BNN verification dataset for Max-SAT Evaluation 2020

Masahiro Sakai
masahiro.sakai@gmail.com

***Abstract*—This article describes a MaxSAT benchmarks that encode the problem of finding minimal adversarial examples for binarized neural networks, and the dataset is submitted to MaxSAT Evaluation 2020.**

***Index Terms*—SAT, MaxSAT, binarized neural networks, adversarial examples**

I. Introduction

Binarized Neural Networks (BNN) [1] are neural networks that their parameters (weights) and inputs are restricted to binary value. Due to these characteristics they are memory efficient and computationally efficient but also are suitable for analysis using SAT-based methods.

One of the interested properties about neural networks is existence of *adversarial examples* [2], [3]. Given a neural network f and its input data x, an adversarial example is a slightly modified input data $x' = x + \tau$ which causes misclassification, *i.e.* $f(x) \neq f(x+\tau)$. Existence of adversarial example may pose a security challenge when deploying the model, therefore it is an important problem to analyze the robustness of neural networks against adversarisal examples.

Narodytska, *et al.* [4] proposed a SAT-based methods to certify the absence of adversarial examples for given BNN f and x within $\|\tau\|_\infty \leq \epsilon$. Our formulation is based on theirs, but we recast it to an optimization problem in the form of maximum satisfiablity problem (MaxSAT) instead of a satisfiability problem (SAT) and we also make some other modifications in encoding as well.

II. Input datasets and training

As in [4], we use digit images from MNIST dataset [5], and its two variants: the MNIST-rot and the MNIST-back-image [6]. Each input image is represented as 8-bit 784 ($= 28 \times 28$) dimension vectors $\{0, \dots, 255\}^{784}$.

Our binarized neural networks have the same architecture as in [4] (see the paper for details). We first scale input image $x \in \{0, \dots, 255\}^{784}$ to $[0, 1]^{784}$, and apply (trained) batch normalization and binarization layers to get binarized image $\{-1, +1\}^{784}$. We denote binarization part as

$$\mathrm{bin} : \{0, \dots, 255\}^{784} \to \{-1, +1\}^{784}$$
$$\mathrm{bin}(x_1, \dots, x_{784}) = (\mathrm{bin}_1(x_1), \dots, \mathrm{bin}_{784}(x_{784}))$$

$$\mathrm{bin}_i : \{0, \dots, 255\} \to \{-1, +1\}.$$

Note that the binarization is performed depending on components (*i.e.* pixel locations).

After that, several internal layers and a output linear layer follow. We denote this part as

$$g : \{-1, +1\}^{784} \to \mathbb{R}^{10},$$

and its output is called *logits*.

Then

$$f(x) = \mathrm{argmax}_{c \in \{0,\dots,9\}} g(\mathrm{bin}(x))_c$$

is the function that our binarized neural network computes.

We implemented and trained BNN models using deep learning framework Chainer [7]. Our implementation is available at https://github.com/msakai/bnn-verification .

III. SAT encoding

In the following we identify $\{\mathrm{false}, \mathrm{true}\}$ with $\{0, 1\}$ for notational simplicity.

A. Inputs and decision variables

In the encoding of [4], they use raw 8-bit image ($x' \in \{0, \dots, 255\}^{784}$) as decision variables and add constraints like $x'_i \in [x - \epsilon, x + \epsilon]$.

We instead use binarized image in $\{-1, +1\}^{784}$ as decision variables and represent $\mathrm{bin}_i(x_i)$ as $2z_i - 1$ using a boolean variables z_i. We consider original 8-bit image as dependent variables, and use them in cost function instead of defining them as decision variables.

B. Relationship between inputs and logits

The g part is encoded to CNF in a way similiar to [4], but we use *totalizer* [8] instead of *sequential counter* [9] for encoding cardinality constraints.

C. Relationship between logits and outputs

For the argmax part, let $\mathrm{logits} \in \mathbb{R}^{10}$ be output of g, and let $\{y_c\}_{c \in \{0,\dots,9\}}$ be ten boolean variables representing output of f (*i.e.* $y_i \Leftrightarrow f(x') = i$).

The fact that $\{y_c\}_c$ is a one-hot vector can be expressed as cardinality constraint $\sum_i y_i = 1$.

Also, if y_c then c-th logit must be largest, *i.e.*

$$y_c \to \mathrm{logits}_c \geq \mathrm{logits}_{c'} \text{ for all } c'.$$

Because logits_c is represented as $(\sum_j W_{c,j}(2u_j - 1)) + b_c$ where $W_{c,j} \in \{-1, +1\}$ and u_j are some boolean variables

introduced in encoding of g, this can be expressed as conditional cardinality constraints:

$$y_c \to \sum_j \frac{(W_{c,j} - W_{c',j})}{2} u_j \geq \left\lceil \frac{\sum_j (W_{c,j} - W_{c',j}) + b_{c'} - b_c}{4} \right\rceil.$$

We encode those (conditional) cardinality constraints using totalizer.

Finally, if $f(x) = c$ then we add $\neg y_c$ to require the input to be misclassified.

IV. Cost function

As we want to find an adversarial example with the smallest perturbation, we want to set a cost for modifying x_is. Because our decision variable is z_is, this corresponds to adding a soft constraint[1] $z_i = \frac{bin_i(x_i)+1}{2}$ with appropriate cost of violation.

First, we define δ_i as the smallest change of x_i to flip its binarized value, *i.e.* $\mathrm{bin}_i(x_i+\delta_i) \neq \mathrm{bin}_i(x_i)$. If it is impossible to flip (this is the case when bin_i is constant function), we define δ_i to be $\bot$.

Then it is easy to define cost functions to minimize L_p-norms including L_∞-norm.

A. L_p-norms for $p \neq \infty$

We add $z_i = \frac{bin_i(x_i)+1}{2}$ as a soft constraint with cost $|\delta_i|^p$ for each i if $\delta_i \neq \bot$, and add it as a hard constraint if $\delta_i = \bot$. Note that we get unweighted (partial) Max-SAT instances in case of L_0-norm.

B. L_∞-norm

Let $\Delta = \{|\delta_i| \mid \delta_i \neq \bot\}$ and $w_1 < \ldots < w_{|\Delta|} \in \Delta$ be sorted in ascending order, and we introduce new boolean variable r_k called *relaxation variable* for each w_k. Then we add following constraints.

- Unit soft clause $(\neg r_k)$ with cost $|w_k|$ for each $k \in \{1, \ldots, |\Delta|\}$.
- Relaxation variables are lower closed: $r_k \to r_{k-1}$ for each $k > 0$
- z_i is allowed to be flipped only when corresponding relaxation variable is true: $\neg r_k \to z_i = \frac{bin_i(x_i)+1}{2}$ for each i with $|\delta_i| = w_k$.
- z_i is fixed if it cannot be flipped: $z_i = \frac{bin_i(x_i)+1}{2}$ for each i with $\delta_i = \bot$.

V. Problem files

We choose 20 images (2 images for each digit class) from test data of each data set (mnist, mnist_rot, mnist_back_image) that trained BNN models predicted true label.

Adversarial example finding problem is generated for each image using the BNN model trained for the dataset. Generated instances have 1.8 M variables and 132 M clauses on average.

[1] As we assume a concrete x, right hand side is 0 or 1, therefore this constraint is represented as a unit clause $\neg z_i$ or z_i

File names are in the form of "`bnn_{dataset_name}_{instance number}_label{true label}_adversarial_norm_inf_totalizer.wcnf`" where

- `dataset name` is mnist, mnist_rot or mnist_back_image,
- `instance number` is image number in test dataset,
- `true label` is one of $0 \ldots 9$.

References

[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4107–4115. [Online]. Available: http://papers.nips.cc/paper/6573-binarized-neural-networks.pdf

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[3] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[4] N. Narodytska, S. P. Kasiviswanathan, L. Ryzhyk, M. Sagiv, and T. Walsh, "Verifying properties of binarized deep neural networks," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, S. A. McIlraith and K. Q. Weinberger, Eds. AAAI Press, 2018, pp. 6615–6624. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16898

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 473–480. [Online]. Available: https://doi.org/10.1145/1273496.1273556

[7] S. Tokui, R. Okuta, T. Akiba, Y. Niitani, T. Ogawa, S. Saito, S. Suzuki, K. Uenishi, B. Vogel, and H. Yamazaki Vincent, "Chainer: A deep learning framework for accelerating the research cycle," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 2002–2011.

[8] O. Bailleux and Y. Boufkhad, "Efficient cnf encoding of boolean cardinality constraints," in *Principles and Practice of Constraint Programming – CP 2003*, F. Rossi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 108–122.

[9] C. Sinz, "Towards an optimal CNF encoding of boolean cardinality constraints," in *Principles and Practice of Constraint Programming - CP 2005*, P. van Beek, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 827–831.