Artifact Evaluation

This section outlines the steps to recreate the plots presented in the paper. It is worth noting that the safety verification method proposed in our work is inherently *statistical*. Consequently, the reproduction of the plots will not yield an exact match but rather a *stochastic recreation*. In other words, the plots are recreated using the same set of parameters as in the draft, which define the distributions. The recreated plots represent one possible outcome from that distribution, hence the term "stochastic recreation". As a result, some figures that were initially inferred to be safe in the draft may yield unsafe results (or vice versa) during this process.

It is important to note that this does not affect the soundness of our method. Rather, it reaffirms the stochastic nature of our work, and such stochastic recreations demonstrate the influence of various parameters on the proposed method, as explored in the research questions of the paper. In simpler terms, while the recreated plots may not be identical (in a deterministic sense), they are stochastically equivalent.

To elaborate on the sources of stochasticity, the following factors contribute to the variability in the generated plots (testing this by running the scripts multiple times could be a good approach):

1. Although the same parameters (logging probability and log noise) are used to generate the logs, the logs themselves are *random*, though stochastically equivalent, as they are derived from the same distribution.

2. Additionally, the safety verification process is inherently stochastic, meaning that safety inferences could vary across different iterations.

Installation & Environment

Python Dependencies

Install the libraries actually used by this codebase:

```
pip install numpy matplotlib docopt tensorflow
```

You can also use keras directly instead of tensorflow; the code tries:

```
from tensorflow.keras.models import load_model
# falls back to:
from keras.models import load_model
```

Environment Variable: POSTO_ROOT_DIR

Many modules expect the project root via:

```
PROJECT_ROOT = os.environ['POSTO_ROOT_DIR']
sys.path.append(PROJECT_ROOT)
```

Set it once:

```
export POSTO_ROOT_DIR=/absolute/path/to/Posto
```

Add this export to your `/.bashrc` file to make it permanent.

Running the Posto Artifact (OVA)

This artifact is distributed as a pre-configured VirtualBox virtual machine to ensure full reproducibility of the experimental results reported in the paper.

1. Install Oracle VirtualBox (version 7.0 or later) from the official website: `https://www.virtualbox.org/wiki/Downloads` Please ensure that the **VirtualBox Extension Pack** corresponding to the same version is also installed.
2. Download the Posto zip from the following link and unzip the contents:

   `https://alabama.box.com/s/6gn0u0anx0wm8tavhff21qy7clrjgs5b`
3. Open **VirtualBox Manager**
4. Select **File → Import Appliance**
5. Choose the downloaded `.ova` file from the unzipped contents
6. Click **Next**, then **Import**
7. Start the imported virtual machine

No additional configuration or installation is required.

Posto Location inside the Virtual Machine

After logging into the virtual machine, the Posto tool is located at:

```
~/Desktop/Posto
```

To access it, open a terminal and run:

```
cd ~/Desktop/Posto
```

From this directory, all commands described in the paper and appendices (including artifact evaluation scripts) can be executed directly.

Recreating Figures

1. **To recreate figs. A.2(a)-(c), A.3(a)-(d), A.4(a)-(d) and B.5 perform the following steps:**

   (a) Make sure you are in location `/path/to/Posto`

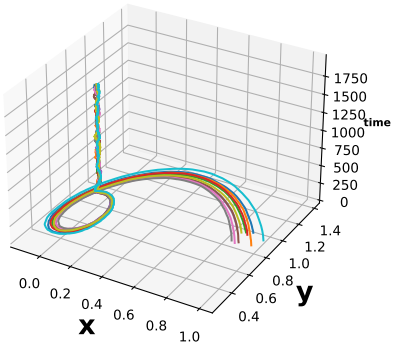
   ```
   cd /path/to/Posto
   ```

   (b) Run the following command if the location is not added to the `/.bashrc`

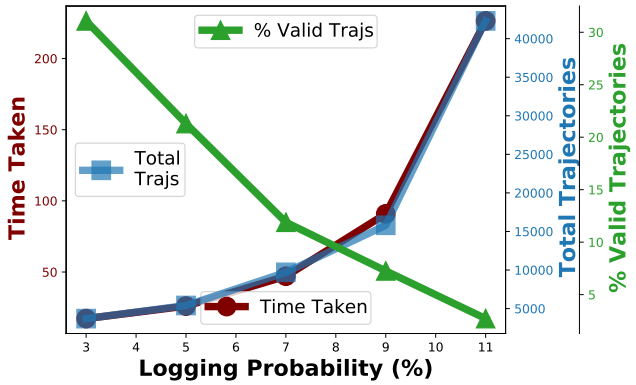
   ```
   export POSTO_ROOT_DIR=/path/to/Posto
   ```

   (c) Run the `artEval.py` with the figure number as the argument to recreate the desired figure

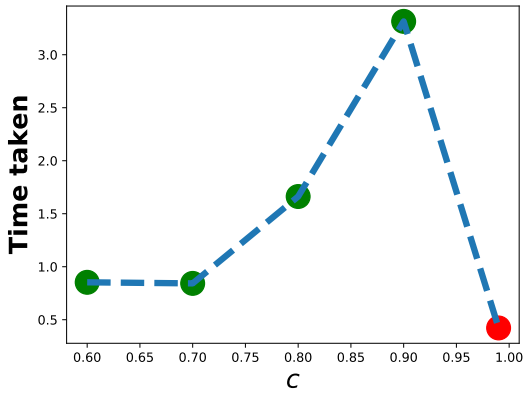
   ```
   python artEval.py --fig=A2a
   ```



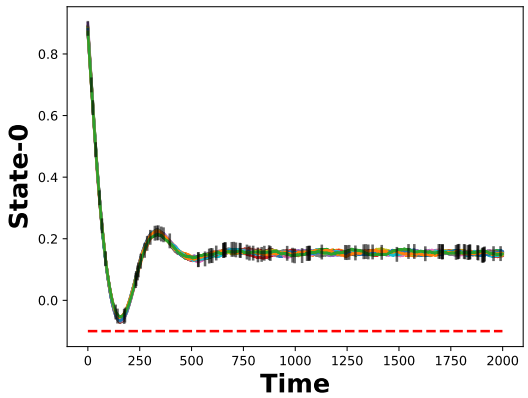
```
python artEval.py --fig=A2b
```




```
python artEval.py --fig=A2c
```

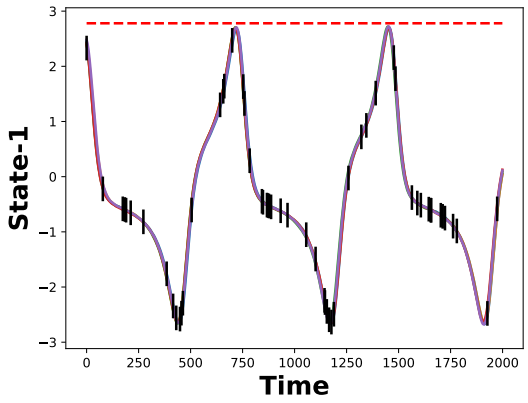



```
#Choose from A3a, A3b, A3c, A3d
python artEval.py --fig=A3a
```

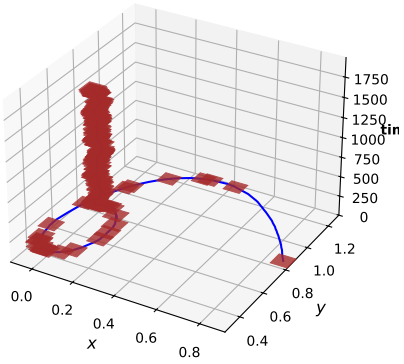


```
#Choose from A4a, A4b, A4c, A4d
python artEval.py --fig=A4a
```

NOTE: This will generate the image showing the plot for the other variable first, followed by fig. A.4(a). Be sure to close the figure window to view the next figure. This is similar for all figures A.4(a)-(d).




```
python artEval.py --fig=B5
```



2. **To recreate figs. B.7(a)-(d) perform the following steps:**

   (a) Make sure you are in location `/path/to/Posto`

   ```
   cd /path/to/Posto
   ```

   (b) Run the following command if the location is not added to the `/.bashrc`

   ```
   export POSTO_ROOT_DIR=/path/to/Posto
   ```

   (c) Run the `artEvalNN.py` with the figure number as the argument to recreate the desired figure

   ```
   #Choose from B6a, B6b, B6d, B6c
   python artEvalNN.py --fig=B6a
   ```