ReachNN*

ReachNN* is a reachability analysis approach based on Bernstein polynomials that can verify neural-network controlled systems (NNCSs) with a more general form of activation functions. We further explore how the specific property, like Lipschitz constant, of the network influences the verification results and propose a new Verification-aware Knowledge Distillation method to distill a new neural network controller that is more verification-friendly and retain the knowledge from the orignal neural network controller.

ReachNN* is a GPU implementation of the originally proposed ReachNN tool and integrate the function of Verification-aware Knowledge Distillation. Experiment results across a set of benchmarks show 7x to 422x efficiency improvement in terms of runtime

Execution on VM

ReachNN* can be tested in Virtual Machine, please download it and import the .ova file using Oracle VM Virtual Box.

We have tested our code with RAM size 3072 MB. The tool is stored in folder ~/ReachNNStar.

In the home directory, credentials.txt includes information about user name and password. readme.pdf contains the instruction to test our tool. howto_vbox_shared_folder.txt is the instruction about file sharing between the virtual machine and your desktop.

We note that since VM does not support GPU usage and has limited RAM, the runtime result could be different from what we report. Our tool does not include the installation of Sherlock and Verisig for comparison. For the comparison with Verisig and Sherlock results, please refer to the results in ReachNN[1].

System Requirements

Ubuntu 18.04, Python 3.6

Running Examples

NOTE: All the capitalized word is the input argument and has no suffix.

Please activate the python virtualenv before running any examples.

```
source ~/venv/bin/activate
```

Reachability Analysis for NNCS

This will replicate the experiments results from #1 to #6.

For examples #1 to #5, the program will require at least 3GB RAM to run.

For example #6, the program will require at least 8GB RAM to run. We recommend to run it on the desktop instead of VM.

```
cd ~/ReachNNStar/ReachNN

# example 1 to 5
./run_exp.sh

# example 6
# note that this program requires at least 8 GB RAM memory to run.
./run_tora.sh
```

The verification results will return to ~/ReachNNStar/ReachNN/outputs/SYSTEM.txt.

The computed flowpipes will be plotted to ~/ReachNNStar/ReachNN/outputs/image/SYSTEM.eps after the program is finished.

Run the user specified NNCS

The code needs to run under the directory ~/ReachNNStar/ReachNN/ with python virtualenv activated.

Please refer to the template in run_exp.sh

The neural network description file is in ~/ReachNNStar/ReachNN/Bernstein_Polynomial_Approximation/nn/

The cpp file that model the system are in ~/ReachNNStar/ReachNN/Bernstein_Polynomial_Approximation/systems/

```
# SYSTEM is the example's cpp filename and network filename; ERROR_BOUND depends on t
he system's sensitivity
./example.sh SYSTEM ERROR_BOUND
```

Checking Result

All results will be stored in ReachNNStar/ReachNN/outputs/

Check the result of SYSTEM

```
# verification result
vim SYSTEM.txt

# plotted flowpipes
gnuplot SYSTEM.plt
```

Check the figures in outputs/images/

Verification-Aware Knowledge Distillation

This will replicate the after KD result in example #1, #2 and #6.

In this section, the new network will be trained given the original network in example #1, #2 and #6. Then, the new networks are fed to the reachability analysis module to obtain the new verification results.

For example #1 and #2, the program will require at least 3GB RAM to run.

For example #6, the program will require at least 8GB RAM to run.

```
cd ~/ReachNNStar/VF_Retraining

# example 1 and 2
./run_distillation_limited_memory.sh

# example 1, 2 and 6
./run_distillation.sh
```

Please check the result in ~/ReachNNStar/ReachNN/outputs/SYSTEM_retrained.txt and ~/ReachNNStar/ReachNN/outputs/images/SYSTEM_retrained.eps.

The one without the "retrained" suffix is the result of the original network.

Run the user specified NNCS

```
cd ~/ReachNNStar/VF_Retraining

# put the original network NETWORK_FILENAME in folder nn/
cp NETWORK_FILENAME nn/

# run KD to distill a new network
./example.sh NETWORK_FILENAME NETWORK_FILENAME_RETRAINED ACTIVATION L_TARGET REGRESSI
ON_ERROR_BOUND SCALAR OFFSET
```

The NETWORK_NEW_FILENAME will be shown in folder nn_retrained/.

After distillation, to rerun the reachability analysis on the new NN, execute the following commands:

```
# put the new network into the reachability analysis module
cp nn_retrained/NETWORK_FILENAME_RETRAINED ../ReachNN/Bernstein_Polynomial_Approximat
ion/nn/

# create a new system file to redo reachability analysis
cd ../ReachNN/Bernstein_Polynomial_Approximation/systems

cp NETWORK_FILENAME.cpp NETWORK_FILENAME_RETRAINED.cpp

vim NETWORK_FILENAME_RETRAINED.cpp # change the network name to NETWORK_FILENAME_RETR
AINED in the cpp file and change the output file name to NETWORK_FILENAME_RETRAINED t
oo.

cd ../../

./example.sh NETWORK_FILENAME_RETRAINED ERROR_BOUND
```

Checking Result

All results will be stored in ~/ReachNNStar/ReachNN/outputs/

Check the result with NETWORK_FILENAME_RETRAINED

```
# verification result
vim NETWORK_FILENAME_RETRAINED.txt

# plotted flowpipes
gnuplot NETWORK_FILENAME_RETRAINED.plt
```

Check the figures in outputs/images/

Reproduce the result in Example Usage

```
cd ~/ReachNNStar/VF_retraining

# If there is less than 8 GB memory, please run the following command
./example_usage_limited_memory.sh

# If there is at least 8GB memory, please run the following command
./example_usage.sh
```

The results will be reported in ~/ReachNNStar/ReachNN/outputs/ with filename nn_1_relu_tanh_origin.txt and nn_1_relu_tanh_retrained.txt. The plotted flowpipes are shown in ~/ReachNNStar/ReachNN/outputs/images/ with filename nn_1_relu_tanh_origin.eps and nn_1_relu_tanh_retrained.eps.

Contributors

Jiameng Fan, Chao Huang, Wenchao Li, Xin Chen, Qi Zhu

References

[1] C.Huang, J.Fan, W.Li, X.Chen, and Q.Zhu. Reachnn: Reachabilityanalysisofneural-network controlled systems. ACM Transactions on Embedded Computing Systems, 18:1–22, 10 2019. doi: 10.1145/3358228.

[2] J.Fan, C.Huang, W.Li, X.Chen, and Q.Zhu. Towards Verification-Aware Knowledge Distillation for Neural-Network Controlled Systems. International Conference on Computer Aided Design (ICCAD), November 2019.

Common Issues

m4: unrecognized option '--gnu'

To fix this problem, try to reinstall m4 first `sudo apt-get install --reinstall m4`