DCNNN

User Manual

Version 1.0

Haoyuan Sun, Yan Du, Fangxing Li

December 30, 2019

1 Introduction

1.1 Background

DCNNN is a power system N − 1 contingency screening tool based on deep Convolutional Neural Network (deep CNN). This tool is used to accelerate N − 1 contingency screening of power systems by training a deep convolutional neural network to calculate AC power flows under N − 1 contingency and uncertain scenarios.

The algorithms in DCNNN were developed by Yan Du of the Enliten Lab at The University of Tennessee, Knoxville under the direction of professor Fangxing Li [1]. Haoyuan Sun designed and coded this GUI for these algorithms.

This tool works in the following way. ① Data Generation: The power flow of the specified test case is solved with Matpower [2] for multiple times under different load and wind power scenarios. ②Training and Testing: The generated data are divided into training set and test set according to a user defined ratio and are used to train and test a deep CNN network.

1.2 Citing DCNNN

We request that the use of DCNNN be acknowledged by citing the following paper [1].

> Y. Du, F. Li, J. Li and T. Zheng, "Achieving 100x acceleration for N-1 contingency screening with uncertain scenarios using deep convolutional neural network," *IEEE Trans. on Power Syst.*, vol. 34, no. 4, pp. 3303-3305, July 2019. DOI: 10.1109/TPWRS.2019.2914860

2 Getting Started

2.1 System Requirements

To use DCNNN 1.0, you will need:

Python 3.6
Tensorflow 1.6
Sklearn

We recommend the Miniconda distribution which includes the conda package manager and Python. Please download and install the latest Miniconda (x64 with Python 3) from https://conda.io/miniconda.html. Use the default installation settings.

Open the Anaconda Prompt and create an environment for DCNNN (optional)

```
conda create --name DCNNN python=3.6

conda activate DCNNN
```

Install Tensorflow and Sklearn

```
pip install tensorflow==1.6

pip install sklearn
```

Add the environment folder to the system environment variable 'Path'

1. Locate the 'DCNNN' environment you just created (or another environment you plan to use). Usually, it is in "C:\Users\[your user name]\Miniconda3\envs\DCNNN or [your environment

name]" or "C:\Users\[your user name]\.conda\envs\DCNNN or [your environment name]". Copy this path.

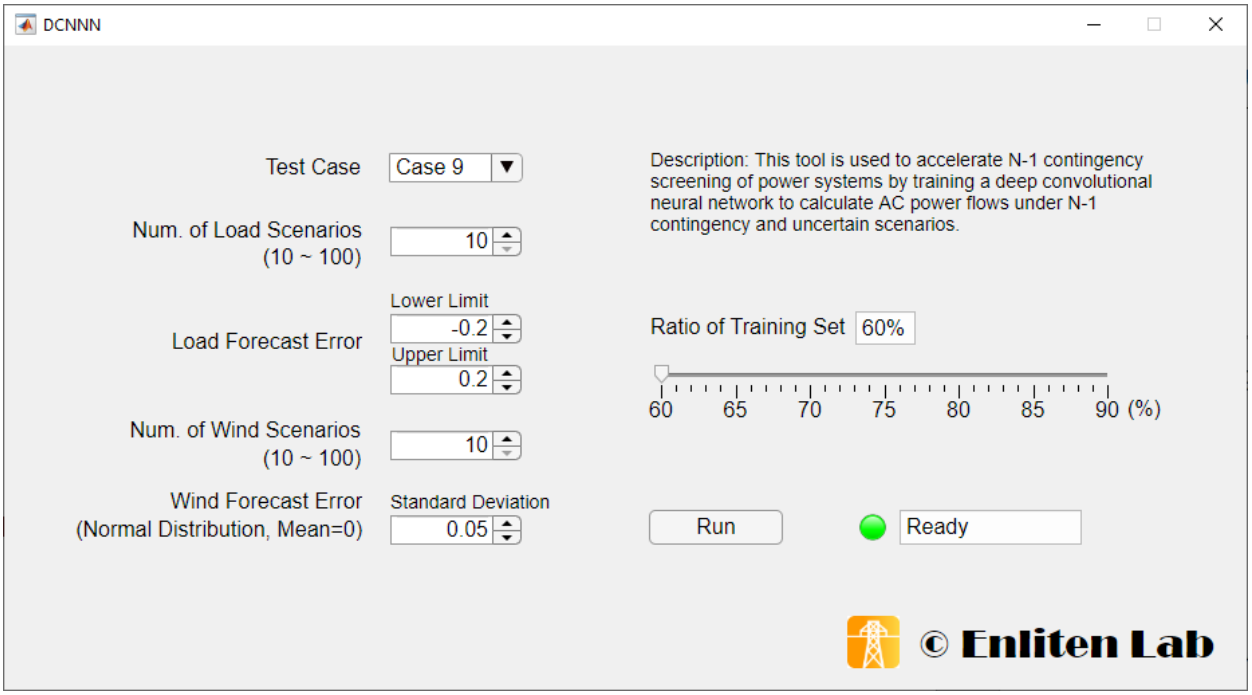
2. Start the system control panel (Control Panel – System), and select the advanced tab.
3. Click the environment variables button.
4. Under system variables, select 'Path', then click Edit.
5. Add the path you just copied.

2.2 Installation

Start 'DCNNN_Installer.exe' and follow the instructions. Default installation settings are recommended.

3 Training and Testing a Network

You will see a command shell and a control panel (shown as below) when you start DCNNN.



The command shell is where training and test results will be displayed. All results are also in a separate 'txt' file for later reference. On the control panel, you can adjust the parameters according to your demand. Here are some suggestions on how to select the parameters:

Name of Parameter	**Comments from Designers**
Test Case	IEEE standard test systems. The case name indicates the number of buses in the system.
Num. of Load Scenarios (10 ~ 100)	Set at scale of 10. More scenarios mean a more generalized network but also longer training time.
Load Forecast Error	Larger error means a more generalized network but also less accuracy. Recommended range is −0.2 ~ +0.2 [3].

Num. of Wind Scenarios (10 ~ 100)	Set at scale of 10. More scenarios mean a more generalized network but also longer training time.
Wind Forecast Error − Standard Deviation (Normal Distribution, Mean = 0)	Larger error means a more generalized network but also less accuracy. Recommended value is 0.05 [4].
Ratio of Training Set	Recommended value is 70% ~ 80% according to designers' experience.

Once all parameters are set, click 'Run' to start training and testing a network. All results will be saved in the folder named 'casex_yyyymmdd_mmss', e.g. case9_20191230_1008.

4 References

[1] Y. Du, F. Li, J. Li and T. Zheng, "Achieving 100x acceleration for N-1 contingency screening with uncertain scenarios using deep convolutional neural network," *IEEE Trans. on Power Syst.*, vol. 34, no. 4, pp. 3303-3305, July 2019. DOI: 10.1109/TPWRS.2019.2914860

[2] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "Matpower: steady-state operations, planning and analysis tools for power systems research and education," *IEEE Trans. on Power Syst.*, vol. 26, no. 1, pp. 12−19, Feb. 2011. DOI: 10.1109/TPWRS.2010.2051168

[3] Y. Liu, N. Zhang, Y. Wang, J. Yang and C. Kang, "Data-driven power flow linearization: a regression approach," *IEEE Trans. on Smart Grid*, vol. 10, no. 3, pp. 2569-2580, May 2019. DOI: 10.1109/TSG.2018.2805169

[4] D. T. Nguyen and L. B. Le, "Risk-constrained profit maximization for microgrid aggregators with demand response," *IEEE Trans. on Smart Grid*, vol. 6, no. 1, pp. 135-146, Jan. 2015. DOI: 10.1109/TSG.2014.2346024