User Manual
DLV Software (Version 0.1)

Xiaowei Huang

University of Oxford

This document provides instructions on how to install and use the software DLV (abbreviated for Deep Learning Verifier) version 0.1.

NB: The document will be updated. Please come back and check to see more details.

1 Installation

The software is dependent on the following packages:

- Python 2.7, which is the shipped version in Mac OSX version 10.11.5.
- Theano, version 0.8.2, which can be installed with the following command:

```
pip install Theano
```

- Keras, which can be installed with the following command:

```
pip install keras
```

The software may require other python libraries such as opencv, numpy, math, etc.

2 Usage

The following command can be used to call the program:

```
python main.py
```

Please use the file "configuration.py" to set the parameters for the system to run. More instructions on how to adapt the parameters will be given in Section 4.

3 Neural Network Models

The software has been configured to work with four neural network models. For the first three models, one may either train a model first and then verify or verify a pre-trained model included in the package without training. For the last model, one can only work with a pre-trained model downloaded from internet (link provided below): it is impractical to train the model since there is no data available for training and the training may take months of time.

3.1 A Classification Network for A Pre-defined 2D Curve

The network structure and the parameters for training can be found in file "twoDcurve_network.py".

3.2 A Classification Network for MNIST Handwritten Dataset

The source file for training the network is downloaded from the following link:

https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

3.3 A Classification Network for CIFAR-10 Small Image Dataset

The source file for training the network is downloaded from the following link:

https://github.com/fchollet/keras/blob/master/examples/cifar10_cnn.py

3.4 A Classification Network for ImageNet Dataset

One needs to download the VGG16 pre-trained model from the following link

https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3

Moreover, the images need to be in the directory networks/imageNet/, and change the corresponding paths in networks/imageNet_network.py

4 Parameters

In this section, we explain how to adapt the parameters for experiments.

task defines the task for execution. In current release, there is only one option.

task = "safety_check"

dataset defines which neural network to work with. One of the following lines should be included and other three lines are commented out by prefixing a symbol #.

dataset = "twoDcurve"
dataset = "mnist"
dataset = "cifar10"
dataset = "imageNet"

experimental_config defines whether to choose the default experimental setting specified in the file "usual_configuration.py". One of the following lines should be included and the other commented out.

experimental_config = True
experimental_config = False

whichMode defines the execution mode: “train” means that the neural network will be trained first and then be verified; “read” means to verify directly the pre-trained neural network. The trained model is stored in their individual directories, e.g., networks/mnist/ for MNIST dataset and networks/cifar10/ for CIFAR-10 dataset. One of the following lines should be included and the other commented out.

whichMode = “read”
whichMode = “train”

dataProcessing defines how to process examples: “single” means that only the example whose index is specified in the parameter *startIndexOfImage* will be processed; “batch” means that a set of *n* examples will be processed, starting from the example whose index is specified in the parameter *startIndexOfImage*. The number *n* is specified in parameter *dataProcessingBatchNum*. One of the following lines should be included and the other commented out.

dataProcessing = “single”
dataProcessing = “batch”

dataProcessingBatchNum defines the number of examples to be processed in a single execution of the program if *dataProcessing* is set to be “batch”.

dataProcessingBatchNum = 200

span defines the size of a span for manipulation in a dimension. Together with the parameter *numSpan*, it defines the size of a dimension to work with. Please refer to the paper for the detailed explanation of this parameter.

span = 255/float(255)

numSpan defines the number of spans to be considered for defining the size of a dimension.

numSpan = 1.0

featureDims defines the number of dimensions for a feature.

featureDims = 5

startIndexOfImage defines the index of an example from the test data set. For MNIST data set, it is in [0,9999].

startIndexOfImage = 197

startLayer defines the first layer to start verifying. Note that, for both MNIST, CIFAR-10, and ImageNet networks, *startLayer* = 0 suggests the first hidden layer.

startLayer = 0

maxLayer defines the deepest layer to verify until.

maxLayer = 0

numOfFeatures defines the number of features to be considered when verifying a layer.

numOfFeatures = 40

checkingMode defines how to verify layer by layer: for "specificLayer", it directly verifies the one specified in *maxLayer*; for "stepwise", it starts from the layer *startLayer* and work layer by layer until completing the *maxLayer*.

checkingMode = "specificLayer"
checkingMode = "stepwise"

5 Output Statistics

In "networks/network_configuration.py" file, one may define or change the directories related to the outputs. It is handy to assign different directories to different neural networks.

directory_model_string specifies the directory in which the pre-trained model should be stored.

directory_pic_string specifies the directory in which the resulting images are stored.

directory_statistics_string specifies the directory in which the text files containing statistics are stored.

6 Further Explanations of Results

For smaller images such as MNIST and CIFAR-10, manipulated images usually have lower confidence than the original images, which may have confidence more than 99%. However, for larger images such as ImageNet, it is possible that manipulated images have higher confidence than the original images, because the confidence of original images may be not as high as those of small images. The confidence change can be e.g., 30% $\rightarrow$ 40%.