Logic-based verification and explanation of NNs

Nina Narodytska,
VMware research

How to train your verifiable Binarized NNs?

Nina Narodytska,
VMware research

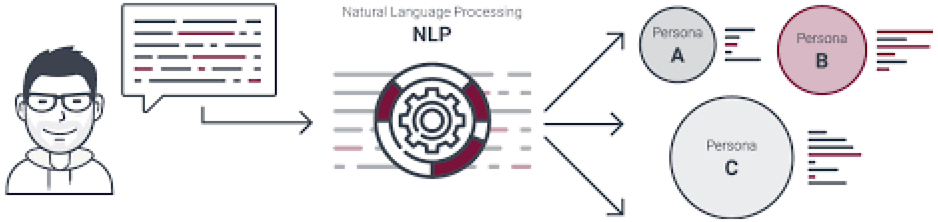
joint work with Hongce Zhang (summer @ VMware), Aarti Gupta

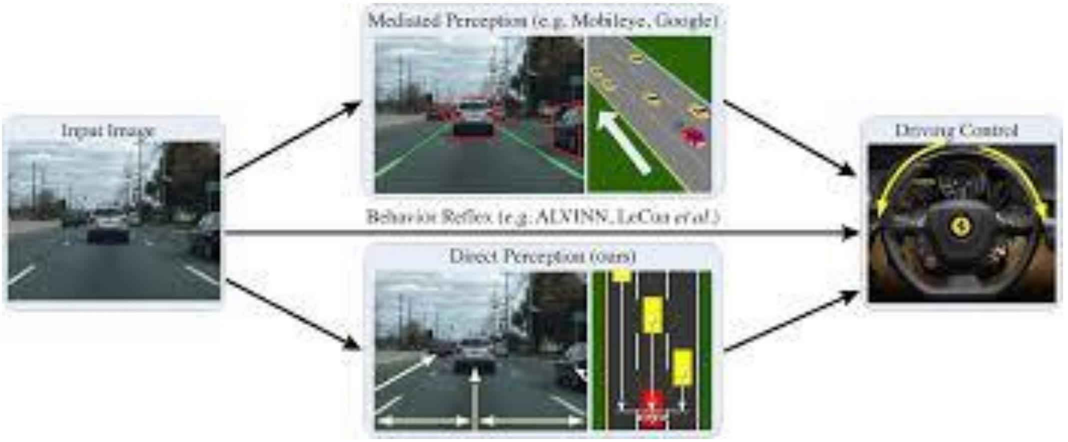
Outline

- Introduction
- Logic-based analysis of NNs
- Scalability of verification techniques
  - Analysis of bottlenecks
  - Experimental evaluation
- Conclusions

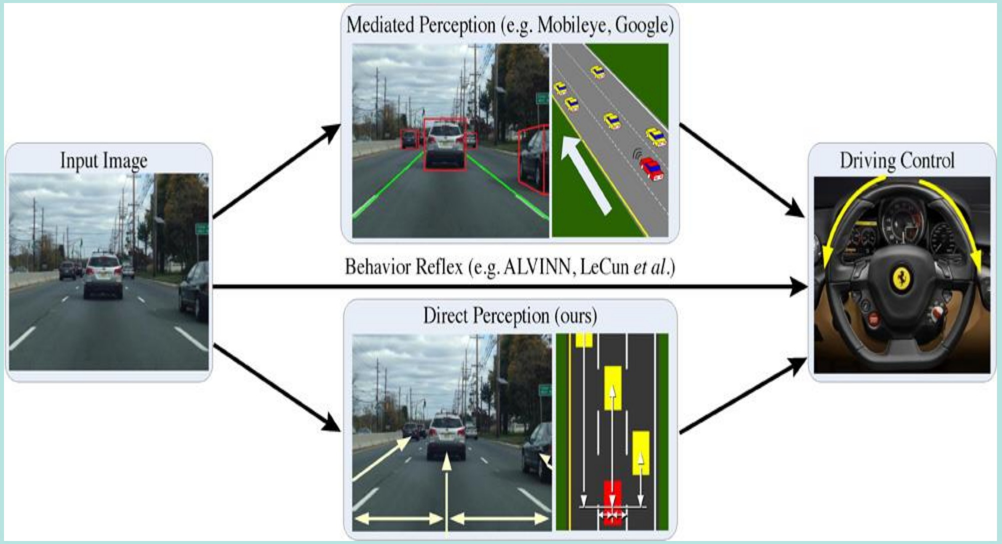
Introduction

ML models


Natural Language Processing
NLP
Persona
A
Persona
B
Persona
C

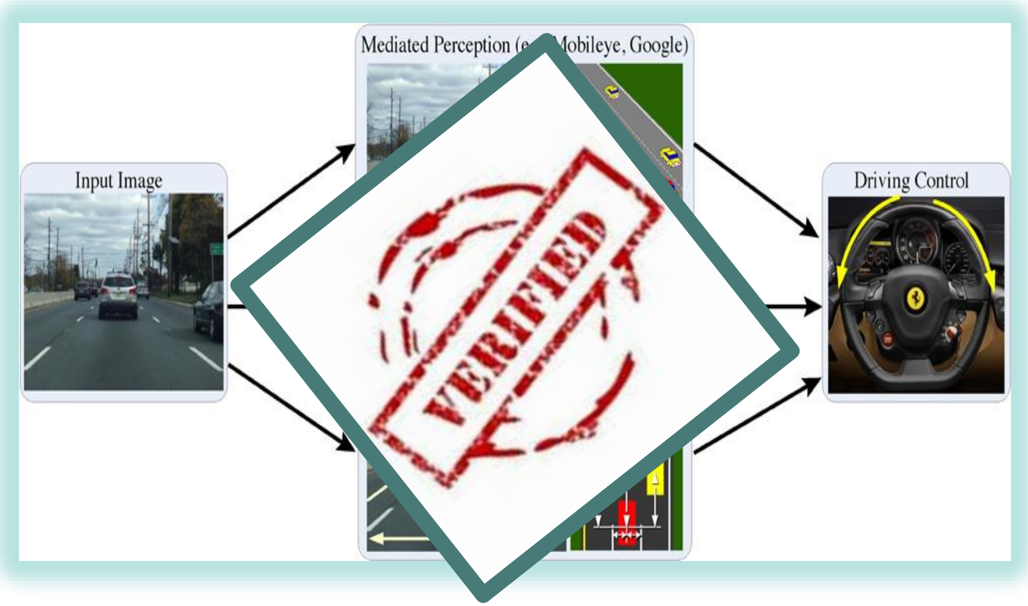

Mediated Perception (e.g. Mobileye, Google)
Input Image
Driving Control
Behavior Reflex (e.g. ALVINN, LeCun et al.)
Direct Perception (ours)

Verification of ML models



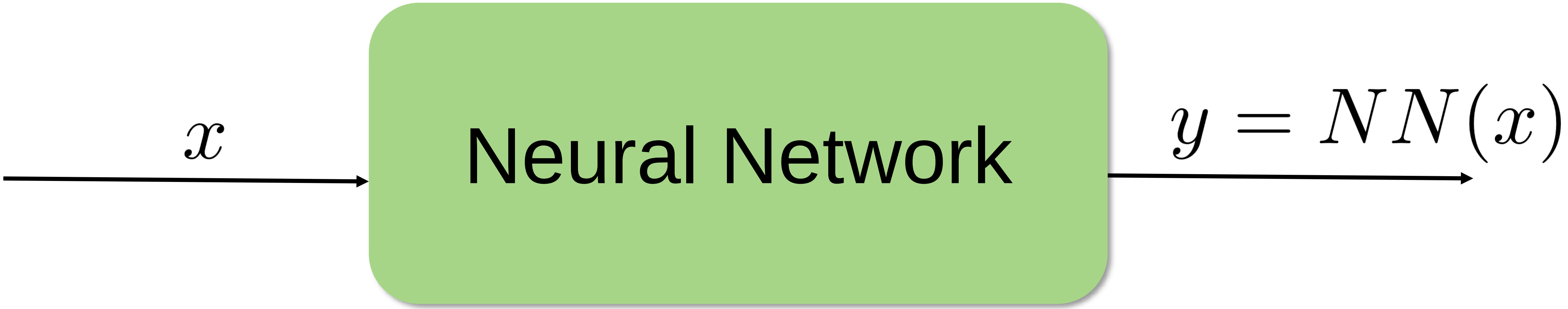
Verification of ML models

Verification of ML models

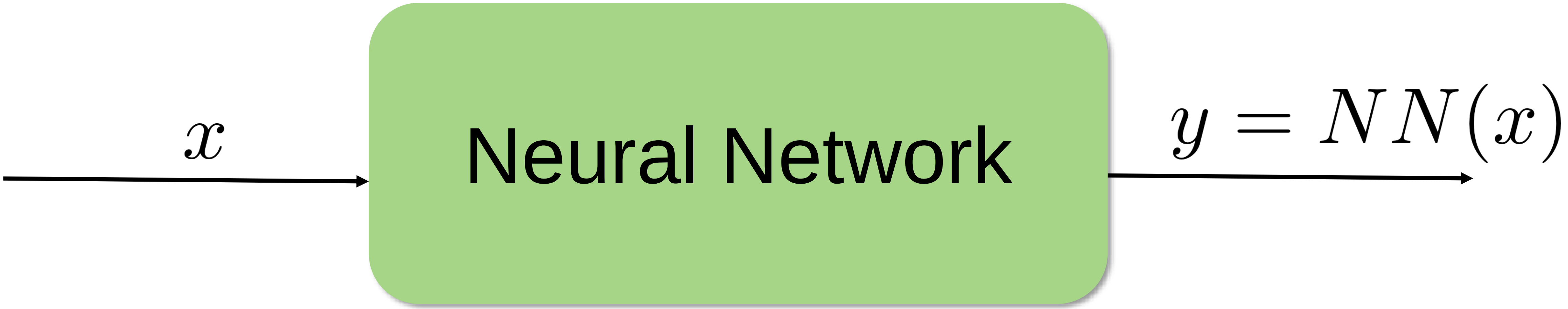


Verification of NNs

Verification of NNs

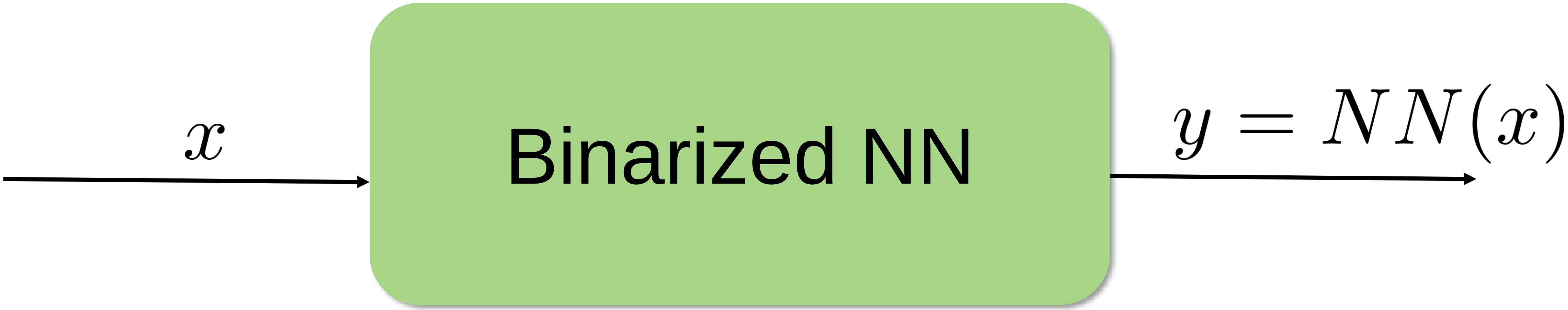


Verification of NNs



$$pre(x) \land y = NN(x) \Rightarrow post(y)$$

Verification of Binarized NNs



$$pre(x) \wedge y = BNN(x) \Rightarrow post(y)$$

Why are BNNs important?

Why BNNs?

Binarized neural networks: Training deep **neural networks** with weights and activations constrained to+ 1 or-1

M Courbariaux, I Hubara, D Soudry, R El-Yaniv… - arXiv preprint arXiv …, 2016 - arxiv.org

We introduce a method to train Binarized Neural Networks (BNNs)-neural networks with binary weights and activations at run-time. At training-time the binary weights and activations are used for computing the parameters gradients. During the forward pass, BNNs drastically …

☆ 99 Cited by 925 Related articles All 9 versions »

Binarized neural networks

I Hubara, M Courbariaux, D Soudry… - Advances in **neural** …, 2016 - papers.nips.cc

We introduce a method to train Binarized Neural Networks (BNNs)-neural networks with binary weights and activations at run-time. At train-time the binary weights and activations are used for computing the parameter gradients. During the forward pass, BNNs drastically …

☆ 99 Cited by 470 Related articles All 5 versions »

Xnor-net: Imagenet classification using binary convolutional **neural networks**

M Rastegari, V Ordonez, J Redmon… - European Conference on …, 2016 - Springer

… Because, at inference we only perform forward propagation with the **binarized** weights … Similar to **binarization** in the forward pass, we can **binarize** \(g^{in}\) in the backward pass … Our **binarization** technique is general, we can use any CNN architecture …

☆ 99 Cited by 1373 Related articles All 8 versions

Compactness

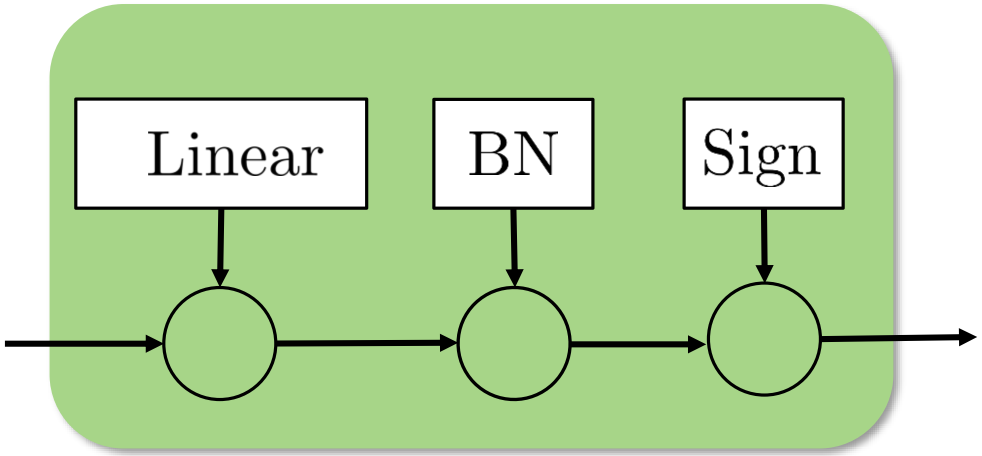
- Only 1 bit per weight, {-1,1}
- Can be deployed on embedded devices

Inference efficiency

- fast binary matrix multiplication
(7X speed up on GPU)
- “Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC” IEEE’2016

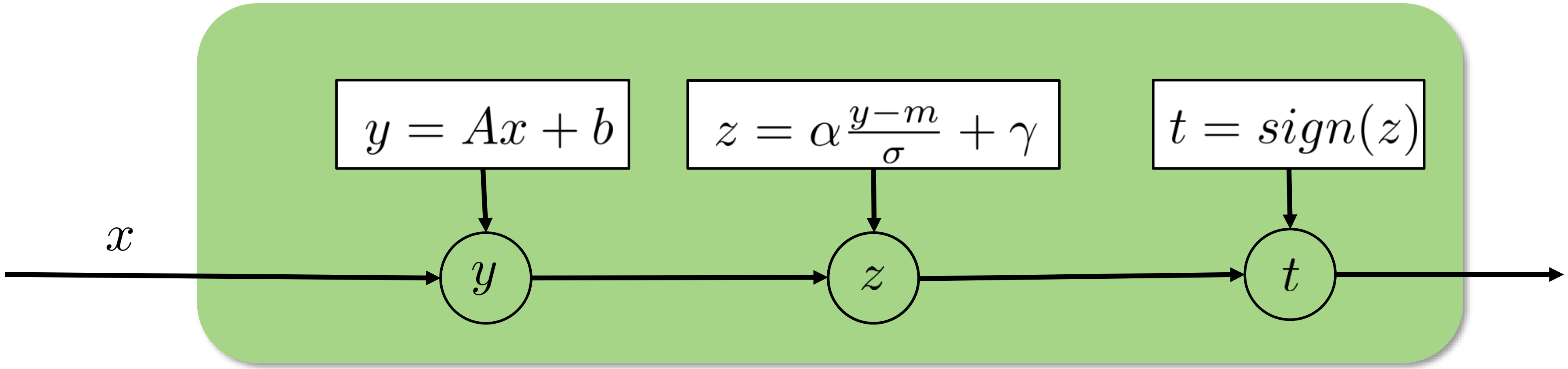
Structure of BNNs

Binarized Neural Networks

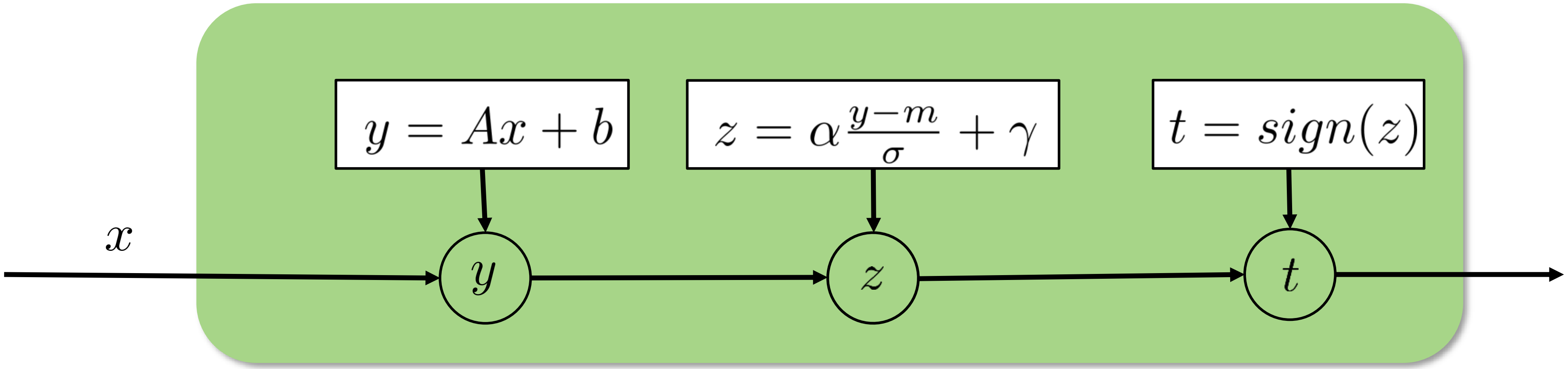


Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1
Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, Yoshua Bengio

Binarized Neural Networks

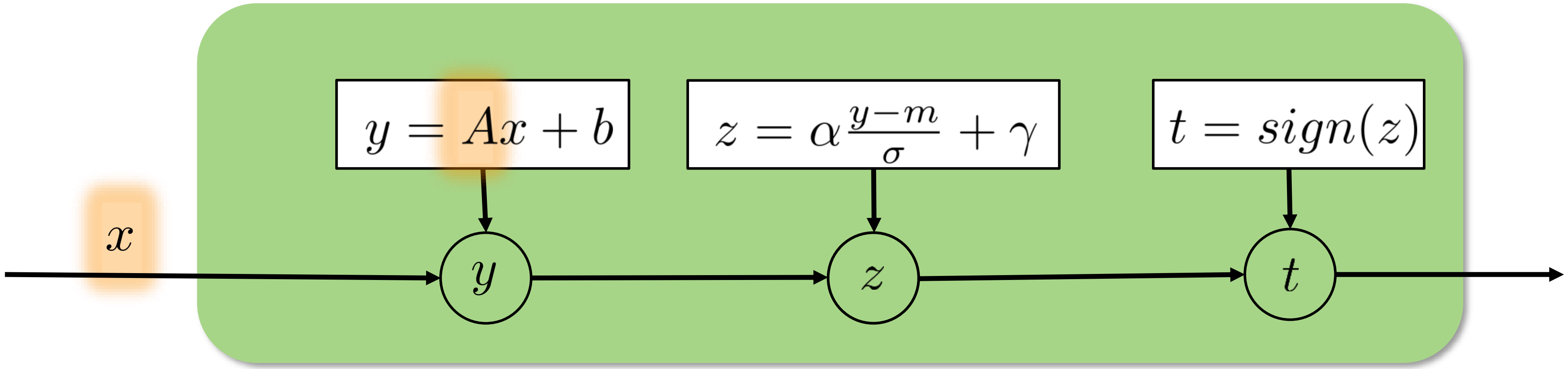


Binarized Neural Networks



$x, a_{i,j} \in \{-1, 1\}$ $\quad$ $b, \alpha, m, \sigma, \gamma \in \mathbf{R}$

Binarized Neural Networks



$x, a_{i,j} \in \{-1, 1\}$ $\quad$ $b, \alpha, m, \sigma, \gamma \in \mathbf{R}$

BNNs and Logic reasoning

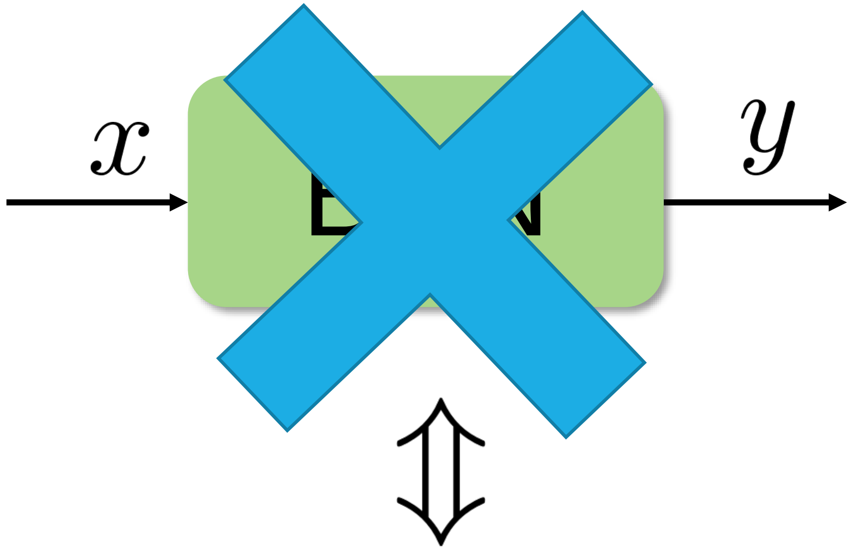
BNNs and Logic

BNNs and Logic

$$\Updownarrow$$

$$SAT(y = BNN(x))$$

BNNs and Logic



$$SAT(y = BNN(x))$$

BNNs and Logic

$$SAT(y = BNN(x))$$

BNNs and Logic

$$BinBNN(x, y) := SAT(y = BNN(x))$$

Logic-based analysis of BNNs

Verification

Explainability

Learning

Logic-based analysis of BNNs

- Properties verification using SAT solvers
- Quantitative reasoning using approximate methods
- Knowledge compilation, e.g. BDD, SDD
- Learning a network using optimization techniques

Verification

Nina Narodytska, Shiva Prasad Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh.
Verifying properties of binarized deep neural networks AAAI'18
Elias B. Khalil, Amrita Gupta, Bistra Dilkina:
Combinatorial Attacks on Binarized Neural Networks ICLR'19

Verification

$$pre(x) \wedge y = BNN(x) \Rightarrow post(y)$$

Verification

$$pre(x) \wedge BinBNN(x, y) \Rightarrow post(y)$$

Verification

$$(x_1 = 0) \wedge BinBNN(x, y) \Rightarrow (y_1 = 0)$$

Verification

$$(x_1 = 0) \wedge BinBNN(x, y) \wedge (y_1 \neq 0)$$

Verification

$$(x_1 = 0) \wedge BinBNN(x, y) \wedge (y_1 \neq 0)$$

$$\Downarrow$$

SAT solver

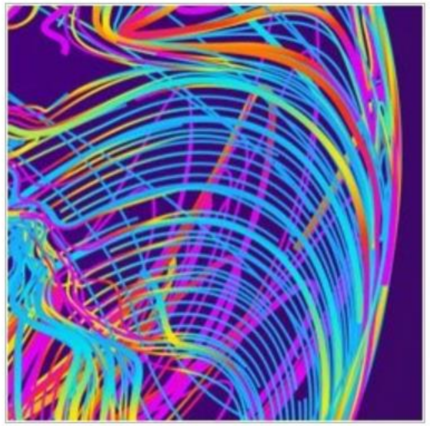
Explainability

Explainability

The Challenge of Crafting Intelligible Intelligence

By Daniel S. Weld, Gagan Bansal
Communications of the ACM, June 2019, Vol. 62 No. 6, Pages 70-79
10.1145/3282486
Comments (1)

VIEW AS: SHARE:

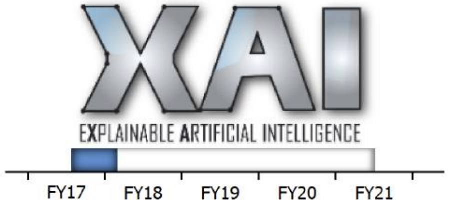


Artificial Intelligence (ai) systems have reached or exceeded human performance for many circumscribed tasks. As a result, they are increasingly deployed in mission-critical roles, such as credit scoring, predicting if a bail candidate will commit another crime, selecting the news we read on social networks, and self-driving cars. Unlike other mission-critical software, extraordinarily complex AI systems are difficult to test: AI decisions are context specific and often based on thousands or millions of factors. Typically, AI behaviors are generated by searching vast action spaces or learned by the opaque optimization of mammoth neural networks operating over prodigious amounts of training data. Almost by definition, no clear-cut method can accomplish these AI tasks.

European Union regulations on algorithmic decision-making and a "right to explanation"

Bryce Goodman,[1*] Seth Flaxman,[2]

Explainable Artificial Intelligence (XAI)



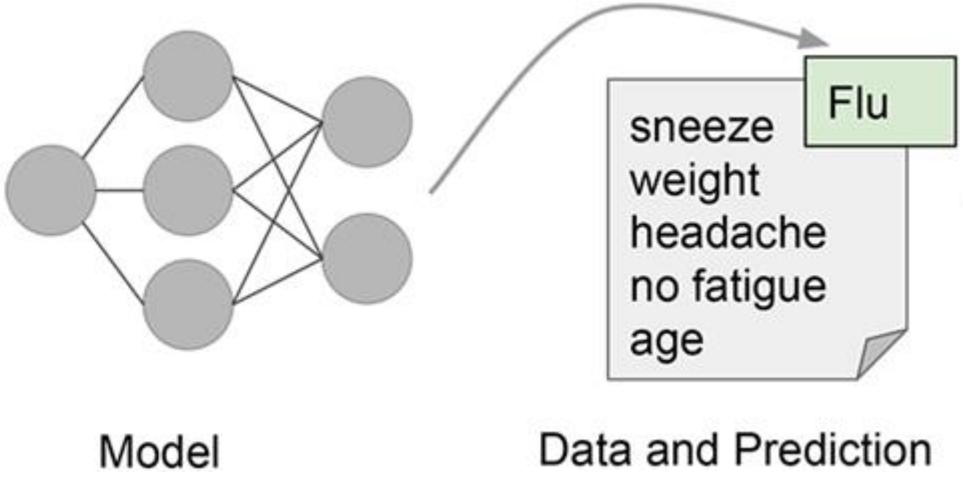
David Gunning
DARPA/I2O
Program Update November 2017

DARPA

Explainability

We summarize the potential impact that the European Union's new General Data Protection Regulation will have on the routine use of machine-learning algorithms. Slated to take effect as law across the European Union in 2018, it will place restrictions on automated individual decision making (that is, algorithms that make decisions based on user-level predictors) that "significantly affect" users. When put into practice, the law may also effectively create a right to explanation, whereby a user can ask for an explanation of an algorithmic decision that significantly affects them. We argue that while this law may pose large challenges for industry, it highlights opportunities for computer scientists to take the lead in designing algorithms and evaluation frameworks that avoid discrimination and enable explanation.

Explainability



Explainability

An **explanation** is a subset of input features so that changes to the rest of inputs do not affect the.

Alexey Ignatiev, Nina Narodytska, João Marques-Silva: AAAI'19**:** Abduction-Based Explanations for Machine Learning Models
Andy Shih and Arthur Choi and Adnan Darwiche : IJCAI'18: A Symbolic Approach to Explaining Bayesian Network Classifiers

Explainability

$$I = (x_1 = 0, x_2 = 0),\ y = 0$$

Explainability

$$I = (x_1 = 0, x_2 = 0),\ y = 0$$

$$I' \models (y = NN(x)) \rightarrow (y = 0), I' \subset I$$

Explainability

$$I = (x_1 = 0, x_2 = 0),\ y = 0$$

$$I' \models BinBNN(x, y) \rightarrow (y = 0), I' \subset I$$

Quantitative reasoning

Nina Narodytska, Aditya A. Shrotri, Kuldeep S. Meel, Alexey Ignatiev, João Marques-Silva:
Assessing Heuristic Machine Learning Explanations with Model Counting SAT'19.

Quantitative Verification of Neural Networks And its Security Applications
Teodora Baluta, Shiqi Shen, Shweta Shinde, Kuldeep S. Meel, Prateek Saxena

Quantitative reasoning

$$(x_1 = 0) \wedge BinBNN(x, y) \wedge (y_1 \neq 0)$$

Quantitative reasoning

$$(x_1 = 0) \wedge BinBNN(x, y) \wedge (y_1 \neq 0)$$

⇩

Model counting solver

Knowledge compilation

Verifying Binarized Neural Networks by Local Automaton Learning
Andy Shih and Adnan Darwiche and Arthur Choi

Knowledge compilation

$$BinBNN(x, y)$$

Verifying Binarized Neural Networks by Local Automaton Learning
Andy Shih and Adnan Darwiche and Arthur Choi

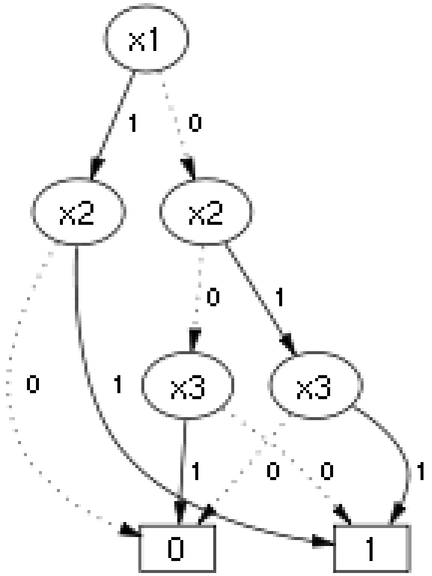
Knowledge compilation

$$BinBNN(x, y)$$ ⇨ CNf2BDD compiler

Verifying Binarized Neural Networks by Local Automaton Learning
Andy Shih and Adnan Darwiche and Arthur Choi

Knowledge compilation

$BinBNN(x, y)$ ⇨ CNf2BDD compiler ⇨



Verifying Binarized Neural Networks by Local Automaton Learning
Andy Shih and Adnan Darwiche and Arthur Choi

Logic-based analysis of BNNs

Verification

Explainability

Learning

Work with small networks

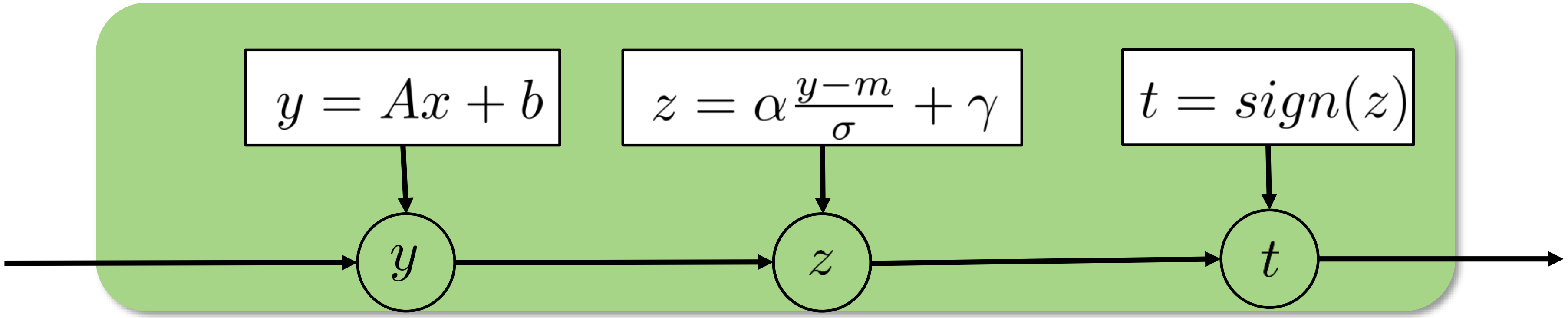
Work with small networks

- Properties verification using SAT solvers
  - < 200K (robustness with a very small epsilon)
- Quantitative reasoning using approximate methods
  - < 51K
- Knowledge compilation, e.g. BDD, SDD
  - < 10K

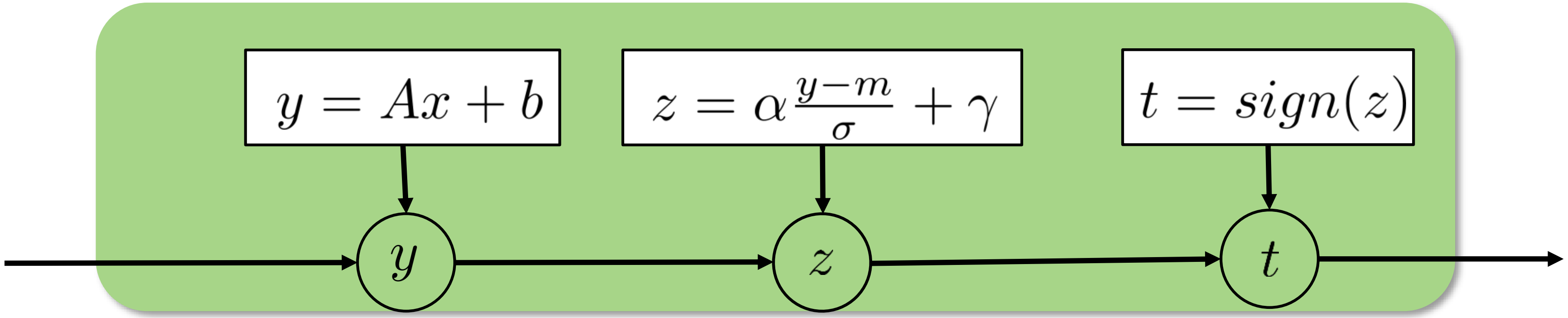
How can we improve scalability?

Translation: BNN to SAT

Translation: BNN to SAT



Translation: BNN to SAT



$$t_i = sign\left(\alpha\frac{(a_{i,1}x_1+\ldots+a_{i,n}x_n+b)-m}{\sigma} - \gamma\right)$$

Translation: BNN to SAT

$$t_i = sign\left(\alpha \frac{(a_{i,1}x_1 + \ldots + a_{i,n}x_n + b) - m}{\sigma} - \gamma\right)$$

Translation: BNN to SAT

$$\left(\alpha\frac{(a_{i,1}x_1+\ldots+a_{i,n}x_n+b)-m}{\sigma} - \gamma \geq 0\right) \Leftrightarrow t_i = 1$$

Translation: BNN to SAT

$$\left(\alpha\frac{(a_{i,1}x_1+\ldots+a_{i,n}x_n+b)-m}{\sigma}-\gamma\geq 0\right)\Leftrightarrow t_i=1$$

assuming $\alpha > 0$

Translation: BNN to SAT

$$\left(\alpha\frac{(a_{i,1}x_1+\ldots+a_{i,n}x_n+b)-m}{\sigma}-\gamma\geq 0\right)\Leftrightarrow t_i=1$$

⇕

$$\left(l_1+\ldots+l_n\geq\lceil\frac{\sigma(-\gamma)}{\alpha}+m-b+c\rceil\right)\Leftrightarrow t_i=1$$

assuming $\alpha > 0$

where
$a_{i,j}=1\Rightarrow l_j=x_j$,
$a_{i,j}=-1\Rightarrow l_j=\bar{x}_j$

Translation: BNN to SAT

$$\left(\alpha \frac{(a_{i,1}x_1+\ldots+a_{i,n}x_n+b)-m}{\sigma} - \gamma \geq 0\right) \Leftrightarrow t_i = 1$$

assuming $\alpha > 0$

$$\left(l_1 + \ldots + l_n \geq \left\lceil \frac{\sigma(-\gamma)}{\alpha} + m - b + c \right\rceil\right) \Leftrightarrow t_i = 1$$

where
$a_{i,j} = 1 \Rightarrow l_j = x_j$,
$a_{i,j} = -1 \Rightarrow l_j = \bar{x}_j$

Translation: BNN to SAT

$$\left(\alpha\frac{(a_{i,1}x_1+\ldots+a_{i,n}x_n+b)-m}{\sigma}-\gamma\geq 0\right)\Leftrightarrow t_i=1$$

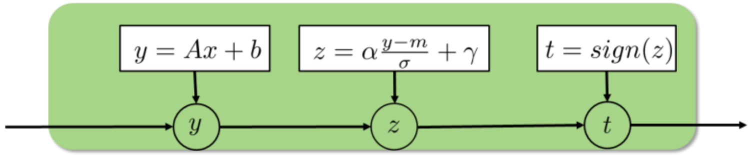
assuming $\alpha > 0$

$$\Updownarrow$$

$$(l_1+\ldots+l_n\geq k)\Leftrightarrow t_i=1$$

where
$a_{i,j}=1\Rightarrow l_j=x_j$,
$a_{i,j}=-1\Rightarrow l_j=\bar{x}_j$

Translation: BNN to SAT

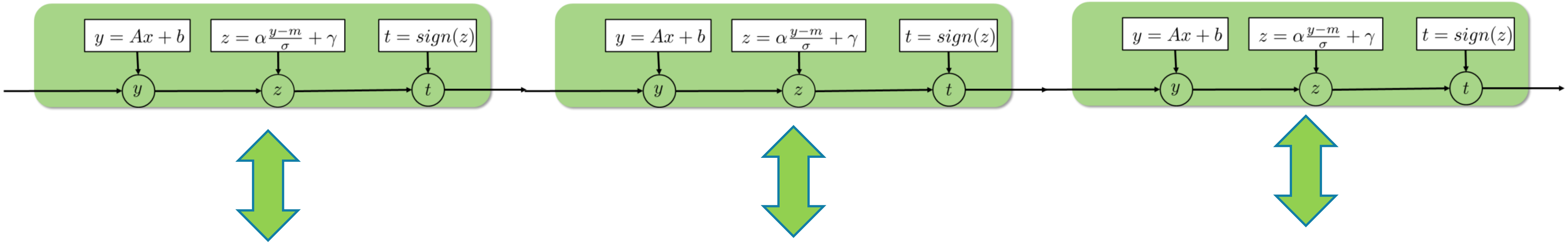


$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$

$$\vdots$$

$$(l_{m,1} + \ldots + l_{m,n} \geq k_m) \Leftrightarrow t_m = 1$$

Translation: BNN to SAT



$$\begin{matrix}(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1^1 = 1 \\ \vdots \\ (l_{m,1} + \ldots + l_{m,n} \geq k_m) \Leftrightarrow t_m^1 = 1\end{matrix} \;\wedge\; \begin{matrix}(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1 \\ \vdots \\ (l_{m,1} + \ldots + l_{m,n} \geq k_m) \Leftrightarrow t_m = 1\end{matrix} \;\wedge\; \begin{matrix}(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1^p = 1 \\ \vdots \\ (l_{m,1} + \ldots + l_{m,n} \geq k_m) \Leftrightarrow t_m^p = 1\end{matrix}$$

BinBNN

Why are BinBNNs hard to solve?

Birds view: a large formula

$$\begin{array}{c}(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1 \\ \vdots \\ (l_{m,1} + \ldots + l_{m,n} \geq k_m) \Leftrightarrow t_m = 1\end{array} \wedge \begin{array}{c}(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1 \\ \vdots \\ (l_{m,1} + \ldots + l_{m,n} \geq k_m) \Leftrightarrow t_m = 1\end{array} \wedge \begin{array}{c}(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1 \\ \vdots \\ (l_{m,1} + \ldots + l_{m,n} \geq k_m) \Leftrightarrow t_m = 1\end{array} \ldots$$

Birds view: a large formula

$$\begin{pmatrix}(l_{1,1}+\ldots+l_{1,n}\geq k_1)\Leftrightarrow t_1=1\\ \vdots\\ (l_{m,1}+\ldots+l_{m,n}\geq k_m)\Leftrightarrow t_m=1\end{pmatrix}\wedge\begin{pmatrix}(l_{1,1}+\ldots+l_{1,n}\geq k_1)\Leftrightarrow t_1=1\\ \vdots\\ (l_{m,1}+\ldots+l_{m,n}\geq k_m)\Leftrightarrow t_m=1\end{pmatrix}\wedge\begin{pmatrix}(l_{1,1}+\ldots+l_{1,n}\geq k_1)\Leftrightarrow t_1=1\\ \vdots\\ (l_{m,1}+\ldots+l_{m,n}\geq k_m)\Leftrightarrow t_m=1\end{pmatrix}\ldots$$

+ a structure aware solver

Birds view: a large formula

$$\begin{pmatrix}(l_{1,1}+\ldots+l_{1,n}\geq k_1)\Leftrightarrow t_1=1\\ \vdots \\ (l_{m,1}+\ldots+l_{m,n}\geq k_m)\Leftrightarrow t_m=1\end{pmatrix} \wedge \begin{pmatrix}(l_{1,1}+\ldots+l_{1,n}\geq k_1)\Leftrightarrow t_1=1\\ \vdots \\ (l_{m,1}+\ldots+l_{m,n}\geq k_m)\Leftrightarrow t_m=1\end{pmatrix} \wedge \begin{pmatrix}(l_{1,1}+\ldots+l_{1,n}\geq k_1)\Leftrightarrow t_1=1\\ \vdots \\ (l_{m,1}+\ldots+l_{m,n}\geq k_m)\Leftrightarrow t_m=1\end{pmatrix} \ldots$$

+ a structure aware solver

Macroview: a large formula for a block

. . .

Macroview: a large block

$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$

$$(l_{2,1} + \ldots + l_{2,n} \geq k_2) \Leftrightarrow t_2 = 1$$

$$(l_{3,1} + \ldots + l_{3,n} \geq k_3) \Leftrightarrow t_3 = 1$$

$\ldots$

Macroview: a large block

$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$
$$(l_{2,1} + \ldots + l_{2,n} \geq k_2) \Leftrightarrow t_2 = 1$$
$$(l_{3,1} + \ldots + l_{3,n} \geq k_3) \Leftrightarrow t_3 = 1$$

$\ldots$

+ a nice shape of a matrix

Macroview: a large block

$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$

$$(l_{2,1} + \ldots + l_{2,n} \geq k_2) \Leftrightarrow t_2 = 1$$

$$(l_{3,1} + \ldots + l_{3,n} \geq k_3) \Leftrightarrow t_3 = 1$$

$\ldots$

+ a nice shape of a matrix

Microview: a large constraint

$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$

...

Microview: a large constraint

$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$

Number of variables

Reification means no propagation!

. . .

Microview: a large constraint

$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$

...

Number of variables

Reification means no propagation!

+ reduce #vars

+ eliminate reifications

Wish list

+ reduce #vars

+ eliminate reifications

+ a nice shape of a matrix

+ a structure aware solver

Wish list

+ reduce #vars

+ eliminate reifications

+ a nice shape of a matrix

+ a structure aware solver

Wish list

+ reduce #vars

+ eliminate reifications

+ a nice shape of a matrix

+ a structure aware solver

BNN parameters and structure are not fixed*

We can train a BNN so that

+ reduce #vars

+ eliminate reifications

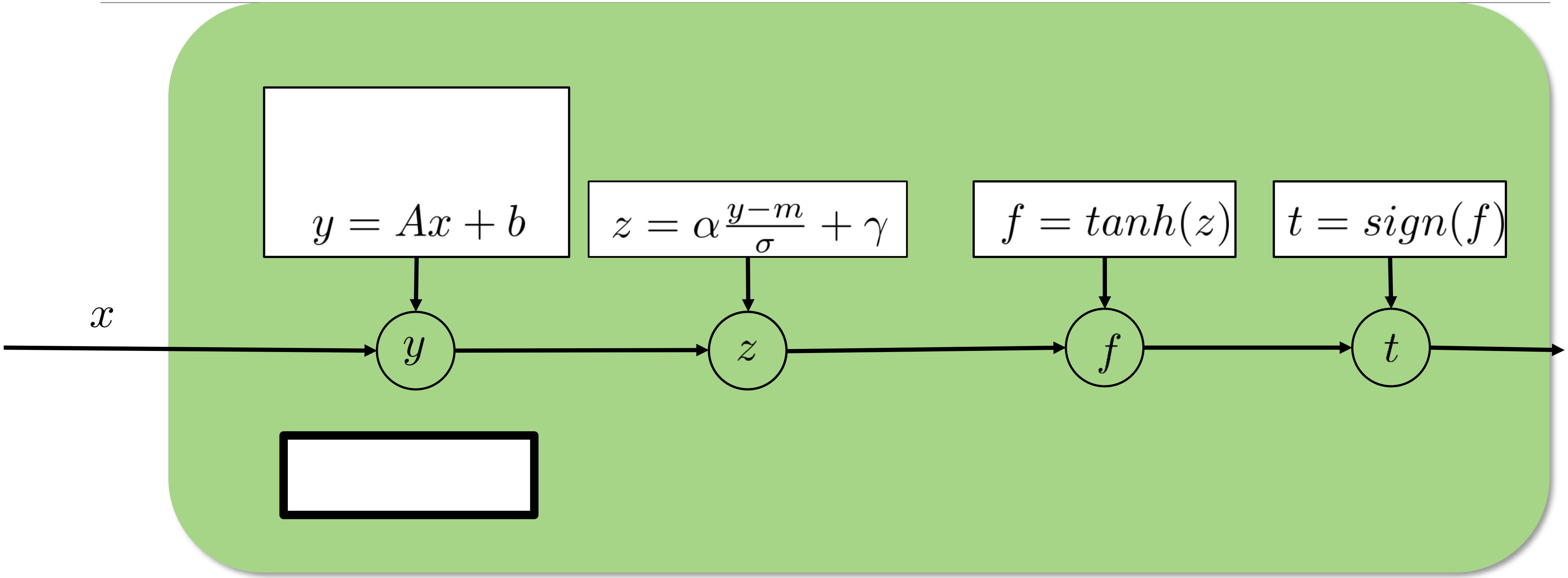
We can train a BNN so that

+ reduce #vars

+ eliminate reifications

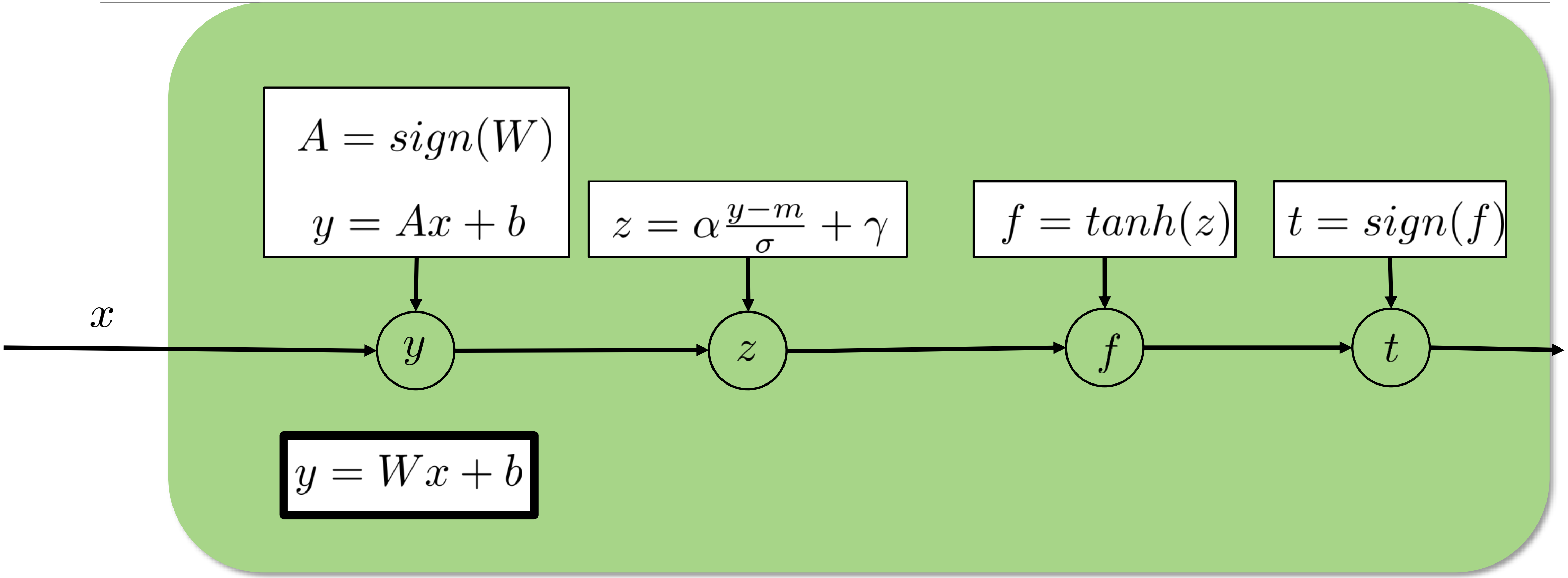
Training for Faster Adversarial Robustness Verification via Inducing ReLU Stability
Kai Y. Xiao, Vincent Tjeng, Nur Muhammad (Mahi) Shafiullah, Aleksander Madry

Binarized Neural Networks



$x, a_{i,j} \in \{-1, 1\}$ $b, \alpha, m, \sigma, \gamma, W \in \mathbf{R}$

Binarized Neural Networks



$x, a_{i,j} \in \{-1, 1\}$ $\quad$ $b, \alpha, m, \sigma, \gamma, W \in \mathbf{R}$

Running example

Dataset: MNIST with background

Problem: Untargeted adversarial examples with ε in {*1,3,5*,10,15,25}

Networks: BNNs with five FC layers

- "Small BNN" with 200K params
- **"Large BNN" with 620K** params

Running example

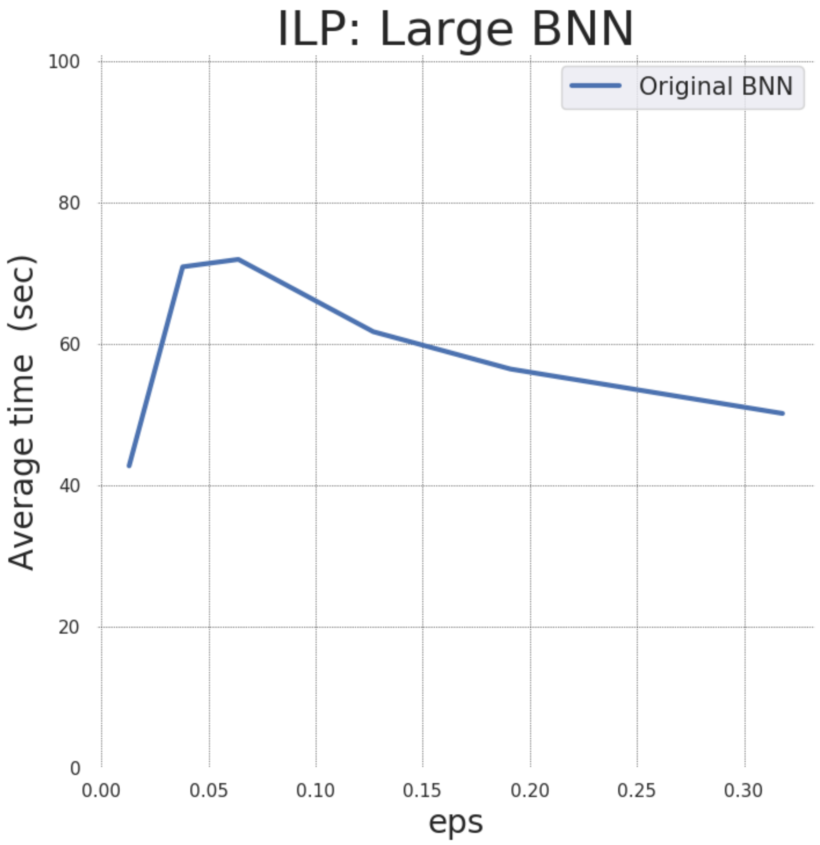
Train: From a pretrain full precision network

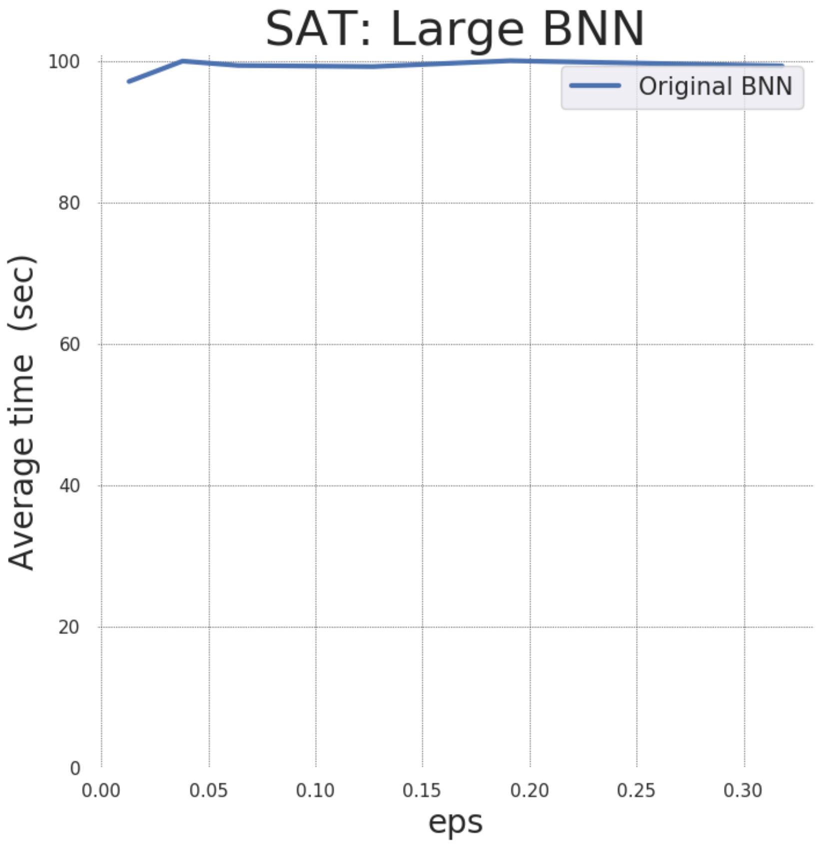
Inputs: Normalized

Results: average time to solve per ε out of 100 benchmarks

Solvers: CPLEX, Glucose (PySAT convertor)

Baseline: verification of original BNNs





Ternary quantization

Ternary quantization

BNN+: Improved Binary Network Training

Sajad Darabi, Mouloud Belbahri, Matthieu Courbariaux, Vahid Partovi Nia

Ternary quantization

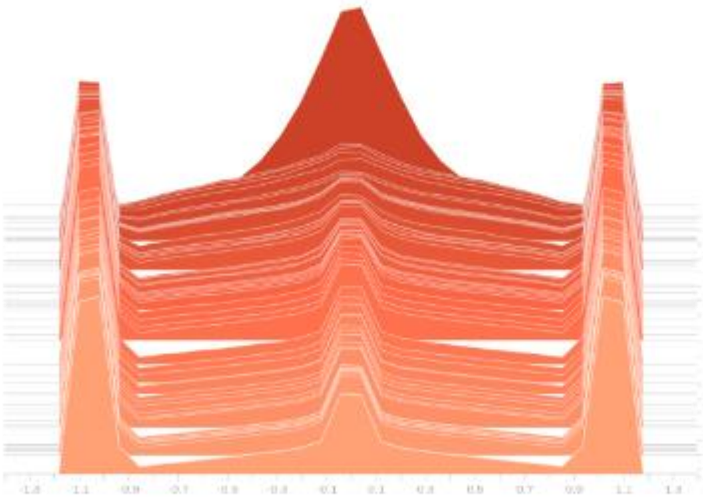


Figure 2: Progression of the weights training in BNN (Hubara et al., 2016) . As training progresses the weights create three modes: at -1, 0, and at $+1$.

BNN+: Improved Binary Network Training

Sajad Darabi, Mouloud Belbahri, Matthieu Courbariaux, Vahid Partovi Nia

Ternary quantization

$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$

where

$$a_{i,j} = 1 \Rightarrow l_j = x_j,$$
$$a_{i,j} = -1 \Rightarrow l_j = \bar{x}_j$$

Ternary quantization

$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$

where

$$a_{i,j} = 1 \Rightarrow l_j = x_j,$$
$$a_{i,j} = 0 \Rightarrow l_j = 0,$$
$$a_{i,j} = -1 \Rightarrow l_j = \bar{x}_j$$

Ternary quantization

Train ternary NN where weights are -1,0,1

Trained Ternary Quantization, ICLR’16, Zhu at el

Ternary quantization

1. Train a BNN
2. Build a distribution of absolute values of weights
3. Select a percentile (40%, 60%), t= 0.03
4. Train a ternary BNN with the two-sided threshold t

$$a_{i,j} = \begin{cases} 0 & \text{if } |w_{i,j}| \leq t \\ sign(w_{i,j}) & \text{otherwise} \end{cases}$$

Ternary quantization

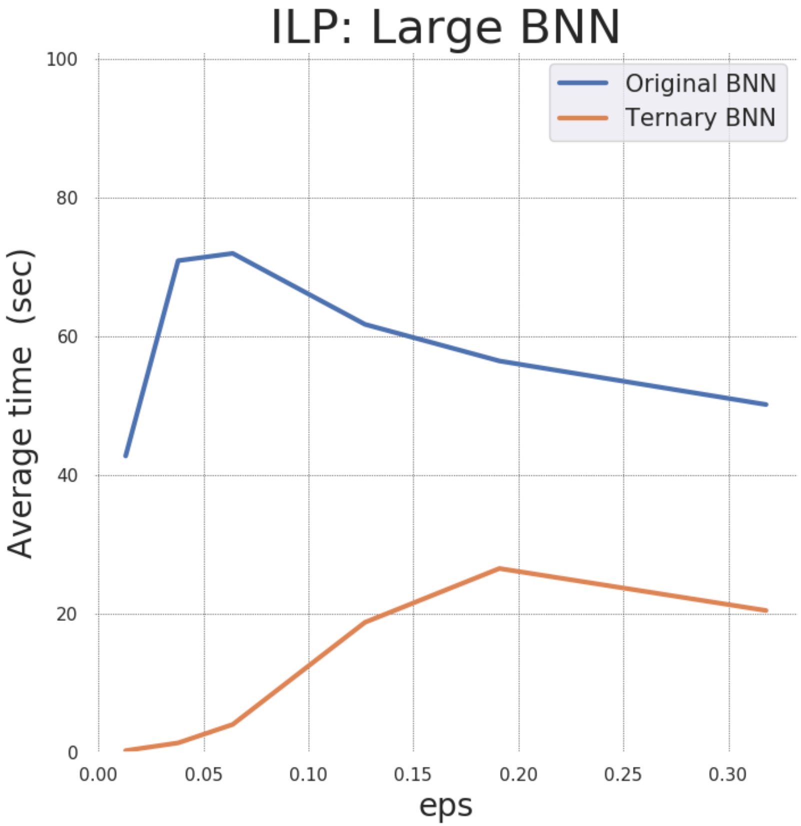
1. Train a BNN
2. Build a distribution of absolute values of weights
3. Select a percentile (40%, 60%), t= 0.03
4. Train a ternary BNN with the two-sided threshold t

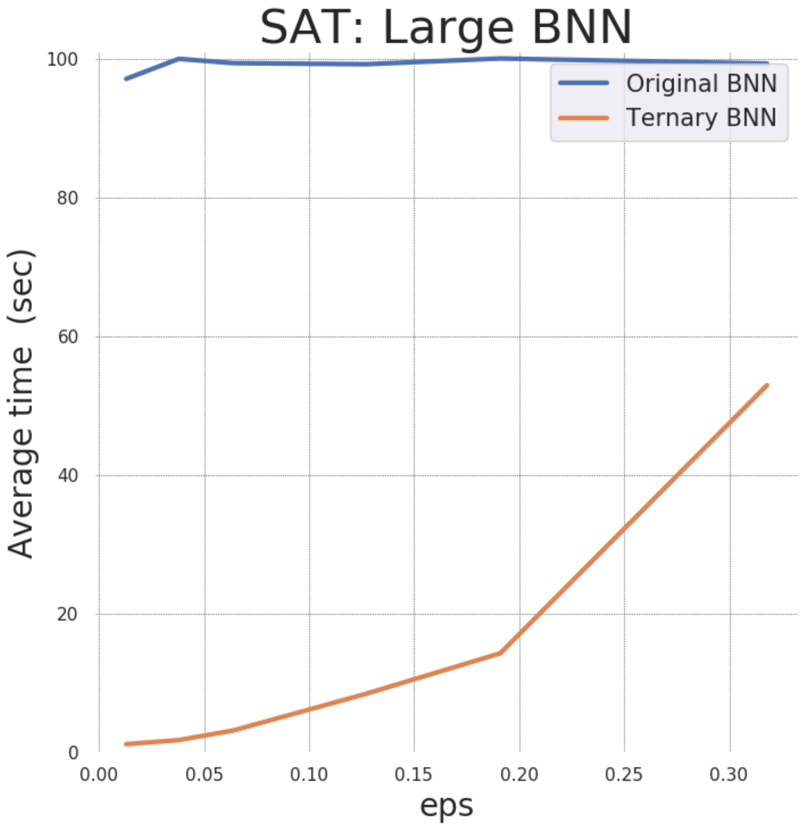
Note: Transformation from BNN to SAT changes a bit

Ternary quantization

	Small BNN	Large BNN
Original BNN	200K (73.0%)	600K (74.0%)
Ternary BNN	26K (75.2%)	40K (78.2%)

Ternary quantization





L1+Ternary quantization

L1+Ternary quantization

$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$

where

$$a_{i,j} = 1 \Rightarrow l_j = x_j,$$
$$a_{i,j} = 0 \Rightarrow l_j = 0,$$
$$a_{i,j} = -1 \Rightarrow l_j = \bar{x}_j$$

L1+Ternary quantization

$$(l_{1,1} + \ldots + l_{1,n} \geq k_1) \Leftrightarrow t_1 = 1$$

where

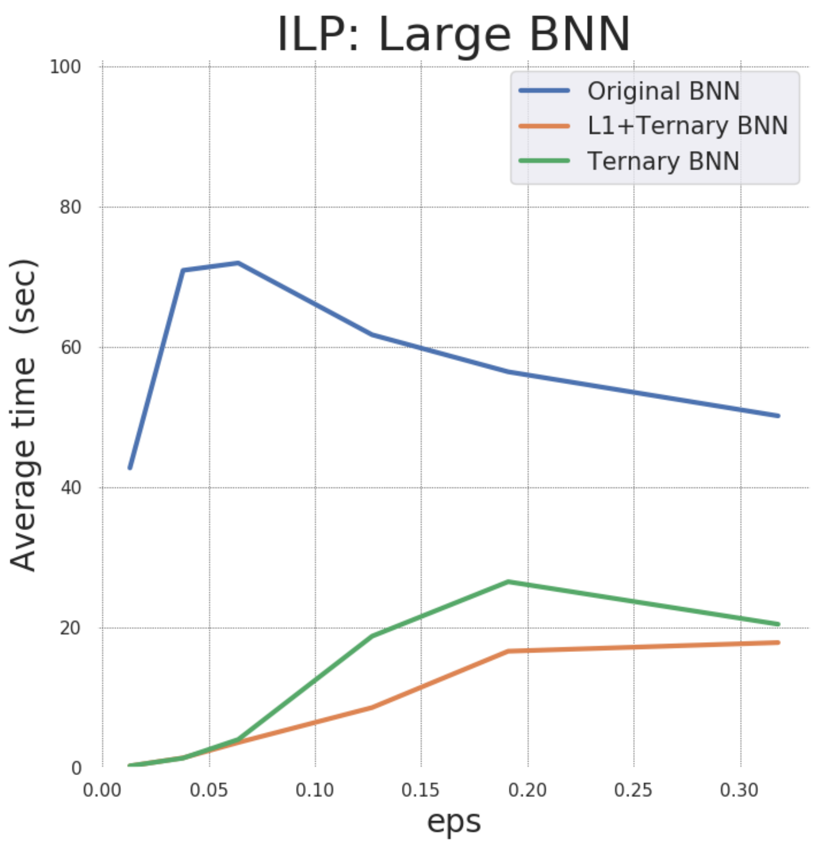
$$a_{i,j} = 1 \Rightarrow l_j = x_j,$$
$$a_{i,j} = 0 \Rightarrow l_j = 0,$$
$$a_{i,j} = -1 \Rightarrow l_j = \bar{x}_j$$

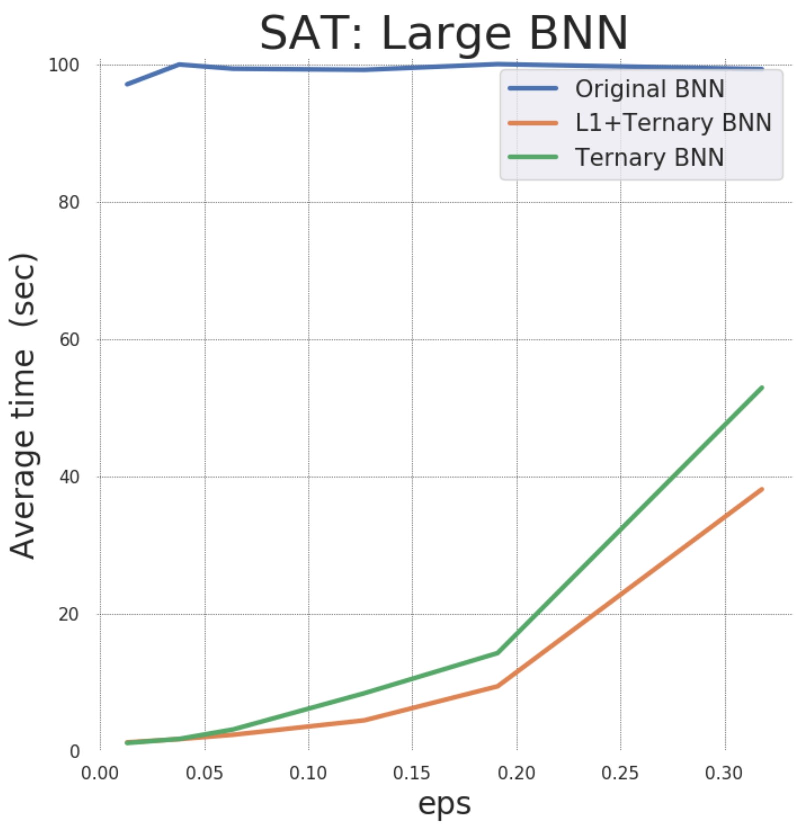
Add L1 regularization

L1+Ternary quantization

	Small BNN	Large BNN
Original BNN	200K (73.0%)	600K (74.0%)
Ternary BNN	26K (75.2%)	40K (78.2%)
L1 + Ternary BNN	24K (75.3%)	36K (78.4%)

L1 + Ternary BNN





Stabilization of SIGN

Stabilization of SIGN

$$(l_{1,1} + \ldots + l_{1,n} - k_1 \geq 0) \Leftrightarrow t_1 = 1$$

Stabilization of SIGN

$$(l_{1,1} + \ldots + l_{1,n} - k_1 \geq 0) \Leftrightarrow t_1 = 1$$

$$LB_{(l_{1,1}+\ldots+l_{1,n}-k_1)} \geq 0$$

Stabilization of SIGN

$$(l_{1,1} + \ldots + l_{1,n} - k_1 \geq 0) \Leftrightarrow t_1 = 1$$

$$LB_{(l_{1,1}+\ldots+l_{1,n}-k_1)} \geq 0 \qquad t_1 = 1$$

Stabilization of SIGN

$$(l_{1,1} + \ldots + l_{1,n} - k_1 \geq 0) \Leftrightarrow t_1 = 1$$

Stabilization of SIGN

$$(l_{1,1} + \ldots + l_{1,n} - k_1 \geq 0) \Leftrightarrow t_1 = 1$$

$$UB_{(l_{1,1}+\ldots+l_{1,n}-k_1)} < 0$$

Stabilization of SIGN

$$(l_{1,1} + \ldots + l_{1,n} - k_1 \geq 0) \Leftrightarrow t_1 = 1$$

$$UB_{(l_{1,1}+\ldots+l_{1,n}-k_1)} < 0 \qquad t_1 = 0$$

Stabilization of SIGN

Encourage LB and UB of a neurons to take the same sign:

$$sign(UB_{i,j}) = sign(LB_{i,j})$$

Stabilization of SIGN

We add a term to the loss function:

$$sign(UB_{i,j}) * sign(LB_{i,j})$$

Stabilization of SIGN

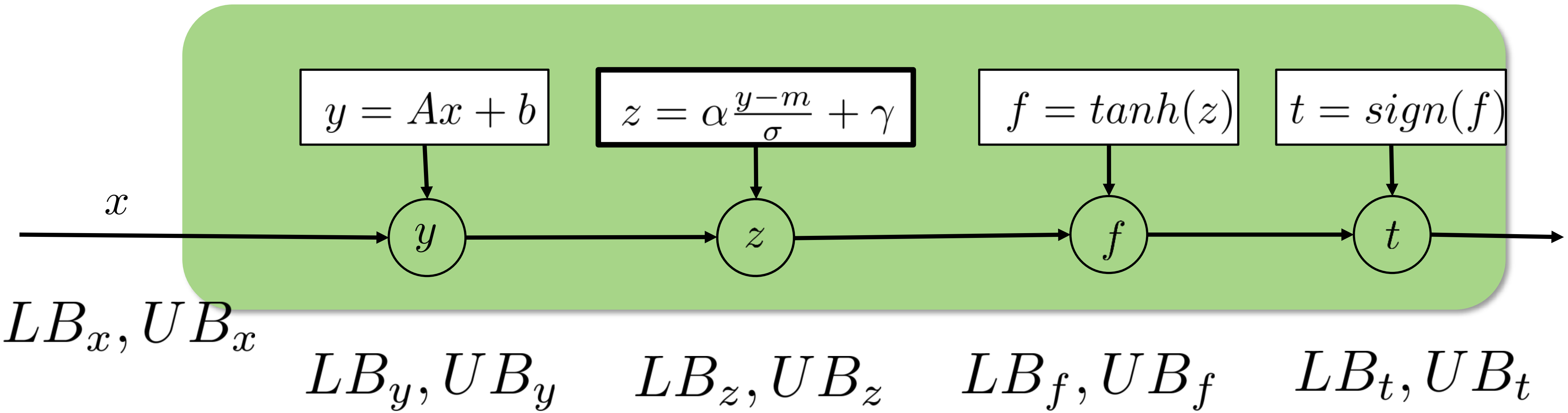
We add a (approximation) term to the loss function:

$$\sout{sign(UB_{i,j}) * sign(LB_{i,j})}$$

$$-tanh(1 + UB_{ij}LB_{ij})$$

Training for Faster Adversarial Robustness Verification via Inducing ReLU Stability
Kai Y. Xiao, Vincent Tjeng, Nur Muhammad (Mahi) Shafiullah, Aleksander Madry

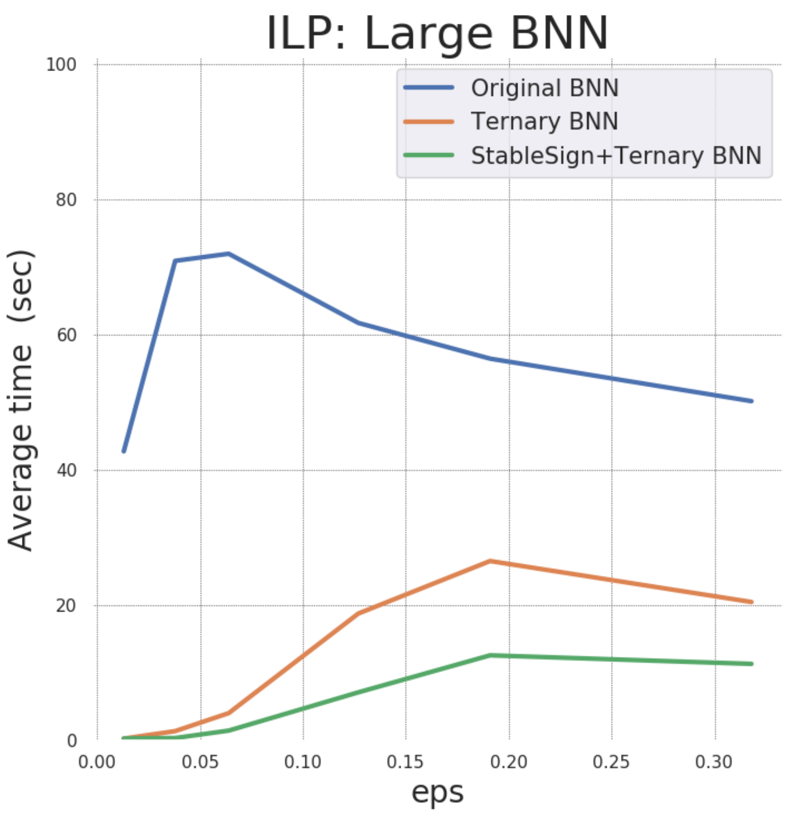
Stabilization of SIGN

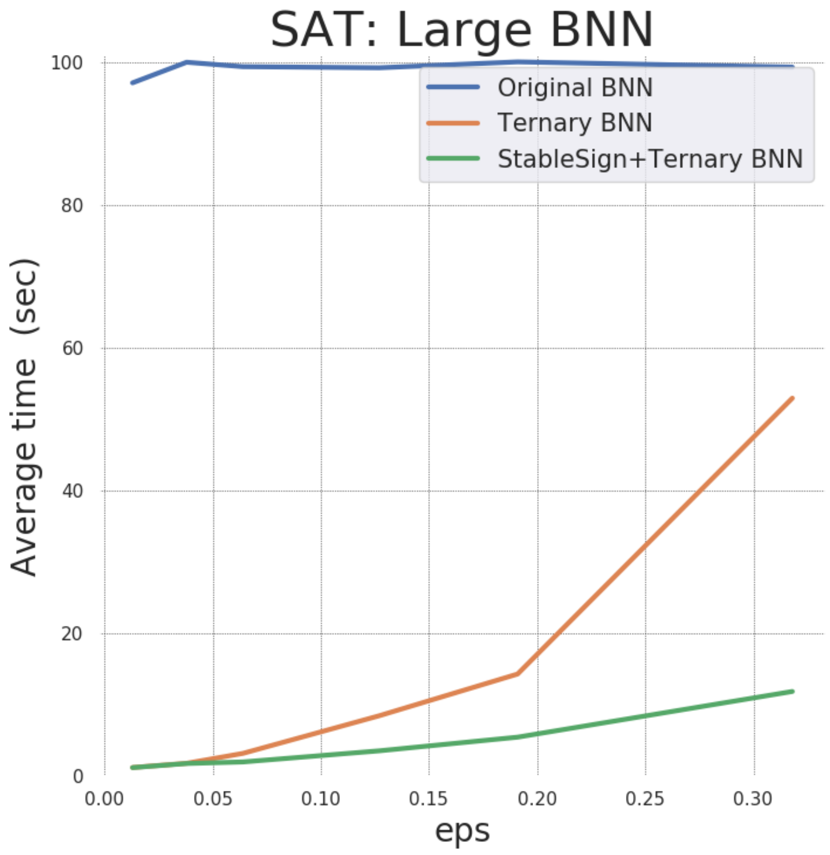


StableSign+Ternary quantization

	Small BNN	Large BNN
Original BNN	200K (73.0%)	600K (74.0%)
Ternary BNN	26K (75.2%)	40K (78.2%)
StableSign + Ternary BNN	25K (76.7%) ~20% stable	38K (78.4%) ~40% stable

StableSign+Ternary quantization

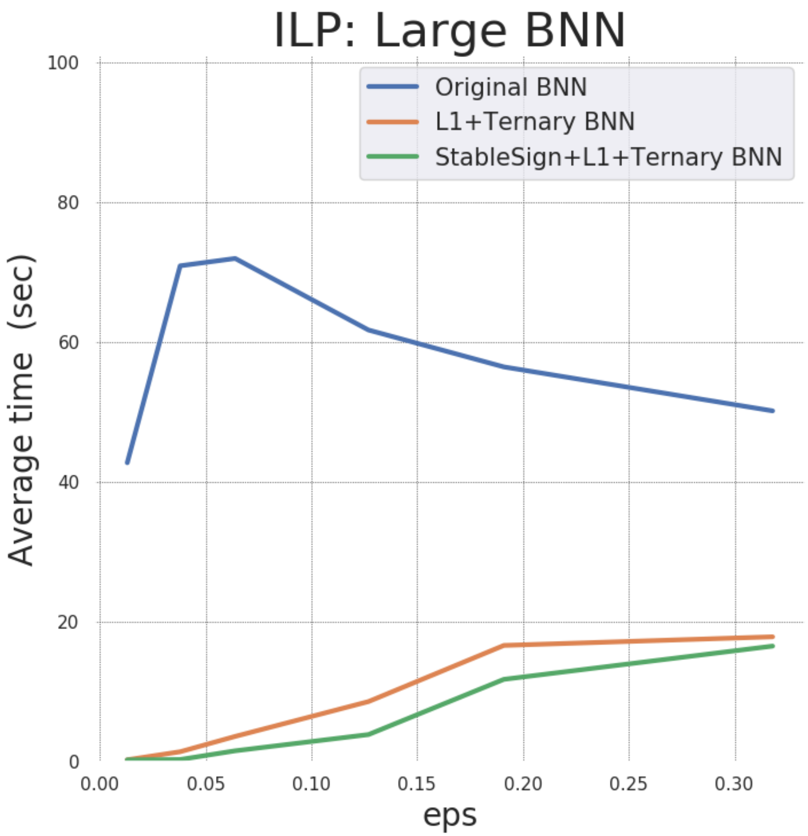


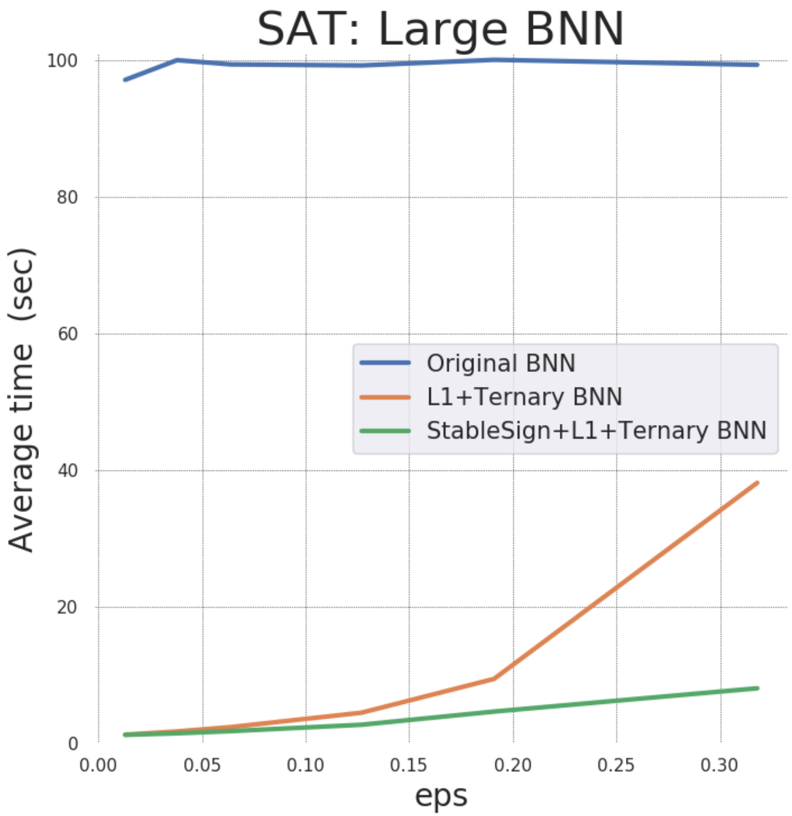


StableSign+ L1+Ternary quantization

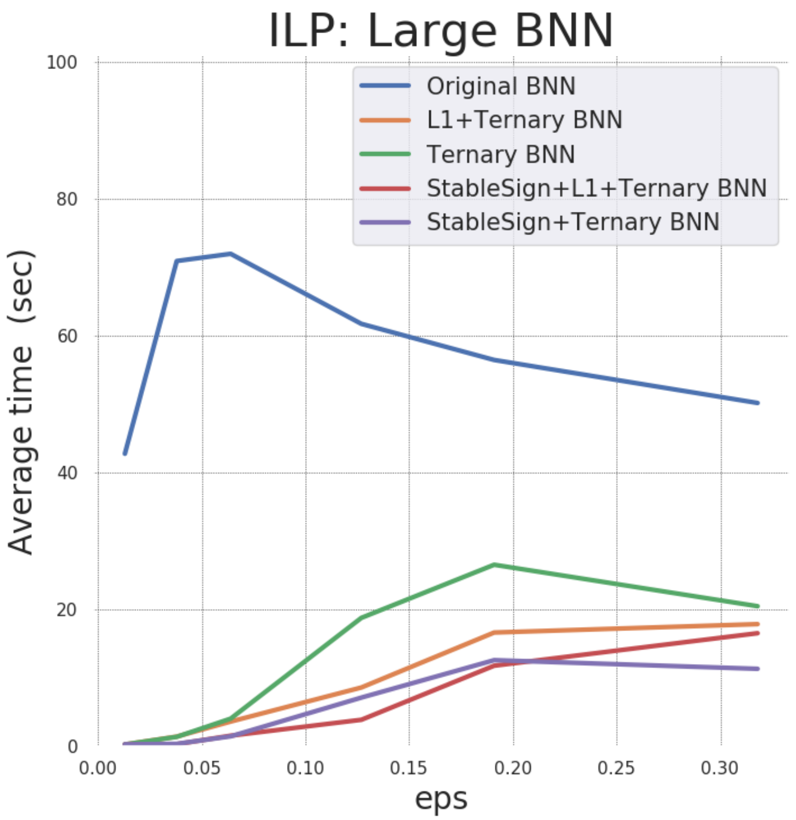
	Small BNN	Large BNN
Original BNN	200K (73.0%)	600K (74.0%)
L1 + Ternary BNN	24K (75.3%)	36K (78.4%)
StableSign + L1 + Ternary BNN	23K (76.6%) ~20% stable	34K (80.4%) 40% stable

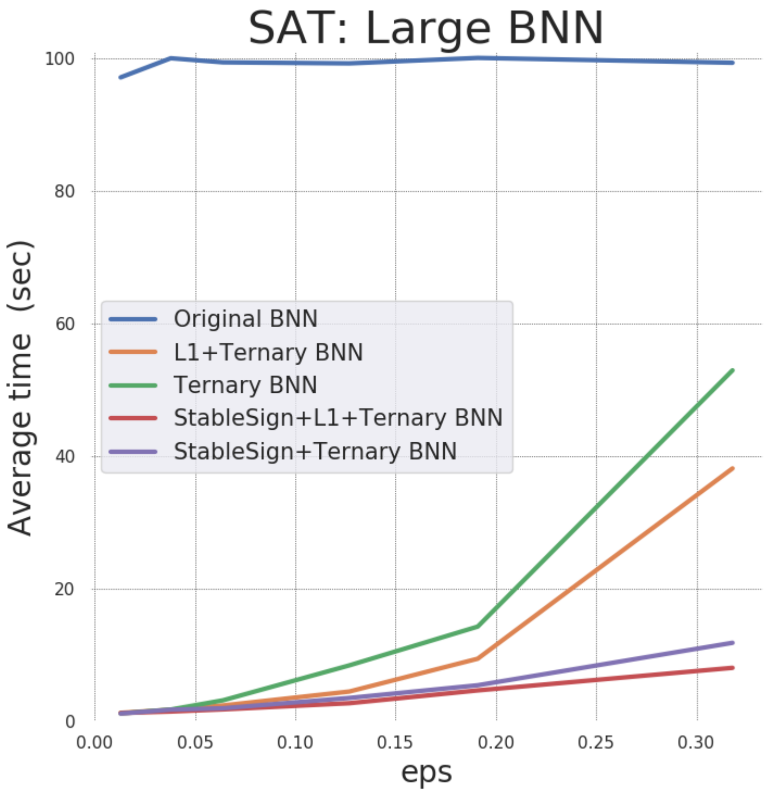
StableSign+ L1+Ternary quantization





Summary


ILP: Large BNN
Original BNN
L1+Ternary BNN
Ternary BNN
StableSign+L1+Ternary BNN
StableSign+Ternary BNN
Average time (sec)
eps
0
20
40
60
80
100
0.00
0.05
0.10
0.15
0.20
0.25
0.30


SAT: Large BNN
Original BNN
L1+Ternary BNN
Ternary BNN
StableSign+L1+Ternary BNN
StableSign+Ternary BNN
Average time (sec)
eps
0
20
40
60
80
100
0.00
0.05
0.10
0.15
0.20
0.25
0.30

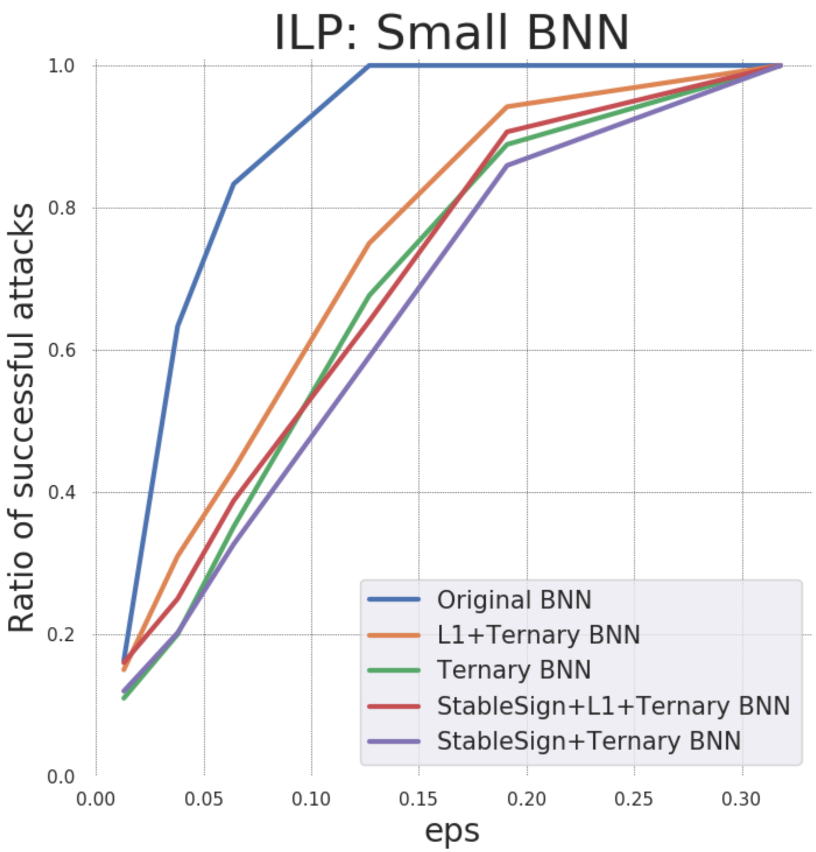
Running example

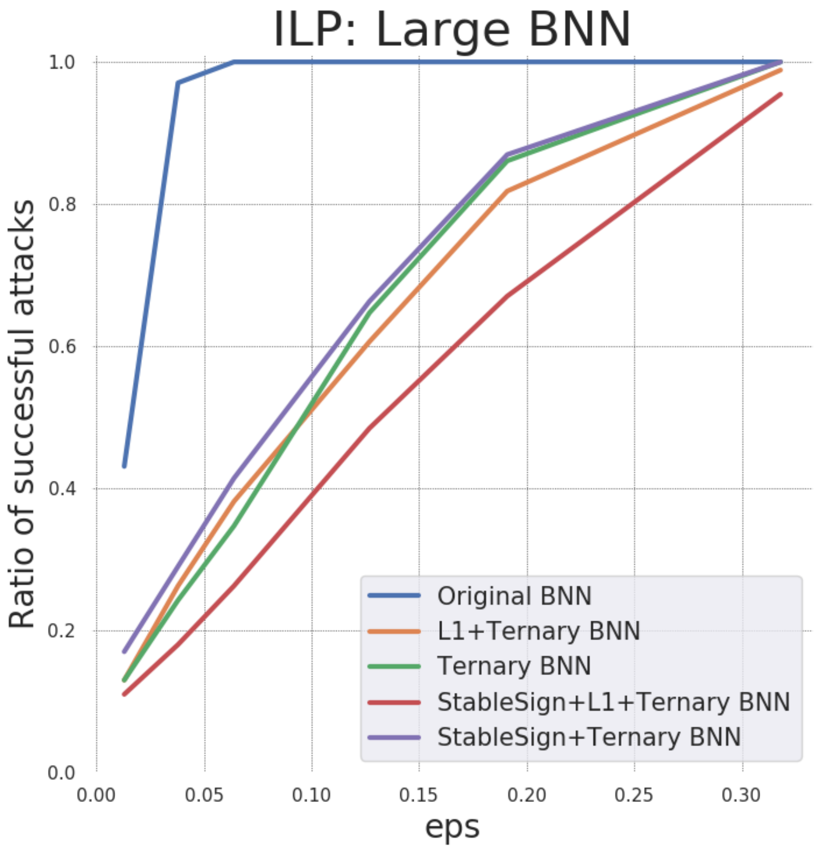
Additional analysis

Running example: Accuracy

	Small BNN	Large BNN
Original BNN	73.0%	74.0%
Ternary BNN	75.2%	78.2%
StableSign+Ternary BNN	76.7%	78.4%
L1+Ternary BNN	75.3%	78.4%
StableSign+L1+Ternary BNN	76.6%	80.0%

Resistance to attacks





Resistance to attacks (median)

Models	Large BNN	
	eps = 5	eps = 10
Original BNN	?	?
Ternary BNN	1%	1%
StableSign+Ternary BNN	1.2%	1.9%
L1+Ternary BNN	2.7%	1.9%
StableSign+L1+Ternary BNN	7%	1.3%

Conclusion

Proposed a method to train a (easier) verifiable BNN.

Conclusion

Proposed a method to train a (easier) verifiable BNN.

- > 10X reduction in the number of coefficients
- ~3 000 000 -> ~50 000 reduction in the #clauses

Thanks!