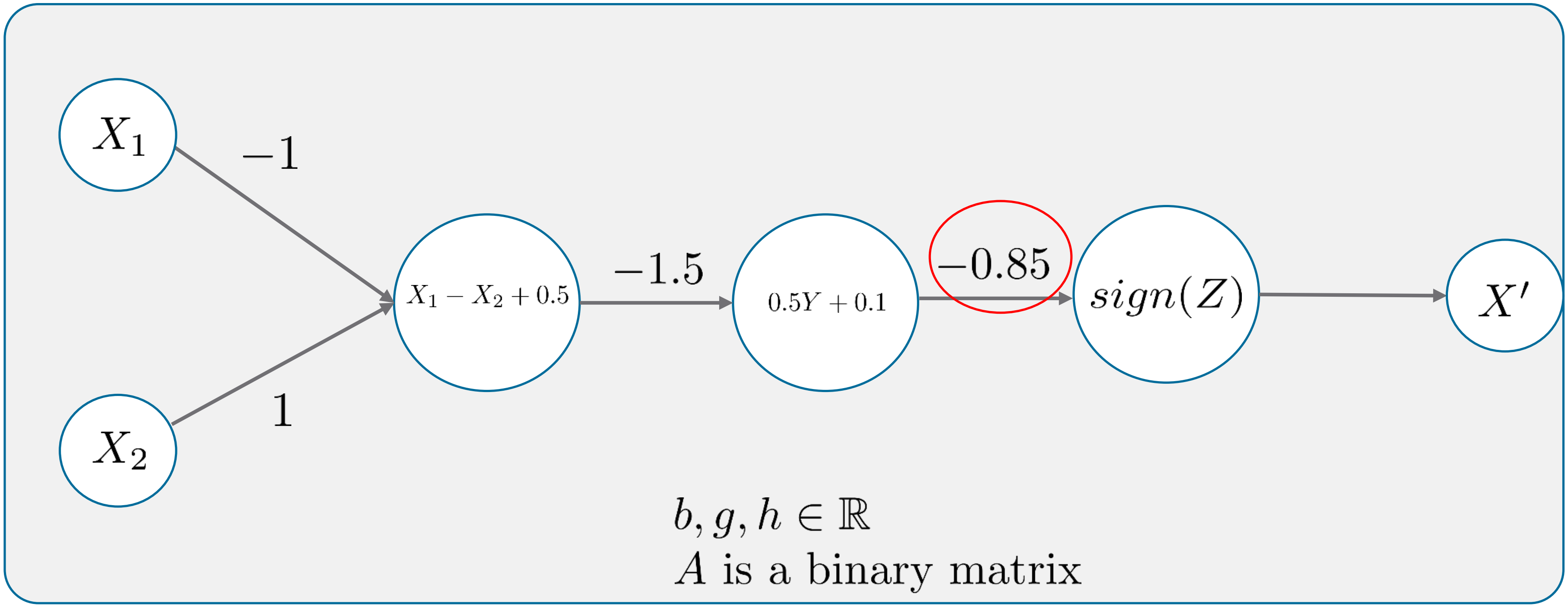
Block

Block-wise BNN encoding



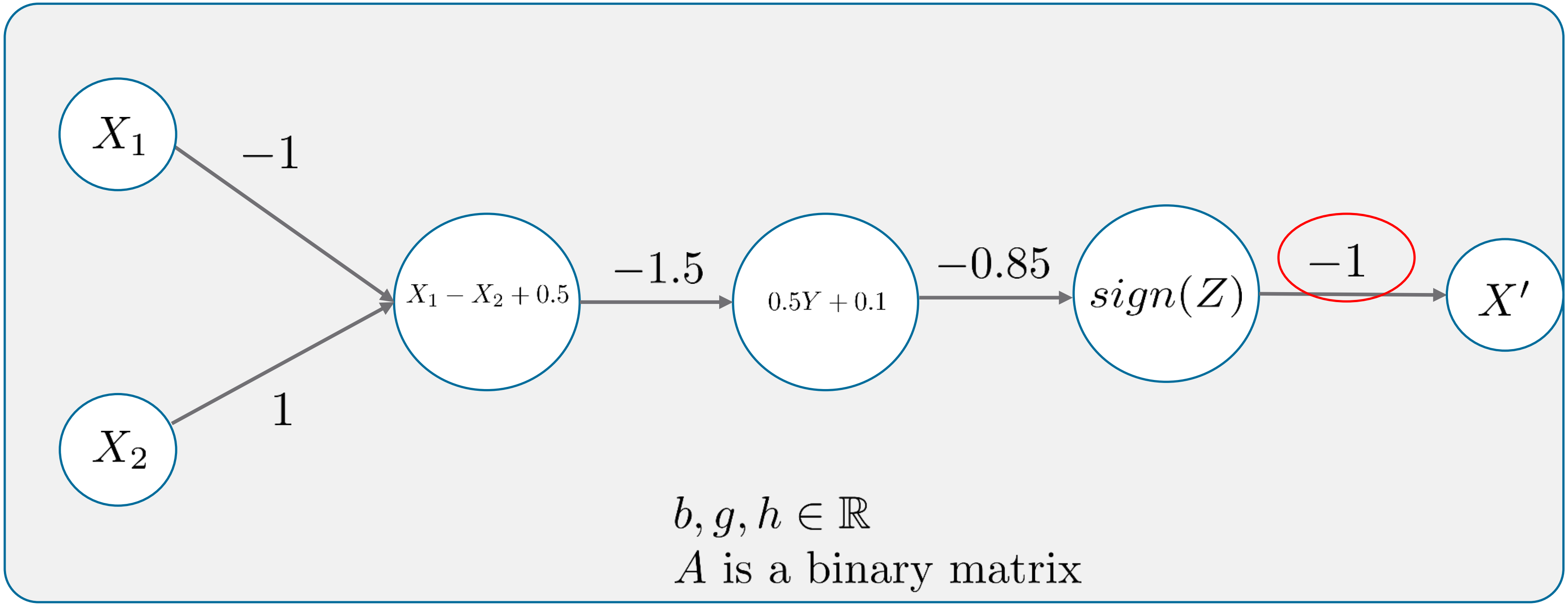
Block

Block-wise BNN encoding



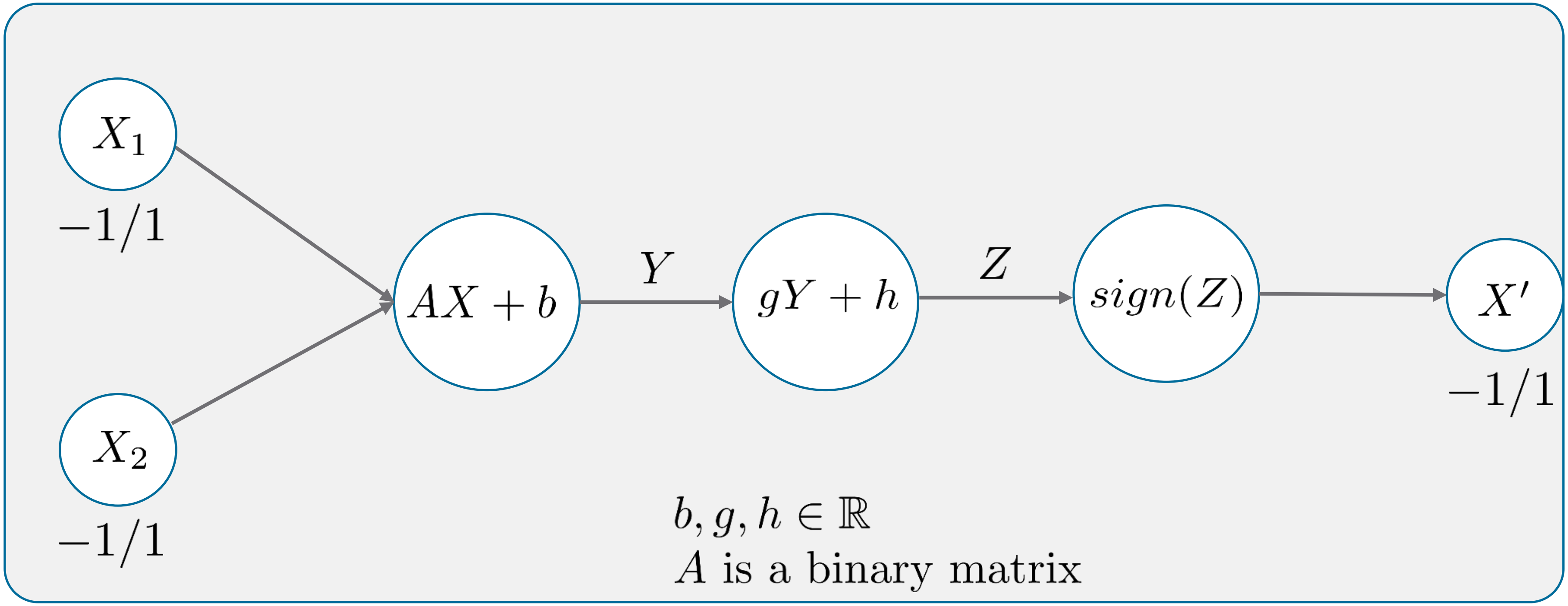
Block

Block-wise BNN encoding



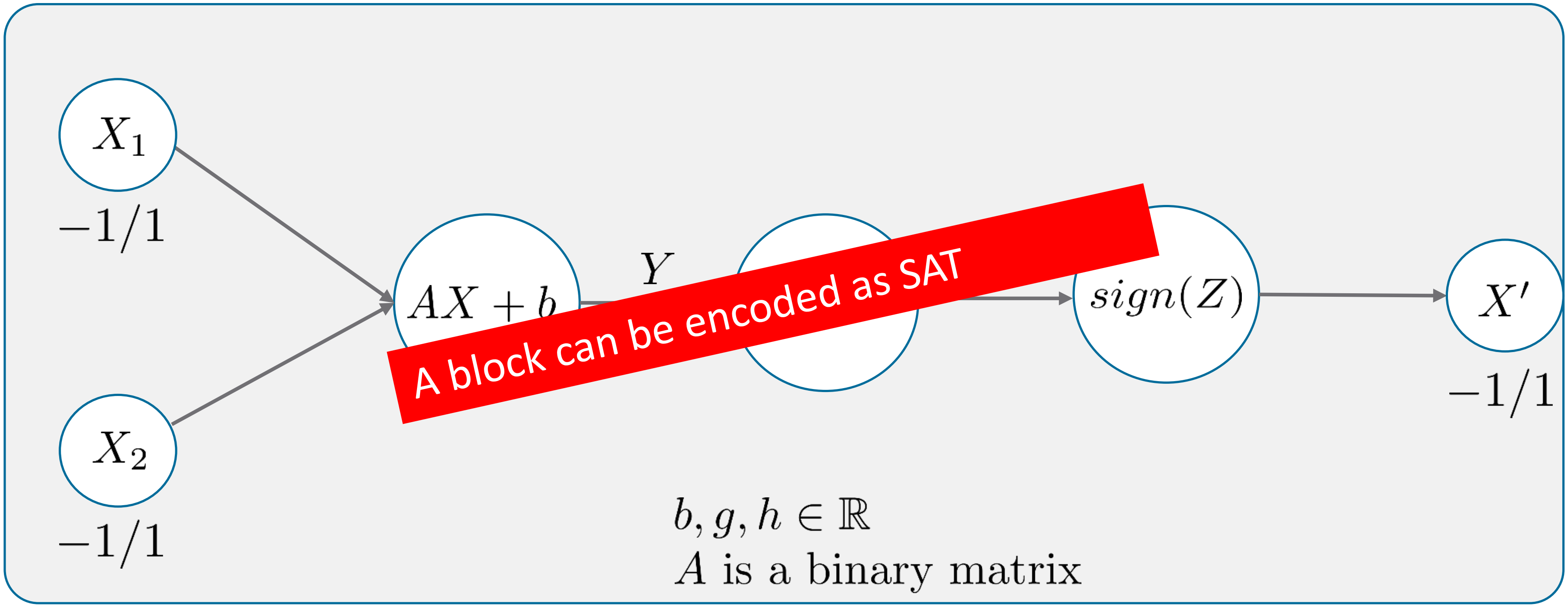
Block

Block-wise BNN encoding



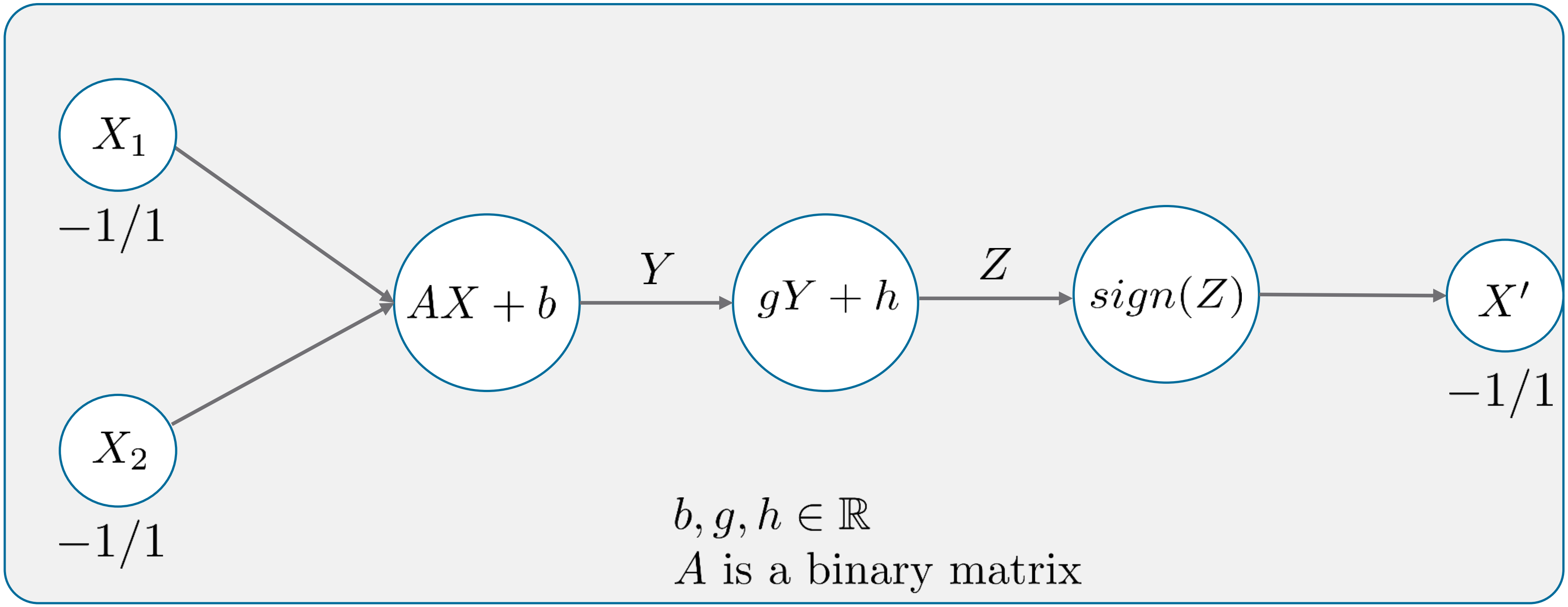
Block

Block-wise BNN encoding



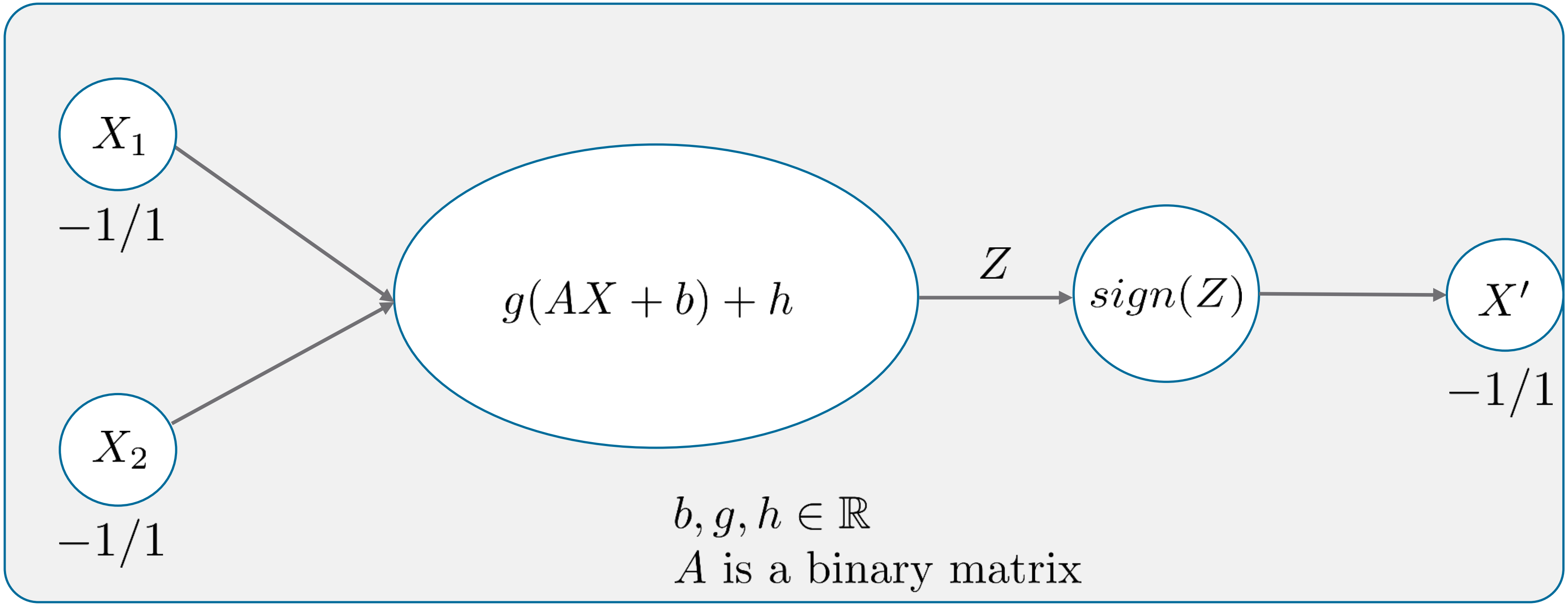
Block

Block-wise BNN encoding



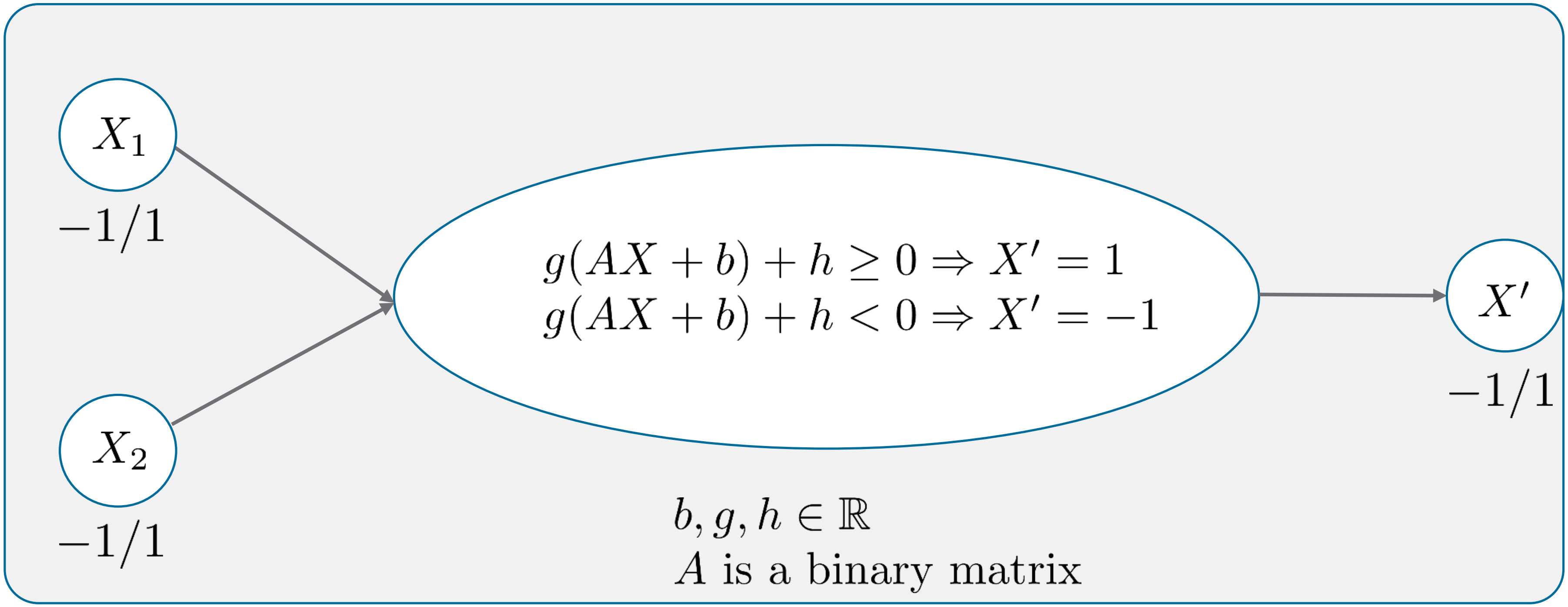
Block

Block-wise BNN encoding



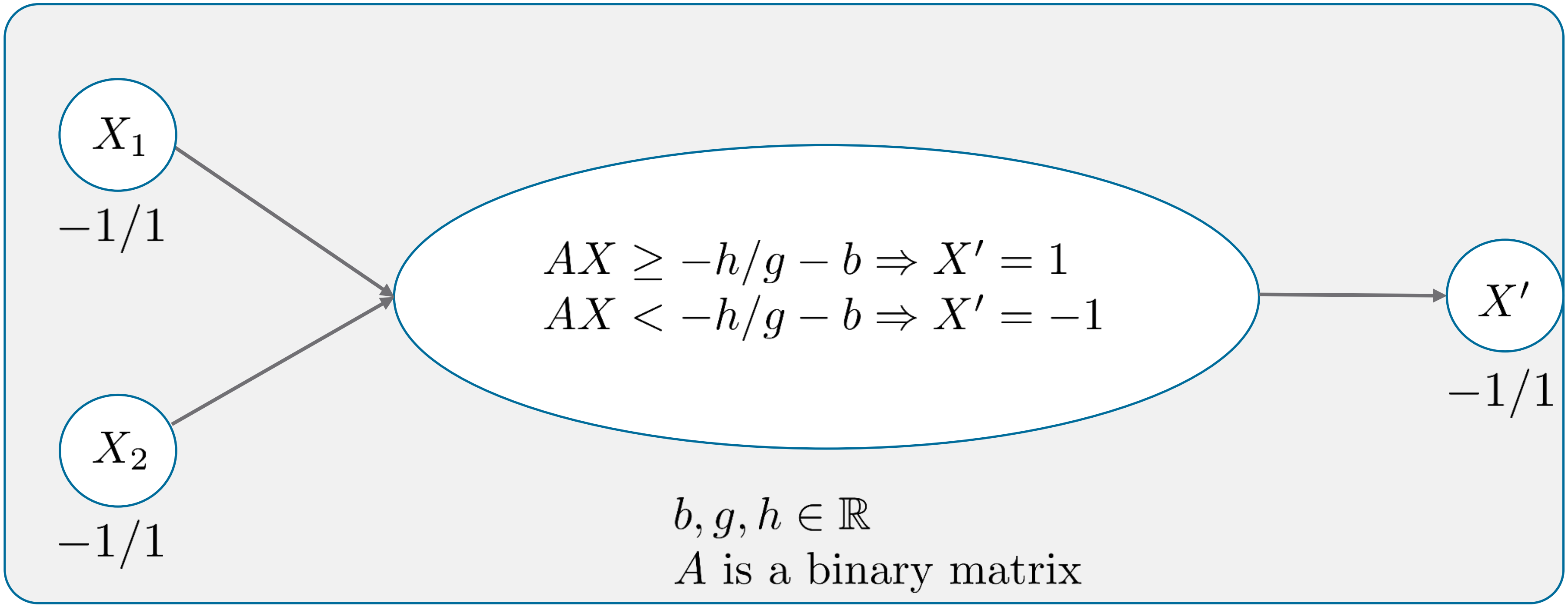
Block

Block-wise BNN encoding



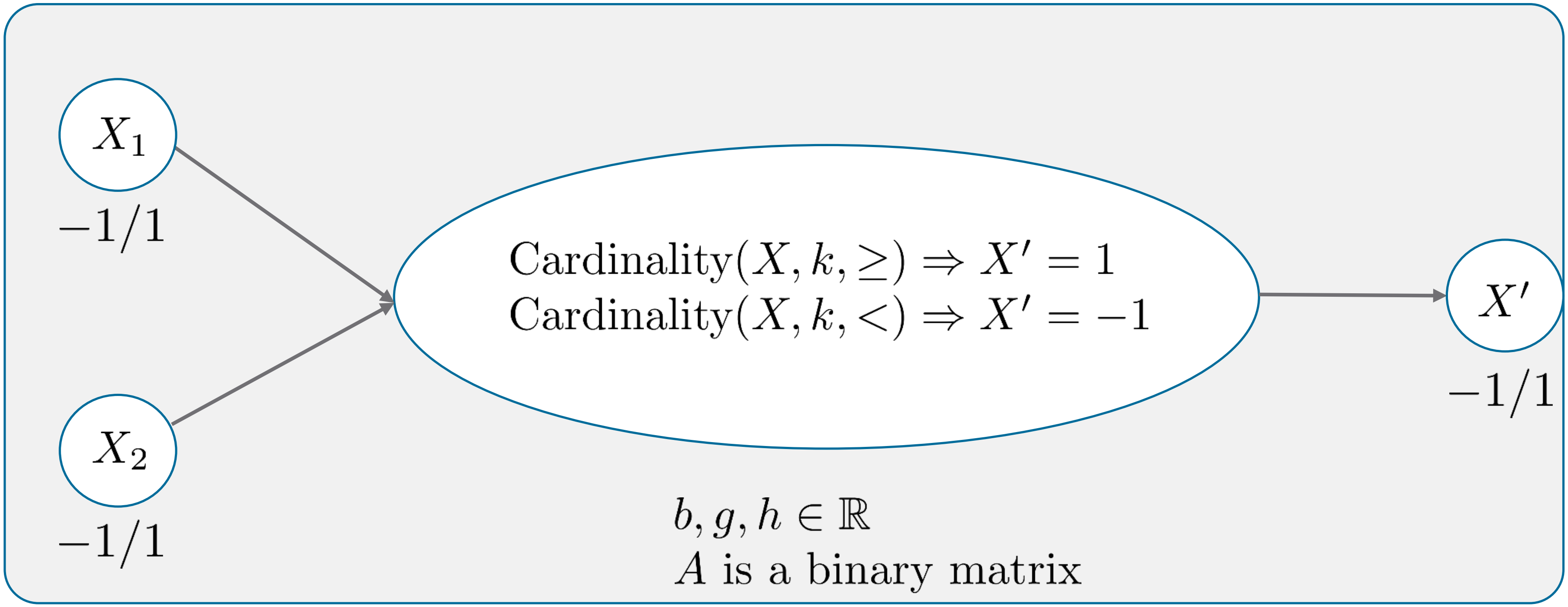
Block

Block-wise BNN encoding



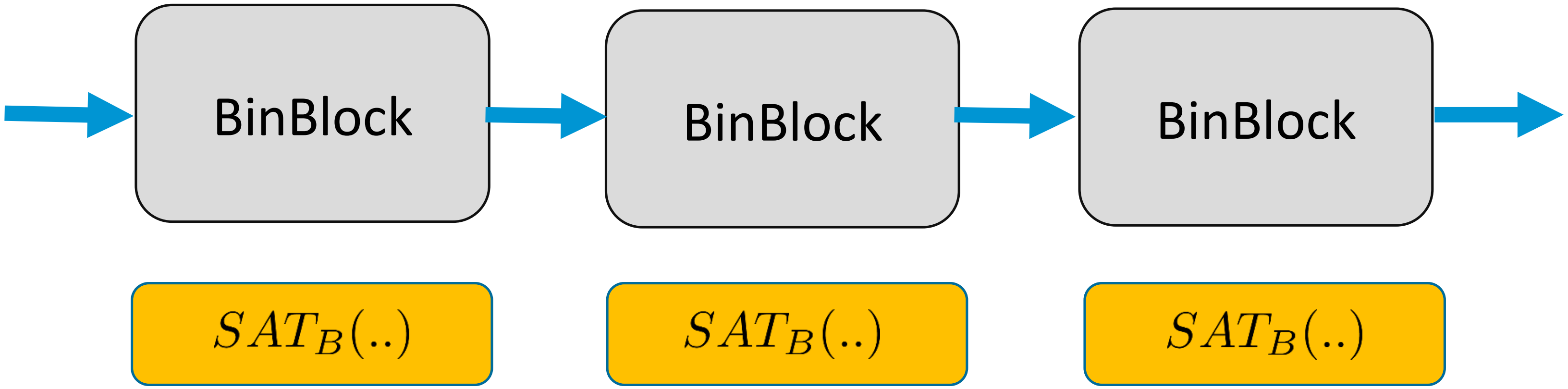
Block

Block-wise BNN encoding

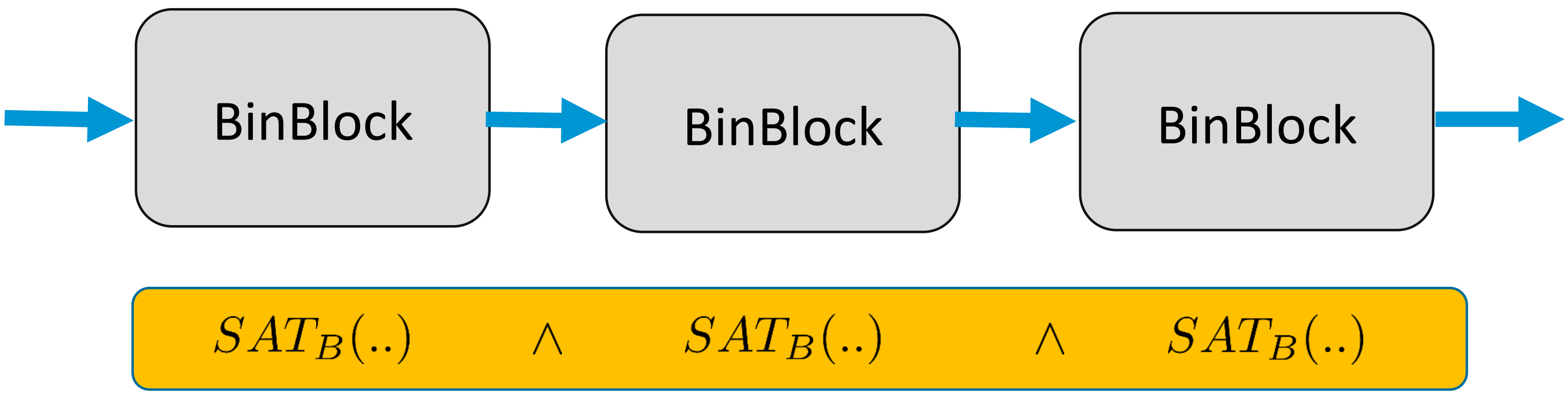


Block

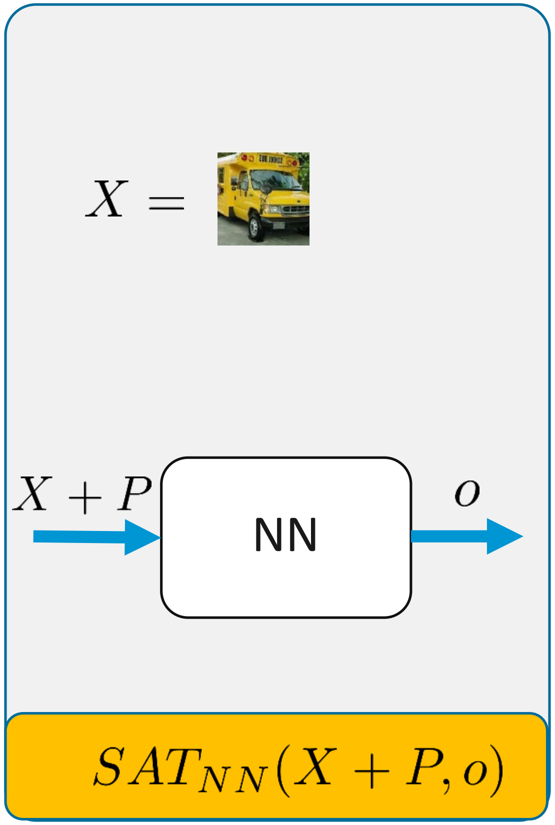
Block-wise BNN encoding



Block-wise BNN encoding



Boolean encoding



Step 1

Step 2

Boolean encoding

$X =$ 

$X + P \rightarrow$ NN $\rightarrow o$

$SAT_{NN}(X + P, o)$

Step 1

- Adversarial goal
- Constraints on perturbation

Step 2

Boolean encoding

$X =$ 

$X + P \rightarrow$ NN $\xrightarrow{o}$

$SAT_{NN}(X + P, o)$

Step 1

$o \neq \text{bus}$

$\max(P) < \epsilon$

Step 2

Boolean encoding

$X =$ 

$X + P \rightarrow \boxed{\text{NN}} \rightarrow o$

$SAT_{NN}(X + P, o)$

Step 1

$o \neq \text{bus}$

$\max(P) < \epsilon$

$SAT_{Ad}(P, bus, o)$

Step 2

Boolean encoding

$X =$ 

$$X + P \rightarrow \boxed{\text{NN}} \rightarrow o$$

$$SAT_{NN}(X + P, o)$$

Step 1

$$o \neq \text{ bus}$$

$$\max(P) < \epsilon$$

$$SAT_{Ad}(P, bus, o)$$

Step 2

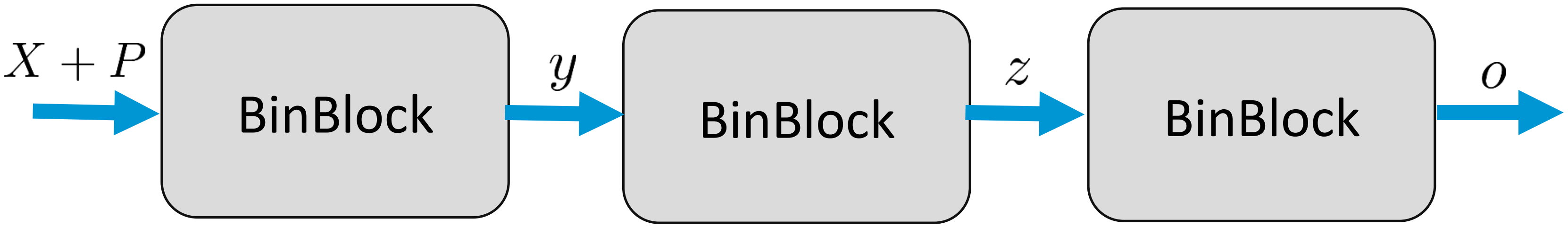
Boolean encoding

$$SAT_{NN}(X + P, o)$$

$$SAT_{Ad}(P, bus, o)$$

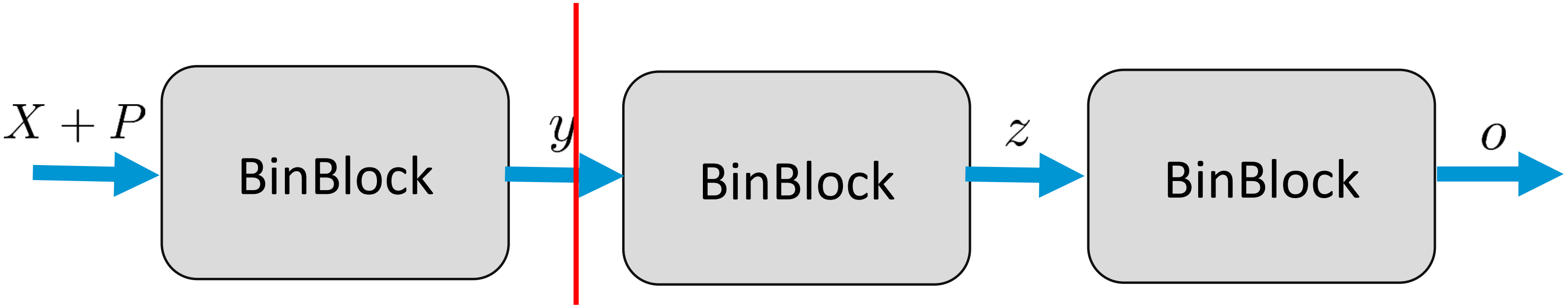
Search procedure

Search procedure



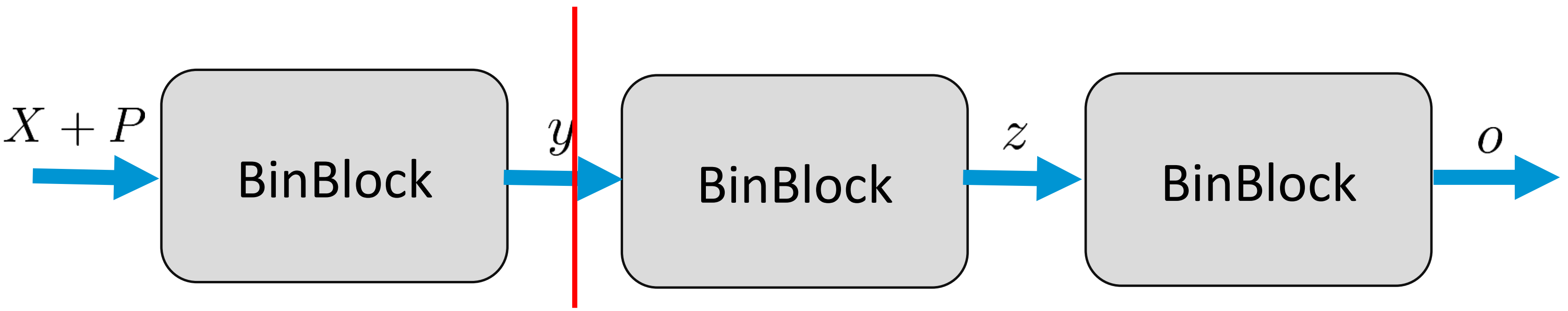
$$Init(P) \wedge SAT_B(X+P, y) \wedge SAT_B(y, z) \wedge SAT_B(z, o) \wedge Ad(o)$$

Search procedure



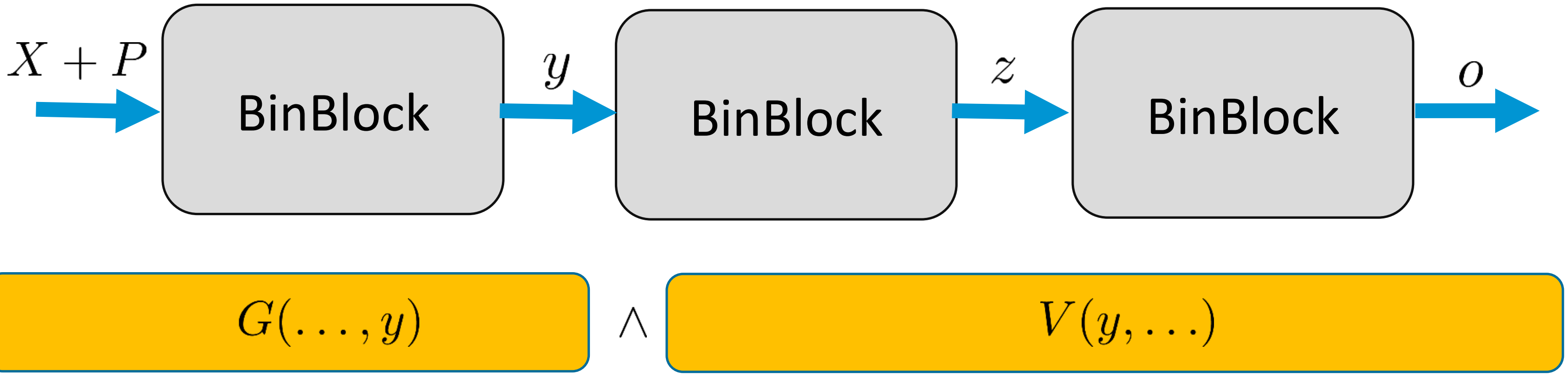
$$Init(P) \wedge SAT_B(X+P, y) \wedge SAT_B(y,z) \wedge SAT_B(z,o) \wedge Ad(o)$$

Search procedure

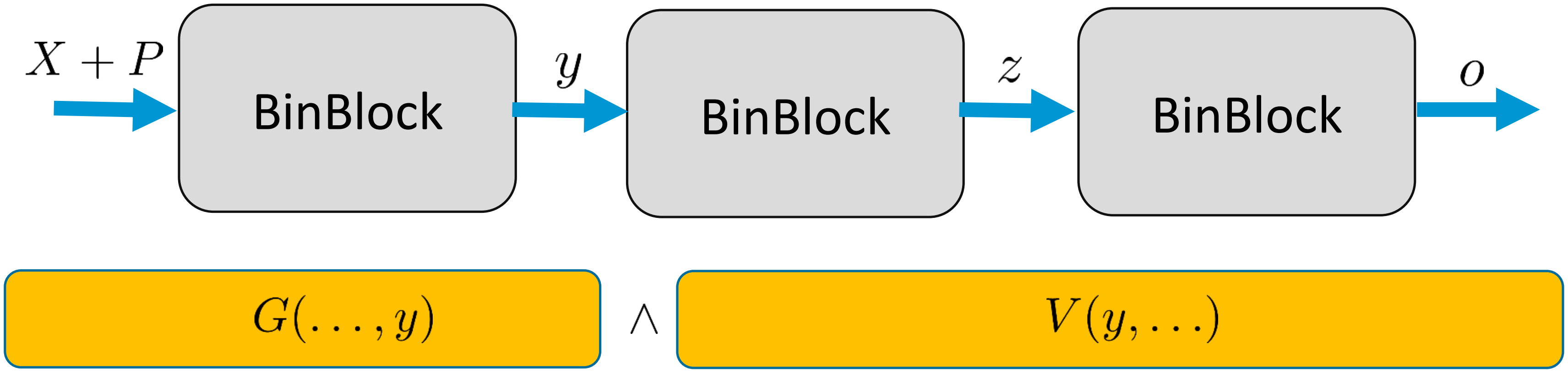


$$\boxed{Init(P) \wedge SAT_B(X+P, y)} \wedge \boxed{SAT_B(y,z) \quad \wedge \quad SAT_B(z,o) \quad \wedge \quad Ad(o)}$$

Search procedure

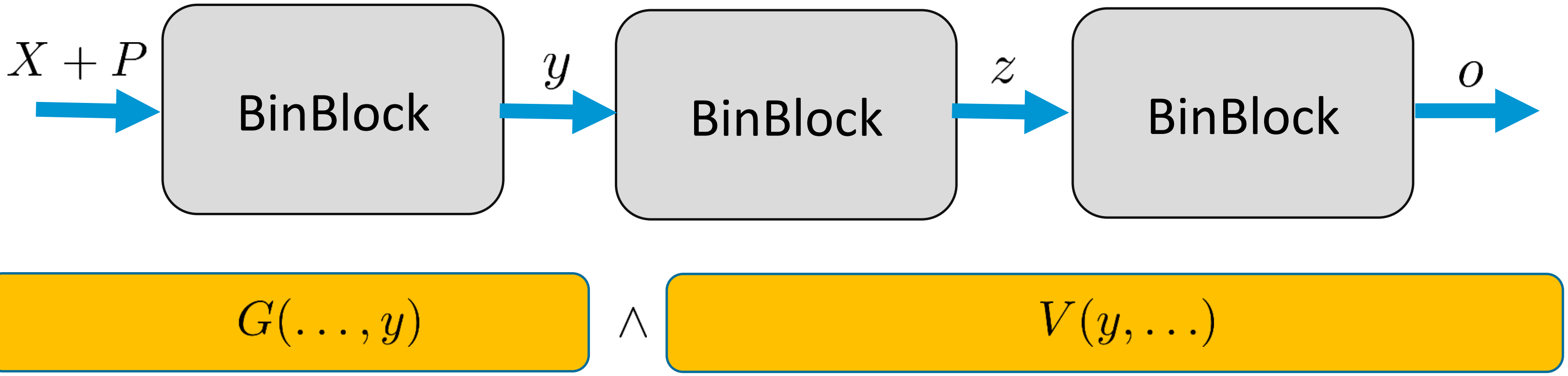


Search procedure



$$G(\ldots, y) \cap V(y, \ldots)$$

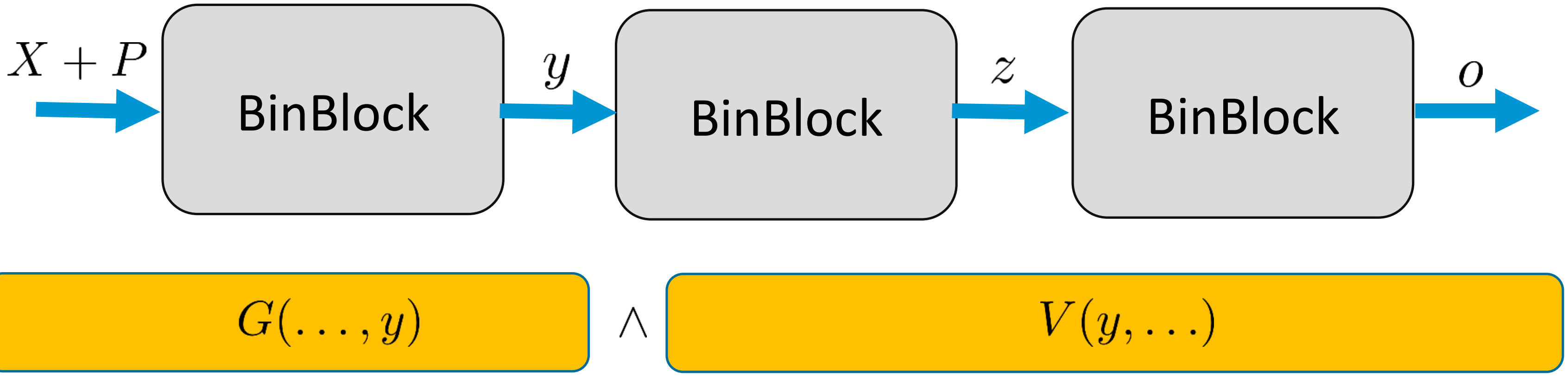
Search procedure



$$G(\ldots, y) \cap V(y, \ldots)$$

Craig interpolants

Search procedure



$$G(\ldots, y) \cap V(y, \ldots)$$

Craig interpolants

Search procedure

$G(\ldots, y)$ $\wedge$ $V(y, \ldots)$

$$Solve(G(y)) \xrightarrow{\hat{y}}$$

Search procedure

$G(\ldots, y)$ $\wedge$ $V(y, \ldots)$

$$Solve(G(y)) \xrightarrow{\hat{y}} Solve(V(y), y = \hat{y})$$

✗ $\rightarrow$ $Compute(I(y))$

✓ $\rightarrow$ $returnP$

Search procedure

$G(\ldots, y) \wedge V(y, \ldots)$

$$Solve(G(y)) \xrightarrow{\hat{y}} Solve(V(y), y = \hat{y})$$

$$returnP$$

$$G(y) = G(y) \wedge \neg I(y) \xleftarrow{I(y)} Compute(I(y))$$

Search procedure

$$G(\dots, y) \quad \land \quad V(y, \dots)$$

$$\begin{tikzcd}
Solve(G(y)) \arrow[r, "\hat{y}"] & Solve(V(y), y = \hat{y}) \arrow[d] \arrow[r] & returnP \\
G(y) = G(y) \land \neg I(y) \arrow[u] & Compute(I(y)) \arrow[l, "I(y)"'] &
\end{tikzcd}$$

Experiments

Experiments

Dataset: MNIST, MNIST-ROT, MNIST-BACK
Network: BNN with FC layers
Problem: Untargeted adversarial examples
Encodings: SAT, ILP, CEG-SAT
+ few simplifications, e.g. un-normalized and binarized inputs

Experiments

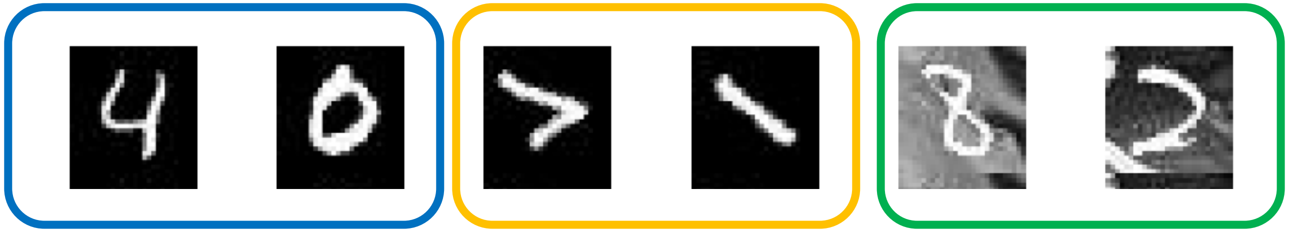
Dataset: MNIST, MNIST-ROT, MNIST-BACK

Network: BNN with FC layers

Problem: Untargeted adversarial examples

Encodings: SAT, ILP, CEG-SAT

+ few simplifications, e.g. un-normalized and binarized inputs



Experiments

Dataset: MNIST, MNIST-ROT, MNIST-BACK
Network: BNN with FC layers
Problem: Untargeted adversarial examples
Encodings: SAT, ILP, CEG-SAT
+ few simplifications, e.g. un-normalized and binarized inputs

Vary:

- the value of maximum perturbation ε

Untargeted adversarial examples

Input: ( , 4)

Untargeted adversarial examples

Input: ( , 4)

Goal:

Untargeted adversarial examples

Input: ( , 4)

Goal:

Adversarial X’ =  + P,

Untargeted adversarial examples

Input: ( , 4)

Goal:

Adversarial X’ =  + P,

$$\max(P_1 \ldots P_n) < \varepsilon$$

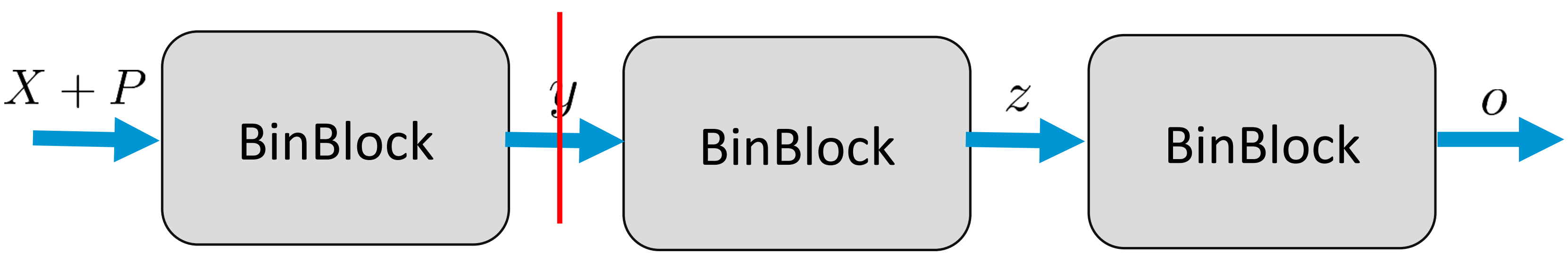
Untargeted adversarial examples

Input: ( , 4)

Goal:

Adversarial X’ =  + P,

max(P_1… P_n) < ε



MNIST

	Solved instances (out of 200)									Certifiably ϵ-robust		
	MNIST			MNIST-rot			MNIST-back-image					
	SAT	ILP	CEG	SAT	ILP	CEG	SAT	ILP	CEG	SAT	ILP	CEG
	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#	#	#
$\epsilon = 1$	180 (77.3)	130 (31.5)	171 (34.1)	179 (57.4)	125 (10.9)	197 (13.5)	191 (18.3)	143 (40.8)	191 (12.8)	138	96	138
$\epsilon = 3$	187 (77.6)	148 (29.0)	181 (35.1)	193 (61.5)	155 (9.3)	198 (13.7)	107 (43.8)	67 (52.7)	119 (44.6)	20	5	21
$\epsilon = 5$	191 (79.5)	165 (29.1)	188 (36.3)	196(62.7)	170(11.3)	198(13.7)	104 (48.8)	70 (53.8)	116 (47.4)	3	–	4

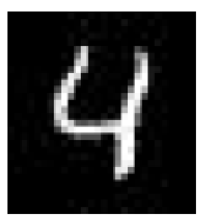
Table 2: Results on MNIST, MNIST-rot and MNIST-back-image datasets.

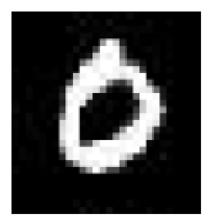
vmware®

MNIST

	Solved instances (out of 200)									Certifiably ϵ-robust		
	MNIST			MNIST-rot			MNIST-back-image					
	SAT	ILP	CEG	SAT	ILP	CEG	SAT	ILP	CEG	SAT	ILP	CEG
	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#	#	#
$\epsilon = 1$	180 (77.3)	130 (31.5)	171 (34.1)	179 (57.4)	125 (10.9)	197 (13.5)	191 (18.3)	143 (40.8)	191 (12.8)	138	96	138
$\epsilon = 3$	187 (77.6)	148 (29.0)	181 (35.1)	193 (61.5)	155 (9.3)	198 (13.7)	107 (43.8)	67 (52.7)	119 (44.6)	20	5	21
$\epsilon = 5$	191 (79.5)	165 (29.1)	188 (36.3)	196(62.7)	170(11.3)	198(13.7)	104 (48.8)	70 (53.8)	116 (47.4)	3	–	4

Table 2: Results on MNIST, MNIST-rot and MNIST-back-image datasets.

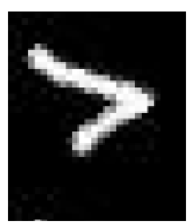


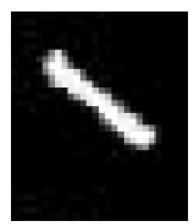


MNIST-ROT

	Solved instances (out of 200)									Certifiably ϵ-robust		
	MNIST			MNIST-rot			MNIST-back-image					
	SAT	ILP	CEG	SAT	ILP	CEG	SAT	ILP	CEG	SAT	ILP	CEG
	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#	#	#
$\epsilon = 1$	180 (77.3)	130 (31.5)	171 (34.1)	179 (57.4)	125 (10.9)	197 (13.5)	191 (18.3)	143 (40.8)	191 (12.8)	138	96	138
$\epsilon = 3$	187 (77.6)	148 (29.0)	181 (35.1)	193 (61.5)	155 (9.3)	198 (13.7)	107 (43.8)	67 (52.7)	119 (44.6)	20	5	21
$\epsilon = 5$	191 (79.5)	165 (29.1)	188 (36.3)	196(62.7)	170(11.3)	198(13.7)	104 (48.8)	70 (53.8)	116 (47.4)	3	–	4

Table 2: Results on MNIST, MNIST-rot and MNIST-back-image datasets.

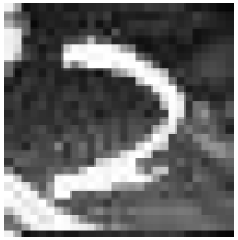




MNIST-BACK

	Solved instances (out of 200)									Certifiably ϵ-robust		
	MNIST			MNIST-rot			MNIST-back-image					
	SAT	ILP	CEG	SAT	ILP	CEG	SAT	ILP	CEG	SAT	ILP	CEG
	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#	#	#
$\epsilon = 1$	180 (77.3)	130 (31.5)	171 (34.1)	179 (57.4)	125 (10.9)	197 (13.5)	191 (18.3)	143 (40.8)	191 (12.8)	138	96	138
$\epsilon = 3$	187 (77.6)	148 (29.0)	181 (35.1)	193 (61.5)	155 (9.3)	198 (13.7)	107 (43.8)	67 (52.7)	119 (44.6)	20	5	21
$\epsilon = 5$	191 (79.5)	165 (29.1)	188 (36.3)	196(62.7)	170(11.3)	198(13.7)	104 (48.8)	70 (53.8)	116 (47.4)	3	–	4

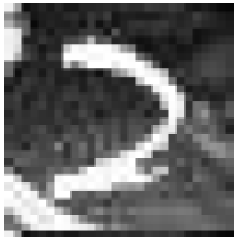
Table 2: Results on MNIST, MNIST-rot and MNIST-back-image datasets.



MNIST-BACK

	Solved instances (out of 200)									Certifiably ϵ-robust		
	MNIST			MNIST-rot			MNIST-back-image					
	SAT	ILP	CEG	SAT	ILP	CEG	SAT	ILP	CEG	SAT	ILP	CEG
	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#solved (t)	#	#	#
$\epsilon = 1$	180 (77.3)	130 (31.5)	171 (34.1)	179 (57.4)	125 (10.9)	197 (13.5)	191 (18.3)	143 (40.8)	191 (12.8)	138	96	138
$\epsilon = 3$	187 (77.6)	148 (29.0)	181 (35.1)	193 (61.5)	155 (9.3)	198 (13.7)	107 (43.8)	67 (52.7)	119 (44.6)	20	5	21
$\epsilon = 5$	191 (79.5)	165 (29.1)	188 (36.3)	196(62.7)	170(11.3)	198(13.7)	104 (48.8)	70 (53.8)	116 (47.4)	3	–	4

Table 2: Results on MNIST, MNIST-rot and MNIST-back-image datasets.



Few observations on properties

Few observations on properties

- Most papers focus on robustness property

Few observations on properties

- Most papers focus on robustness property
- Network equivalence
- Invertibility of the network

Why robustness property?

$$y_1 = 1 \times max(0, 10 \times x_1 + x_2)$$

$$y_2 = 2 \times max(0, -5 \times x_1 + x_2)$$

$$x_1 \in [0.9, 1]$$
$$x_2 \in [-1, 1]$$
$$y_i > 1, i = 1, 2$$

Why robustness property?

$$y_1 = 1 \times max(0, 10 \times x_1 + x_2)$$

$$y_2 = 2 \times max(0, -5 \times x_1 + x_2)$$

$$x_1 \in [0.9, 1]$$
$$x_2 \in [-1, 1]$$
$$y_i > 1, i = 1, 2$$

Why robustness property?

$$y_1 = 1 \times max(0, 10 \times x_1 + x_2)$$

$$y_2 = 2 \times max(0, -5 \times x_1 + x_2)$$

$$\boxed{x_1 \in [0.9, 1]}$$

$$x_2 \in [-1, 1]$$

$$y_i > 1, i = 1, 2$$

Why robustness property?

$$y_1 = 1 \times max(0, 10 \times x_1 + x_2)$$

$$x_1 \in [0.9, 1]$$
$$x_2 \in [-1, 1]$$
$$y_i > 1, i = 1, 2$$

Why robustness property?

$$y_1 = 1 \times \qquad (10 \times x_1 + x_2)$$

$$x_1 \in [0.9, 1]$$
$$x_2 \in [-1, 1]$$
$$y_i > 1, i = 1, 2$$

Few observations on properties

- Most papers focus on robustness property
- Network equivalence
- Invertibility of the network

Few observations on networks

Few observations on networks

- Most papers focus on classification problems
- Generative adversarial networks
- Reinforcement learning

Summary

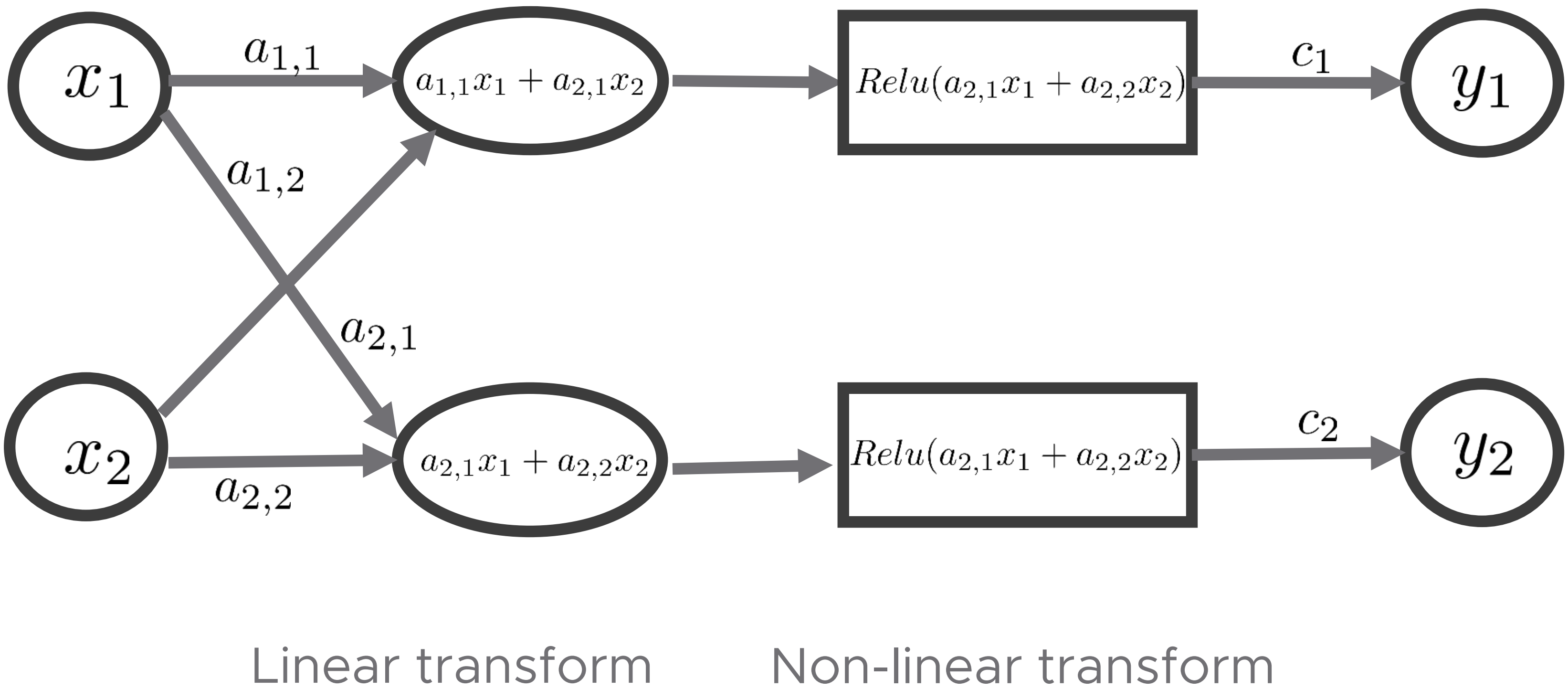
Summary

- Scalability remains the main issue
- We need to look beyond robustness

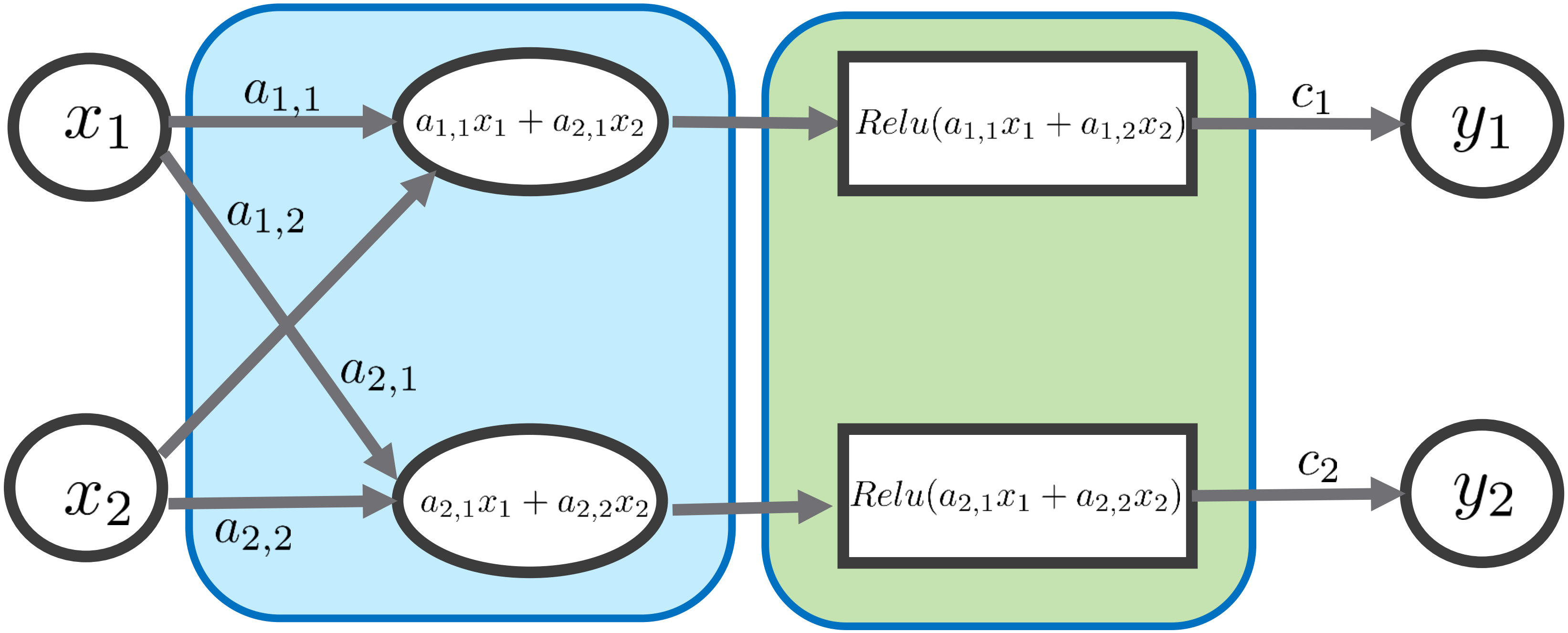
Verification of Neural Networks is an emerging exciting area!

Thanks!

High-level structure



High-level structure



Linear transform Non-linear transform

Network formula

$$y_1 = c_1 Relu(a_{1,1}x_1 + a_{1,2}x_2)$$
$$y_2 = c_2 Relu(a_{2,1}x_1 + a_{2,2}x_2)$$

Decision (robustness) problem

$$y_1 = c_1 Relu(a_{1,1}x_1 + a_{1,2}x_2)$$
$$y_2 = c_2 Relu(a_{2,1}x_1 + a_{2,2}x_2)$$

$$x_i \in [w_1, w_2], i = 1, 2$$
$$y_i > q, i = 1, 2$$

vmware®