The VNNLIB standard for benchmarks
— DRAFT —

Clark Barrett, Guy Katz, Dario Guidotti,
Luca Pulina, Nina Narodytska, Armando Tacchella

DIBRIS — University of Genoa

Abstract

The purpose of this document is to propose a standard for neural network verification benchmarks to be stored in VNNLIB. The standard builds on the Open Neural Network Exchange (ONNX) format for model description, and on the Satisfiabily Modulo Theories Library (SMTLIB) format for property specification. Throughout the document we will refer to the combination of model and property description formats as the *VNNLIB format*.

0.1 Introduction

This document is structured as follows. In Section 0.2 we present a proposal for a model specification language, i.e., how to describe the neural network for verification purposes; in Section 0.3 we present a proposal for a property specification language, i.e., how to specify the requirements that the neural network ought to satisfy. Finally, in Section 0.4 we present some examples of benchmarks to further illustrate the capabilities and the limitations of the VNNLIB format.

0.2 Modeling language

Referring to the official model zoo supplied by ONNX we have searched for a subset of operators which would allow us to represent the majority of models in the zoo and, at the same time, to limit the variety of operators considered. We consider the zoo as representative for the kind of model architecturs, and thus specific operators, that are commonly used in the Machile Learning community. In particular, we have focused on models used in image classification tasks. We refer to the official ONNX documentation for additional information about different operators.

The subset of ONNX operators we consider for the VNNLIB format is briefly described in the following:

- *AveragePool (Average Pooling)* operator supports downsampling with averaging.
- *BatchNormalization (Batch Normalization)* operator supports adjusting and scaling the activations functions, and it is expressive enough to represent general batch normalization.
- *Conv (Convolutional)* operator supports all the attributes to encode a generic convolutional layer.
- *Dropout (Dropout)* operator supports random dropping of units (during training). This operator should not appear on trained models.
- *Flatten (Flatten)* this operator converts multidimensional arrays (tensors) to single dimensional ones; it is used instead of *Reshape* in some of the models in the zoo.
- *Gemm (General Matrix Multiplication)* operator encodes matrix multiplication possibly with a scalar coefficient and the addition of another

matrix; as such *Gemm* can encode fully connected layers in neural networks.

- *LRN (Local Response Normalization)* operator supports normalization over local input regions; it is uatilized in Alexnet and derived networks.
- *MaxPool (Maximum pooling)* operator supports downsampling with maximization.
- *ReLU (Rectified Linear Unit)* operator encodes the corresponding activation function $\sigma(x) = \max(0, x)$.
- *Reshape (Reshape)* operator supports reshaping of the tensor's dimensions.
- *Sigmoid (Logistic Unit)* operator encodes the corresponding activation function $\sigma(x) = \frac{1}{1+e^{-x}}$.
- *SoftMax (Softmax Unit)* operator transforms vectors into probabilities, e.g., for selecting among different classes and it is commonly utilized in state of the art networks.
- *Unsqueeze (Unsqueeze)* operator removes dimensions of size 1 from tensors, and it is utilized, e.g., in *Densenet* and *Inception2*.

Overall, the collection of operators considered above is sufficient to encode the majority of the image classification networks provided in the ONNX model zoo.

0.3 Property specification language

Inputs and outputs of operators are *tensors*, i.e., multidimensional arrays over some domain, usually numerical. If we let $\mathbb{D}$ be any such domain, a k-dimensional tensor on $\mathbb{D}$ is denoted as $x \in \mathbb{D}^{n_1 \times \ldots \times n_k}$. For example, a vector of n real numbers is a 1-dimensional tensor $x \in \mathbb{R}^n$, whereas a matrix of $n \times n$ Booleans is a 2-dimensional tensor $x \in \mathbb{B}^{n \times n}$ with $\mathbb{B} = \{0, 1\}$. A specific element of a tensor can be singled-out via *subscripting*. Given a k-dimensional tensor $x \in \mathbb{D}^{n_1 \times \ldots \times n_k}$, the element $x_{i_1,\ldots,i_k} \in \mathbb{D}$ is a scalar corresponding to the indexes $i_1, \ldots, i_k$. For example, in a vector of real numbers $x \in \mathbb{R}^n$, x_1 is the first element, x_2 the second and so on. In a matrix of Boleans $x \in \mathbb{B}^{n \times n}$, $x_{1,1}$ is the first element of the first row, $x_{2,1}$ is the first element of the second and so on.

An *operator* f is a function on tensors $f : \mathbb{D}^{n_1 \times n_h} \to \mathbb{D}^{m_1 \times m_k}$ where h is the dimension of the input tensor and k is the dimension of the output tensor. Given a set $F = \{f_1, \ldots, f_p\}$ of p operators, a *feedforward neural network* is a function $\nu = f_p(f_{p-1}(\ldots f_2(f_1(x))\ldots))$ obtained through the composition of the operators in F assuming that the dimensions of their inputs and outputs are *compatible*, i.e., if the output of f_i is a k-dimensional tensor, then the input of f_{i+1} is also a k-dimensional tensor, for all $1 \leq i < p$.

Given a neural network $\nu : \mathbb{D}^{n_1 \times n_h} \to \mathbb{D}^{m_1 \times m_k}$ built on the set of operators $\{f_1, \ldots, f_p\}$, let $x \in \mathbb{D}^{n_1 \times n_h}$ denote the input of ν and $y_1, \ldots, y_p$ denote the outputs of the operators $f_1, \ldots, f_p$ — therefore y_p is also the output y of ν. We assume that, in general, a *property* is a first order formula $P(x, y_1, \ldots y_p)$ which should be satisfied given ν.

TODO: Expand the part related to conditions and the connection to operators

The common format we propose to use to formalize neural networks properties is the SMT-LIB language. This is a well known format used to formalize Satisfiability Modulo Theories problems. We believe it is powerful enough to represent the properties of interest. In the following we will show some example of networks and corresponding properties in SMT-LIB language.

0.4 Examples

0.4.1 ACAS XU

We consider a standard example of an ACAS XU network with 6 hidden layers of 50 ReLU neurons each; input and output layers consists of 5 neurons. In Figure 0.4.1 we show a graphical representation of this model. ACAS XU is thus a function $\nu : I^5 \to O^5$ with $I = O = \mathbb{R}$. A property of interest for this kind of network can be stated as follows:

$$
\begin{aligned}
&-\varepsilon_0 \leq x_0 \leq +\varepsilon_0 \\
&-\varepsilon_1 \leq x_1 \leq +\varepsilon_1 \\
&-\varepsilon_2 \leq x_2 \leq +\varepsilon_2 \\
&-\varepsilon_3 \leq x_3 \leq +\varepsilon_3 \\
&-\varepsilon_4 \leq x_4 \leq +\varepsilon_4 \\
&+y_0 - y_1 \leq 0 \\
&+y_0 - y_2 \leq 0 \\
&+y_0 - y_3 \leq 0 \\
&+y_0 - y_4 \leq 0
\end{aligned}
\tag{1}
$$

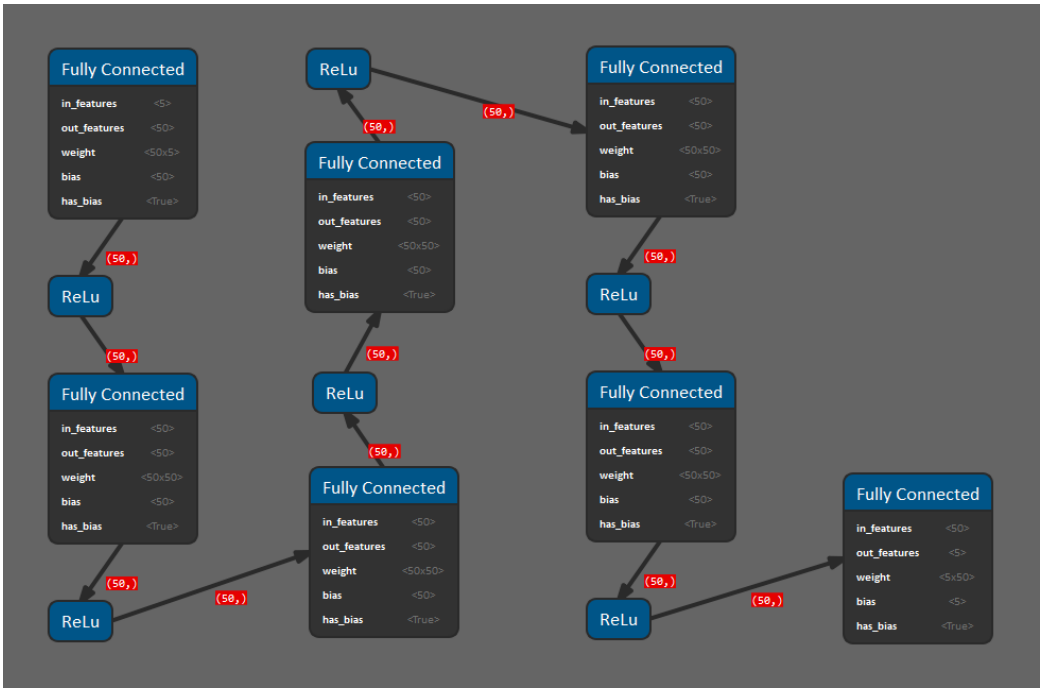

Figure 1: Graphical representation of a ONNX model of an ACAS XU network. The image is generated using our tool NeVer2.

where $\varepsilon_i \in \mathbb{R}+$ with $0 \leq i \leq 4$ are arbitrary positive constants, $(x_0, \ldots, x_4) \in \mathbb{R}^5$ is an input vector and $(y_0, \ldots, y_4) \in \mathbb{R}^5$ is an output vector.

The property can be experessed in the SMT-LIB language using specific identifiers for input and output vectors derived from the identifiers used in the ONNX model. In particular, ONNX only provides an identifier for the output of each layer and a global identifier for the network input; for this reason we identify the input tensor as **X** and the output tensor as the last Fully Connected (*Gemm*) node identifier **FC6**, both with dimension 5. We can then express property (1) in SMTLIB as:

```
; definition of the variables of interest
(declare-fun X_0 () Real)
(declare-fun X_1 () Real)
(declare-fun X_2 () Real)
(declare-fun X_3 () Real)
(declare-fun X_4 () Real)
(declare-fun FC6_0 () Real)
```

```
(declare-fun FC6_1 () Real)
(declare-fun FC6_2 () Real)
(declare-fun FC6_3 () Real)
(declare-fun FC6_4 () Real)

; numeric coefficients
(declare-fun eps_0 () Real)
(declare-fun eps_1 () Real)
(declare-fun eps_2 () Real)
(declare-fun eps_3 () Real)
(declare-fun eps_4 () Real)

; definition of the constraints
(assert (<= X_0 eps_0))
(assert (>= X_0 -eps_0))
(assert (<= X_1 eps_1))
(assert (>= X_1 -eps_1))
(assert (<= X_2 eps_2))
(assert (>= X_2 -eps_2))
(assert (<= X_3 eps_3))
(assert (>= X_3 -eps_3))
(assert (<= X_4 eps_4))
(assert (>= X_4 -eps_4))
(assert (<= (- FC6_0 FC6_1) 0.0))
(assert (<= (- FC6_0 FC6_2) 0.0))
(assert (<= (- FC6_0 FC6_3) 0.0))
(assert (<= (- FC6_0 FC6_4) 0.0))
```

This fragment of code is compliant with the SMT-LIB 2 language standard and it defines the property of interest in term of input-output relations. Clearly, this code is not complete in terms of internal constraints of the network: these will be extracted based on the verification methodology of interest from the ONNX model of the network.

In Figure 0.4.1 we show how properties are defined within NeVer2. Whenever a property is linked to a node it is considered a post-condition on the output of that node, whereas a property with no connection is considered a pre-condition on the network input.

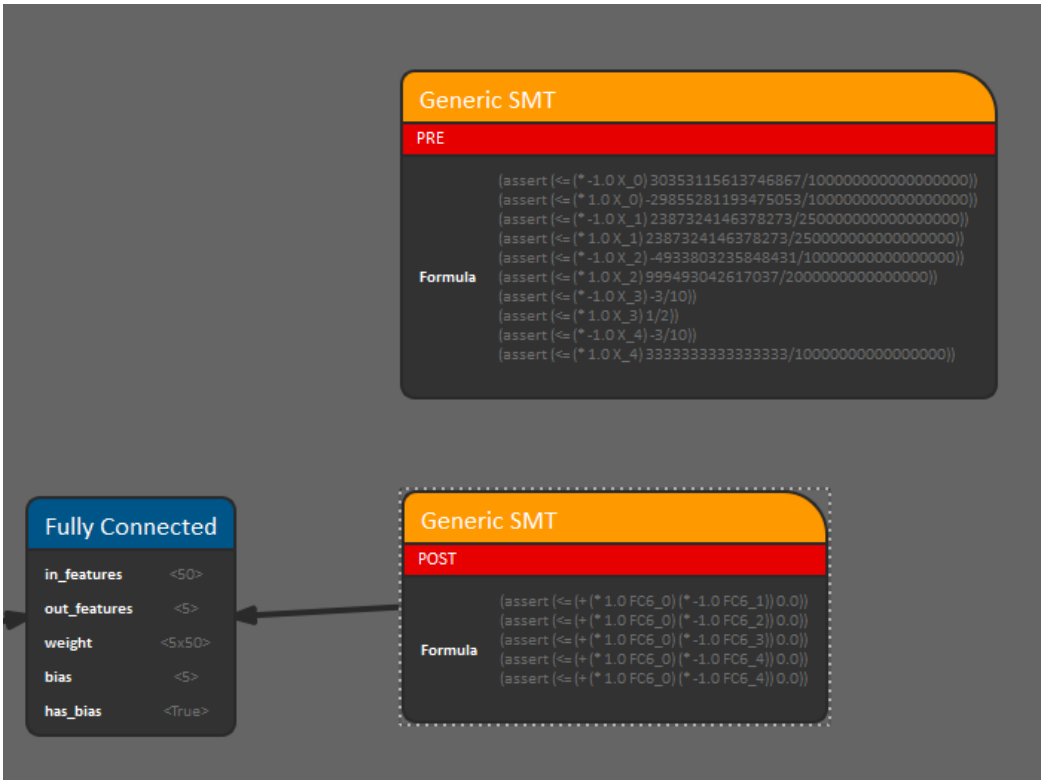

Figure 2: Graphical representation of property representation in our tool NeVer2.

0.4.2 MNIST

We consider a standard example of a convolutional MNIST network with two convolutional layers consisting of a convolution with a 5×5 kernel with 2 pixel padding, a ReLU activation function and a Max Pooling with a 2×2 kernel. The first convolution generates 16 channels, and the second 32; the result is flattened to a single vector which is finally fed to a Fully Connected layer. The input layer consists of a three-dimensional tensor $1 \times 28 \times 28$ and the output layer consists of 10 neurons for the classification. In Figure 0.4.2 we show a graphical representation of this model. MNIST is thus a function $\nu : I^{1,28,28} \to O^{10}$ with $I = O = \mathbb{R}$.

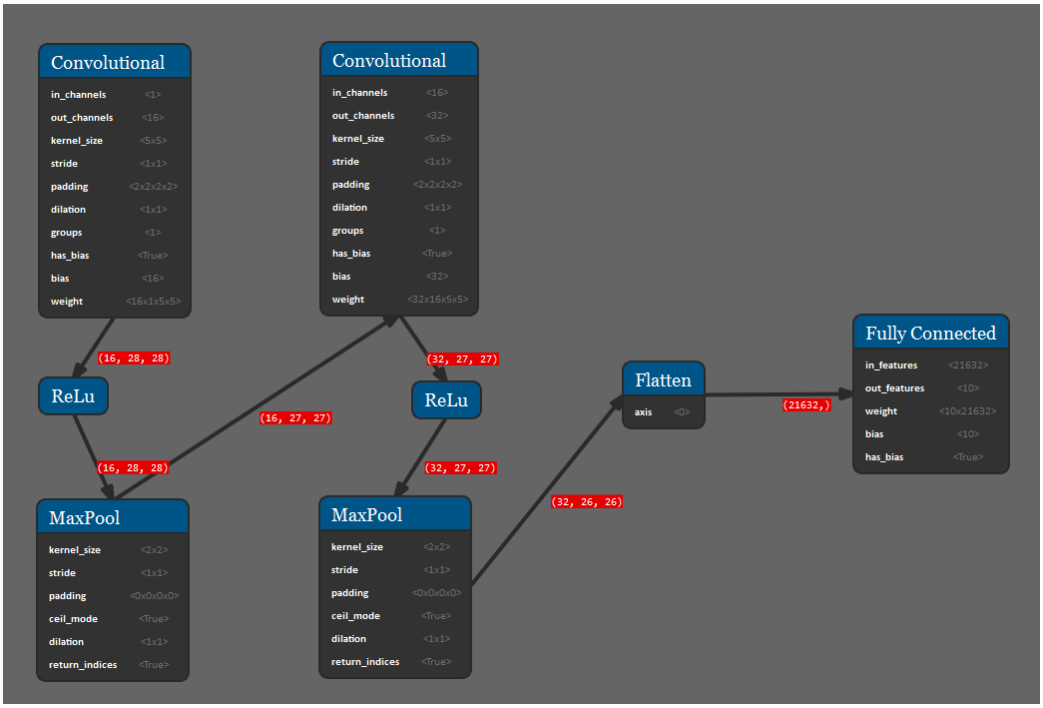

Figure 3: Graphical representation of a ONNX model of a convolutional MNIST network. The image is generated using our tool NeVer2.

Bibliography