Backdoor Pony

Evaluating backdoor attacks and defenses in different domains

Mercier, Arthur; Smolin, Nikita; Sihlovec, Oliver; Koffas, Stefanos; Picek, Stjepan

DOI
10.1016/j.softx.2023.101387

Publication date
2023

Document Version
Final published version

Published in
SoftwareX

SoftwareX 22 (2023) 101387

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Backdoor Pony: Evaluating backdoor attacks and defenses in different domains

Arthur Mercier [a,b,*,1], Nikita Smolin [b,1], Oliver Sihlovec [b,1], Stefanos Koffas [b], Stjepan Picek [c,b]

[a] Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands
[b] Cybersecurity Group, Delft University of Technology, The Netherlands
[c] Digital Security Group, Radboud University, The Netherlands

ARTICLE INFO

Article history:
Received 12 December 2022
Received in revised form 20 March 2023
Accepted 28 March 2023

Keywords:
Neural networks
Backdoor attacks
Backdoor defenses
Framework

ABSTRACT

Outsourced training and crowdsourced datasets lead to a new threat for deep learning models: the backdoor attack. In this attack, the adversary inserts a secret functionality in a model, activated through malicious inputs. Backdoor attacks represent an active research area due to diverse settings where they represent a real threat. Still, there is no framework to evaluate existing attacks and defenses in different domains. Only a few toolboxes have been implemented, but most of them focus on computer vision and are difficult to use. To bridge this gap, we implement Backdoor Pony, a framework for evaluating attacks and defenses in different domains through a user-friendly GUI.

Code metadata

Current code version	v2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00419
Permanent link to Reproducible Capsule	–
Legal Code License	Copyright 2022 AISyLab
Code versioning system used	git
Software code languages, tools, and services used	Python, JavaScript, Docker, Vue
Compilation requirements, operating environments & dependencies	Linux
If available Link to developer documentation/manual	
Support email for questions	info.backdoorpony@gmail.com

1. Motivation and significance

Neural Networks have proven to work more accurately and considerably faster than humans in several areas, such as facial recognition [1], stock market predictions [2], and detection of "fake news" on social media [3]. For this reason, they are widely used and recently deployed on critical applications like autonomous driving [4]. Thus, we must ensure their secure operation and deployment in the real world. The backdoor attack is a threat that has recently emerged and can have dire consequences as the trained model can be manipulated to the adversary's benefit [5]. It is, therefore, essential to develop defenses to identify and counter these attacks. Up to date, only a few tools facilitate the evaluation of such attacks and defenses. However, most of them target only computer vision [6–8], or only offer a command line interface [8,9]. As such, they are not convenient for a broader audience.

To bridge this gap, we implemented Backdoor Pony.[2] Backdoor Pony is a web-based application containing a GUI through which users can evaluate backdoor attacks and defenses in different application domains. Its GUI is user-friendly, making research more accessible to a large audience. The user controls various hyperparameters for the attacks and defenses through the GUI without the need to build complex pipelines. After each experiment, various plots are shown to help users understand the

* Corresponding author.
E-mail addresses: info.backdoorpony@gmail.com (Arthur Mercier), smolin.projects@gmail.com (Nikita Smolin), O.Sihlovec@student.tudelft.nl (Oliver Sihlovec), s.koffas@tudelft.nl (Stefanos Koffas), stjepan.picek@ru.nl (Stjepan Picek).

[1] Equal contribution.

[2] https://github.com/Rezonansce/backdoorponyv2

https://doi.org/10.1016/j.softx.2023.101387

Table 1
A comparison of Backdoor Pony and the state-of-the-art.

Tool	Image	Text	Audio	Graph	GUI	#Attacks	#Defenses
BackdoorBox [6]	x					12	7
TrojanZoo [8]	x					8	14
OpenBackdoor [10]		x				12	4
BackdoorBench [7]	x					8	9
Backdoor Toolbox [11]	x					13	10
Backdoor Pony	x	x	x	x	x	6	5

performance of their experiments. Our tool can follow the recent developments in this field, as it can be extended with new attacks and defenses in all domains. To conclude, our tool could lower the barrier for new researchers and establish a common point of reference for backdoor attacks and defenses.

The main difference between Backdoor Pony and the state-of-the-art can be seen in Table 1. We see that only OpenBackdoor [10] targets text applications, and the rest are focused only on computer vision [6–8,11]. Backdoor Pony is the first tool to include audio and graph data. It is also the first to offer a GUI. While the existing tools have implemented more attacks and defenses, we do not consider this a major limitation, as we can use our extension API to extend our tool.

The implemented attacks and defenses by the state-of-the-art are shown in Table 2 and Backdoor Pony's supported attacks and defenses in Table 3. In image classification, we chose patch-based triggers as they are the most popular. In particular, we chose the Badnet [5] and the clean-label attack [12]. From Table 2 we see that these attacks are widely supported by the state-of-the-art. Even though it is very popular, we did not use the Blended attack [13] as its performance is very similar to the Badnet attack. We chose only three different defenses that target different parts of the pipeline (input sanitization, backdoor removal, or monitoring) to avoid overlaps in our evaluations. For text classification, we implemented BadNL attack [14] as it supports various triggers (character, word, or sentence). As a defense, we chose ONION [15] as it is one of the first defenses for textual backdoors. In graph and audio domains, we picked the most popular attacks from the literature. In the graph domain, to the best of our knowledge, there is not yet a domain-specific defense, and for audio, we chose STRIP-VITA [16]. Our tool supports fewer defenses and attacks as we initially aimed for usability and application variety. Now that many domains are supported, we can easily extend our tool with additional attacks and defenses through our extension API.

Our contributions can be summarized as follows:

- We implemented Backdoor Pony, the first user-friendly web-based framework that allows smooth and fast evaluation of backdoor attacks and defenses.
- Backdoor Pony is the first tool that supports different applications like computer vision, text classification, audio recognition, and graph neural networks.
- Like other tools, Backdoor Pony is extensible with arbitrary attacks and defenses in all four provided domains.

2. Backdoor attacks on neural networks

Recent trends like outsourced training, machine learning as a service (MLaaS), and crowdsourced datasets introduced a new threat for deep learning models, the backdoor attack [5]. In this attack, an adversary injects a secret functionality in a trained model that is activated through malicious inputs. In any other input, the model behaves normally to avoid raising any suspicions. This attack can be mounted through data poisoning [5], code poisoning [22], or weight poisoning [75]. Currently, we support only data poisoning attacks as they are the most popular and present the large majority of the related works.

In a classifier, this secret functionality may be targeted misclassifications that happen when the model's input contains a specific property, the trigger. For example, a stop sign with a yellow post-it note (trigger) can be misclassified as a speed limit from a backdoored model deployed in an autonomous car [5]. In the recent literature, various attacks have been designed. The most targeted application is computer vision [76], but backdoor attacks have already been implemented against text [14,77], audio [71,78], and graph neural networks [74,79].

Defending against backdoor attacks is very difficult as there is no method to exhaustively verify what a model has learned. New attacks that bypass existing countermeasures are continuously developed in the related literature, making this whole field a cat-and-mouse game between the attackers and the defenders [75]. There are various countermeasures already implemented that can be divided into different categories based on the assumptions used. Some of them remove the triggers from the inputs without knowing if the model is poisoned [80] or search for poisoned samples in the training dataset [40,41]. Other countermeasures inspect the trained model offline to reverse engineer a trigger pattern and unlearn the backdoor [44,45], or monitor the operation of a model and raise alarms when there is suspicious behavior [42]. Some of these countermeasures are white-box and assume the defender has full access to the model and its training data [40,41], or in the opposite case, black-box [42,46]. This field is generally very active, and new attacks and defenses are published regularly, making it very challenging for researchers to keep up-to-date.

2.1. Metrics

We use the attack success rate (ASR) and the clean accuracy drop (CAD) to measure the backdoor's effectiveness. ASR is the percentage of the successfully activated backdoors over a number of tries and should be as high as possible. The clean accuracy drop shows the effect of the backdoor on the original task and should be as small as possible to avoid raising any suspicions. To see the effectiveness of a defense, we can either show the ASR's drop after the defense's implementation or a confusion matrix with statistics regarding poisoned and clean data samples. In this case, we have the following categories:

- **True/False positive rate**: the probability the classifier correctly/incorrectly identifies a poisoned input.
- **True/False negative rate**: the probability the classifier correctly/incorrectly identifies a benign input.

3. Software description

Backdoor Pony uses a client–server architecture, shown in *Fig.* 1. The front-end consists of a web application with a GUI that controls the tool. The web application is written in javascript and mainly uses the Vue library. The back-end server is where all the services are run. The server is written in python and uses the Flask framework. Its `/src` can be found in the `server` folder, while the front-end `/src` can be found in the `gui` folder. The server and front-end use REST-API to communicate.

Table 2
The implemented attacks and defenses by the state-of-the-art.

Tool	Type	Attack	Defense
BackdoorBox	image	Badnets [5], Blended [13], Refool [17], Clean-label [12], TUAP [18], SleeperAgent [19], ISSBA [20], WaNet [21], Blind [22], Input-aware [23], PhysicalBA [24], LIRA [25]	3-layer Autoencoder [26], ShrinkPad [24], Fine-tuning, Fine-Pruning [27], MCR [28], NAD [29], ABL [30]
TrojanZoo	image	Badnets [5], ESB [31], TNN [32], Refool [17], Blended [13], LB [33], ABE [34], IMC [35]	Randomized Smoothing [36], DownUp [37], Adversarial Retraining [38], Manifold-projection [39], AC [40], Spectral-signatures [41], STRIP [42], NEO [43], Fine-Pruning [27], Neural Cleanse [44], DeepInspect [45], TABOR [46], Neuron Inspect [47], ABS
OpenBackdoor	text	Badnets [5], AddSent [48], SynBkd [49], StyleBkd [50], POR [51], TrojanLM [52], SOS [53], LWP [54], EP [55], NeuBA [56], LWS [57], RIPPLES [58]	ONION [15], STRIP-VITA [16], RAP [59], BKI [60]
BackdoorBench	image	Badnets [5], Blended [13], Clean-label [12], SIG [61], Low-frequency [62], ISSBA [20], Input-aware [23], WaNet [21]	Fine-tuning, Fine-pruning [27], NAD [29], Neural Cleanse [44], ANP [63], ABL [30], DBD [64] Activation Clustering [40], Spectral-signatures [41],
Backdoor Toolbox	image	Badnets [5], Badnets all-to-all [5], Blended [13], TNN [32], Clean-label [12], Input-aware [23], SIG [61], ISSBA [20], WaNet [21], Refool [17], TaCT [65], Adaptive [66], SleeperAgent [19]	SCAn [65], Activation clustering [40], Spectral-signatures [41], SPECTRE [67], STRIP [42], Sentinet [68], Low-frequency [62], Neural Cleanse [44], Fine-pruning [27], ABL [30]

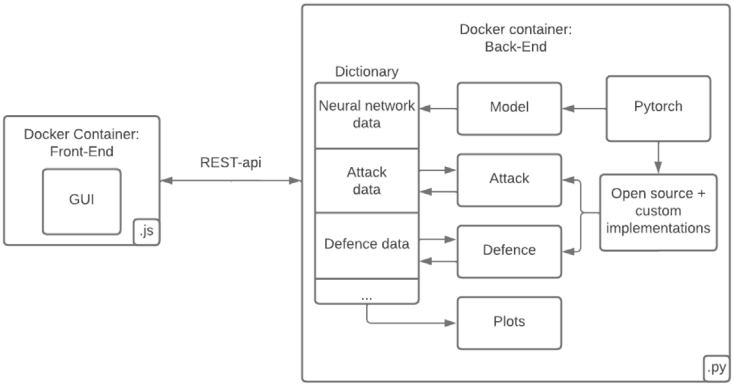

Fig. 1. Software architecture overview.

The server functions in parallel to the GUI interface. The highest level of the server's architecture consists of a runner that keeps track of the current state of the front-end application. Its job is to execute the instructions in the incoming REST requests on the appropriate service. The runner also ensures that the requests are made in the correct order, making it impossible, for example, to execute an attack if the model has not been chosen.

The different services executed in order are "model", "attack", "defense", and "plot". All the hyperparameters and the resulting data of the services are stored in a python dictionary used to pass information through the framework. The interaction between the services and the dictionary can be seen in *Fig.* 1. After the user makes a selection from the available models and updates its hyperparameters, the model is trained on one of the datasets included in the framework. After the model is trained, the user selects and runs the attacks and defenses. Some of our implemented neural network models use PyTorch, and a few of our attacks and defenses also use the Adversarial Robustness Toolbox [9]. We implemented the rest of them from scratch using PyTorch. After applying the attacks and the defenses, we

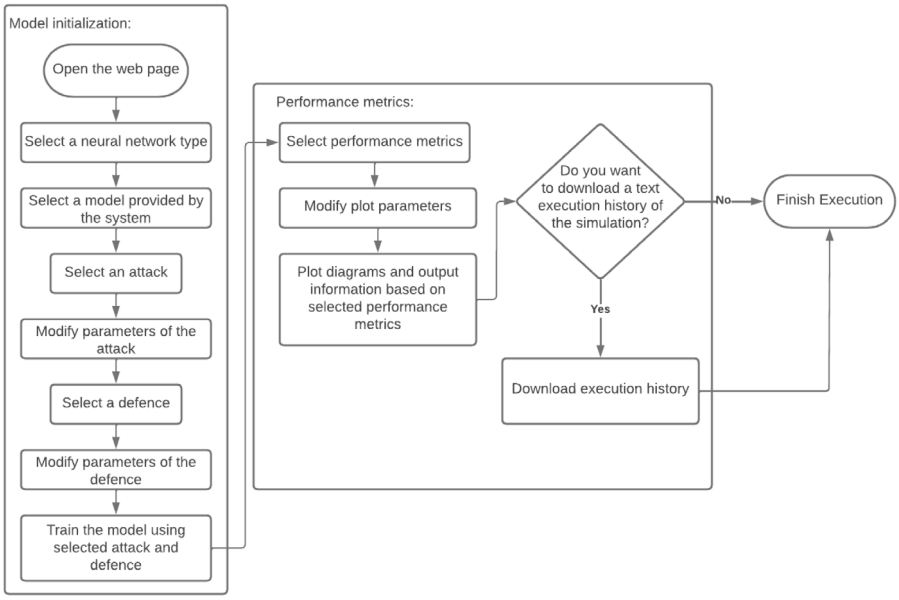

Fig. 2. General workflow of the application.

Table 3
The implemented attacks and defenses per data type.

Model	Attack	Defenses
Image	Badnet [5], Clean-label [12]	De-trigger Autoencoder [69], IBAU [70], STRIP [42]
Audio	BAASV [71]	STRIP-VITA [16]
Text	BadNl [72]	ONION [15]
Graph	Zaixizhang [73], GTA [74]	N/A

save all the corresponding hyperparameters and results into the dictionary. This information is used for the plots in the last step of our tool. The available metrics differ based on what attacks and defenses were selected. As stated before, the model service stores its results in the dictionary before the attacks and defenses are run. In this way, the user can easily compare the network's performance before or after the attack or the defense. Once the plot service has computed all the statistics, they are sent in a specific format to the GUI, which will generate the corresponding plots.

Backdoor Pony runs with Docker. The server and the front-end run in a separate container. Docker is also used to connect to the integrated GPU support. Running "docker-compose up" with an existing PyTorch image and CUDA/NVIDIA drivers will automatically expose all available GPUs in the system to the container. The user may also select which specific GPUs will be used. One of our focuses for Backdoor Pony was to maintain high code quality. We have achieved this by implementing engineering practices, creating unit tests, and comprehensive documentation. This was done to ease future implementation or the integration of new attacks and defenses into Backdoor Pony.

4. Illustrative example

In this section, we first show a high-level overview of the framework's operating flow (*Fig.* 2) and then provide an illustrative example of its usage. As we show in *Fig.* 2, the user selects a dataset and a model, an attack (optionally), and a defense (optionally) and then proceeds to the training. When the training finishes, the user selects the plots needed and, optionally, downloads the execution history. Our tool has information buttons – the "*i*" letter enclosed in a red circular background – to facilitate its usage. When this button is clicked, a pop-up window shows all relevant information about that field.

4.1. Dataset selection

First we need to select a dataset. The selection screen will appear after the user clicks on the **Start** button from the home page. Here, a set of available datasets is presented (*Fig.* 3). Only one can be chosen at a time. Image, audio, text, and graph are the four types of available data categories, each having a unique icon. The right side of the screen displays a list of configurable hyperparameters for the model that will be trained on the selected dataset. In *Fig.* 3, the user chooses the full CIFAR10 dataset (50000 training images), SGD as the model's optimizer, and a learning rate of 0.001.

When a choice has been made, the server loads a neural network that will be used to classify the data. If the framework has previously used this dataset, and the hyperparameters were not changed from their default values, a pre-trained model will be loaded in the current execution. Otherwise, the network is trained with the data after clicking the **Continue** button.

4.2. Choosing the attack and defense

Next, the user selects an attack and a defense. Similarly to the datasets, attacks, and defenses are type-specific. Thus, only the attacks and defenses that match the dataset's type are shown for selection on the screen (*Fig.* 4). Each attack and defense has a set of hyperparameters appearing on the right side of the page after selecting the desired mechanism (*Fig.* 4). Multiple values can be typed in for a single field, separated by commas which will trigger an execution of the simulation for each unique combination of values.

Additionally, the user can skip running a defense for the current configuration by selecting the **None** label in the list of defenses. This choice impacts the available metrics after the attack has been completed. Once the user has chosen the means of attack, defense, and hyperparameters, clicking the **Execute** button – placed in the bottom right of the page (*Fig.* 4) – will start applying the attack and defense on the neural network.

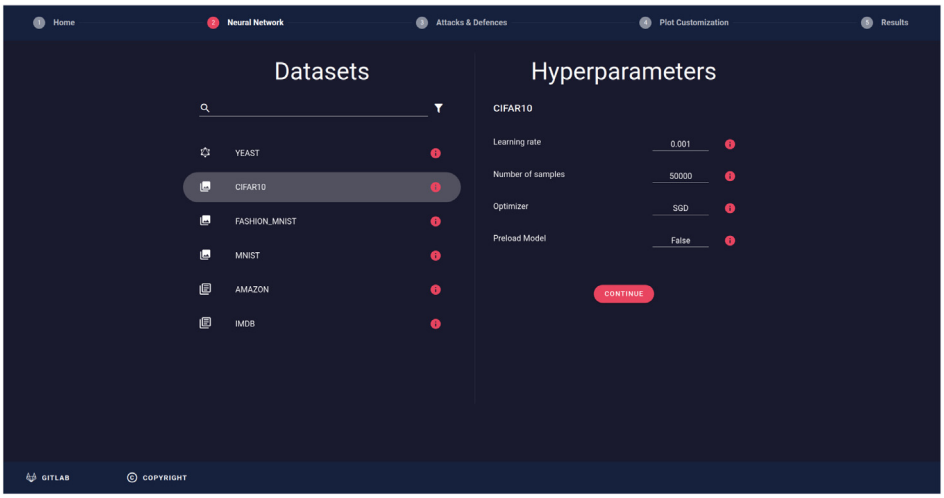

Fig. 3. Dataset selection screen.

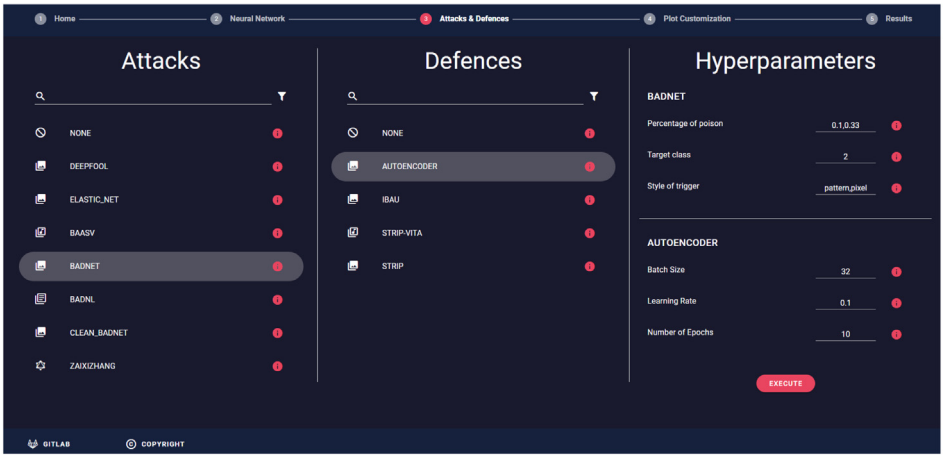

Fig. 4. Attack (parameter selection upper right window) and Defense (parameters selection lower right window) selection screen.

4.3. Metrics configuration

The metrics' settings page will appear once the previous step has finished its execution. The tool can generate multiple plots that indicate the model's performance or the attack's and defense's effectiveness. These plots include the model's accuracy, the ASR, the CAD, the true and false positive rates, and the true and false negative rates.

To create a plot, the user clicks on the "**+**" sign of the desired metric and specifies the variables for the "***X***" and "***Y***" axes and the fixed hyperparameters selected for the attacks and defenses. An example plot would be the poisoned model's accuracy in the "***Y***" axis and the poisoning rate (the percentage of the poisoned data) in the "***X***" axis. The fixed hyperparameters would be the rest of the options given by the user in the previous screens (target class, style of trigger) as shown in *Fig.* 5. The **Add** button saves the plot configuration and the **Plot** button produces the corresponding plots. To create these plots, the server feeds clean or poisoned inputs to the model and calculates the chosen metrics.

4.4. Results

Finally, the plots will be presented in the **Results** tab. In this tab (*Fig.* 6), we can see the clean accuracy (leftmost plot), which is the model's accuracy on clean inputs, the ASR (middle plot), which shows the attack's effectiveness, and the clean accuracy drop (rightmost plot), which shows the percentage drop of the clean accuracy by the backdoor in. Besides the metrics, a text file can be downloaded. The file includes details about the dataset, attacks, defenses, and the chosen hyperparameters. This file presents some information about basic user choices in simple sentences and shows more complex options in a `json` format. An example of this file is shown in Listing 1 in Appendix A.

5. Impact

Backdoor Pony is a framework that targets both cutting-edge research and education. The framework's documentation shows how users can extend this tool through a few simple steps and add new attacks and defenses. This extension mechanism will keep Backdoor Pony up to date with all the recent developments in the field. An up-to-date tool will reduce the load of re-implementing existing works from the literature and lead to faster results and more accurate research outcomes. Additionally, Backdoor Pony can run multiple experiments sequentially by accepting a list of values for the experiment's hyperparameters. This speeds up ablation studies allowing users to focus on identifying trends instead of building pipelines. We have already used this tool in our internal research projects, and we believe it can be a valuable contribution to the community. Unlike current tools, e.g., [6–8], Backdoor Pony supports many application domains.

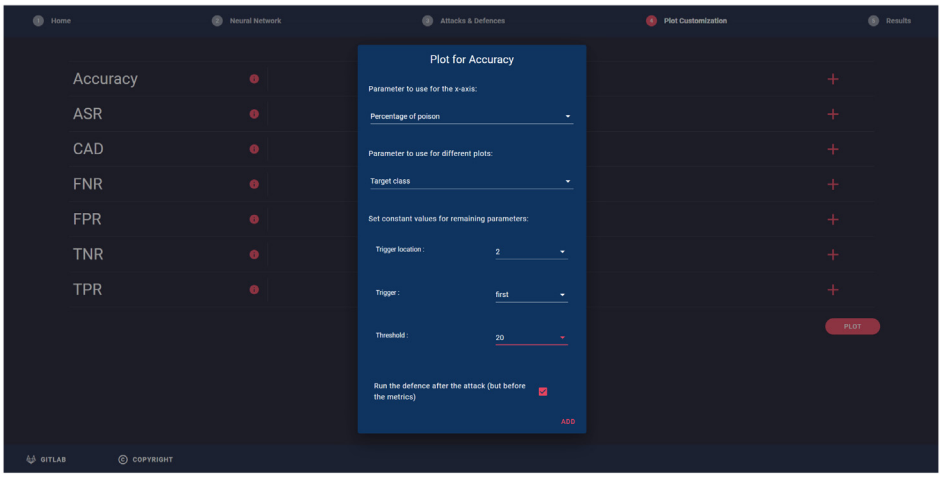

Fig. 5. Network accuracy metric settings.

Fig. 6. Example of resulting plots.

Thus, it can become a common point of reference for experiments among researchers leading to fair comparisons between different attacks and defenses. Finally, its simplicity and user-friendliness will lower the barriers to entry in this field and attract a broader audience. As a result, it can become a valuable educational framework.

6. Limitations and future work

While our tool can be a valuable contribution to the field we believe that there are a few limitations. First, a few popular attacks and defenses are missing. However, we plan to add them in the future allowing researchers to keep up with state-of-the-art without too much effort. Additionally, in this version, custom models and datasets are not supported as we wanted to ensure that our pipeline is functional before allowing arbitrary inputs from the users. In the future, we will add this feature to Backdoor Pony allowing the seamless evaluation of the attacks and the defenses on new data and tasks.

7. Conclusion

In this work, we presented Backdoor Pony which is a framework for evaluating backdoor attacks and defenses in various application domains. To our knowledge, Backdoor Pony is the first framework that supports four application domains (image, text, speech, and graphs) as most of the previous tools focus only on computer vision. It also offers full control of the hyperparameters involved through a user-friendly GUI making it suitable for cutting-edge research and education.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The code repository is open to public access

Acknowledgments

We would like to thank Kristián Gogora and Vlad Murgoci for being an integral part of the development of Backdoor Pony's second version. We would also like to thank Gregor Schram, Dragos Dumitrescu, Anneke Linssen, Nicolas Plas, Daniela Toader for the first version of this tool. We also thank all the researchers who provided open-source attacks and defenses, thus allowing Backdoor Pony to amass a wider catalog.

Appendix A. Configuration file

In this section, we show an example of the generated configuration file. It contains info about the chosen dataset, the model used, the attack and hyperparameters (trigger's position and poisoning rate), the defense and hyperparameters, and the plot configuration.

```
1. Select the IMDB dataset;
2. Select 'use built-in model';
3. Choose the stealthybadnl attack;
```

```
3a. Input [1] for target class;
  Attack parameters:
  {"target_class": {"pretty_name": "Target class", "value": [1]},
  "location": {"pretty_name": "Trigger location",  "value": ["start
     "],                "values": ["start", "middle", "end"]},
  "trigger": {"pretty_name": "Trigger", "value": ["char"],
  "values": ["char", "word", "sentence"]},
  "poison_percent": {"pretty_name": "Percentage of poison", "value
     ": [0.1],                  "minimum": 0, "maximum": 1, "
     isValid": true}}
4. Choose the onion defense;
Defense parameters:
{"threshold": {"pretty_name": "Threshold", "value": [20]}}
5. Press execute.
6. Metrics:
{"metrics0": {"metric": "Accuracy", "x_axis": "Percentage of poison
     ",
             "plot": "Target class",
             "graph": {"1": {"points": [{"x": 0.1, "y": 50.8}],
                             "name": 1}}},
 "metrics1": {"metric": "Accuracy", "x_axis": "Percentage of poison
     ",
             "plot": "Target class",
             "graph": {"1": {"points": [{"x": 0.1, "y": 52.8}],
                             "name": 1}}}}
```

Listing 1: Example configuration file

References

[1] Phillips PJ, O'toole AJ. Comparison of human and computer performance across face recognition experiments. Image Vis Comput 2014;32(1):74–85.

[2] Chen S, He H. Stock prediction using convolutional neural network. IOP Conf Ser: Mater Sci Eng 2018;435(1):012026. http://dx.doi.org/10.1088/1757-899X/435/1/012026.

[3] Yang Y, Zheng L, Zhang J, Cui Q, Li Z, Yu PS. TI-CNN: Convolutional neural networks for fake news detection. 2018, http://dx.doi.org/10.48550/ARXIV.1806.00749, arXiv URL https://arxiv.org/abs/1806.00749.

[4] Bojarski M, Yeres P, Choromanska A, Choromanski K, Firner B, Jackel L, et al. Explaining how a deep neural network trained with end-to-end learning steers a car. 2017, arXiv preprint arXiv:1704.07911.

[5] Gu T, Dolan-Gavitt B, Garg S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. 2017, arXiv preprint arXiv:1708.06733.

[6] Li Y, Ya M, Bai Y, Jiang Y, Xia S-T. BackdoorBox: A Python toolbox for backdoor learning. 2022.

[7] Wu B, Chen H, Zhang M, Zhu Z, Wei S, Yuan D, et al. BackdoorBench: A comprehensive benchmark of backdoor learning. In: NeurIPS 2022 track datasets and benchmarks. 2022.

[8] Pang R, Zhang Z, Gao X, Xi Z, Ji S, Cheng P, et al. TrojanZoo: Towards unified, holistic, and practical evaluation of neural backdoors. In: Proceedings of IEEE european symposium on security and privacy. 2022.

[9] Nicolae M-I, Sinn M, Tran MN, Buesser B, Rawat A, Wistuba M, et al. Adversarial robustness toolbox v1. 0.0. 2018, arXiv preprint arXiv:1807.01069.

[10] Cui G, Yuan L, He B, Chen Y, Liu Z, Sun M. A unified evaluation of textual backdoor learning: Frameworks and benchmarks. In: Proceedings of neurIPS: datasets and benchmarks. 2022.

[11] Tinghao X. Backdoor toolbox. 2022, GitHub URL https://github.com/vtu81/backdoor-toolbox [Online, Accessed: 31 March 2023].

[12] Turner A, Tsipras D, Mdry A. Clean-label backdoor attacks. MIT, URL https://people.csail.mit.edu/madry/lab/cleanlabel.pdf.

[13] Chen X, Liu C, Li B, Lu K, Song D. Targeted backdoor attacks on deep learning systems using data poisoning. 2017, arXiv preprint arXiv:1712.05526.

[14] Chen X, Salem A, Backes M, Ma S, Zhang Y. Badnl: Backdoor attacks against nlp models. In: ICML 2021 workshop on adversarial machine learning. 2021.

[15] Qi F, Chen Y, Li M, Yao Y, Liu Z, Sun M. Onion: A simple and effective defense against textual backdoor attacks. 2020, arXiv preprint arXiv:2011.10369.

[16] Gao Y, Kim Y, Doan BG, Zhang Z, Zhang G, Nepal S, et al. Design and evaluation of a multi-domain Trojan detection method on deep neural networks. IEEE Trans Dependable Secure Comput 2021;19(4):2349–64.

[17] Liu Y, Ma X, Bailey J, Lu F. Reflection backdoor: A natural backdoor attack on deep neural networks. In: Computer vision–ECCV 2020: 16th european conference. Springer; 2020, p. 182–99.

[18] Zhao S, Ma X, Zheng X, Bailey J, Chen J, Jiang Y-G. Clean-label backdoor attacks on video recognition models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 14443–52.

[19] Souri H, Fowl L, Chellappa R, Goldblum M, Goldstein T. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. 2021.

[20] Li Y, Li Y, Wu B, Li L, He R, Lyu S. Invisible backdoor attack with sample-specific triggers. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 16463–72.

[21] Nguyen A, Tran A. Wanet–imperceptible warping-based backdoor attack. 2021, arXiv preprint arXiv:2102.10369.

[22] Bagdasaryan E, Shmatikov V. Blind backdoors in deep learning models. In: 30th USENIX security symposium. 2021, p. 1505–21.

[23] Nguyen TA, Tran A. Input-aware dynamic backdoor attack. Adv Neural Inf Process Syst 2020;33:3454–64.

[24] Li Y, Zhai T, Jiang Y, Li Z, Xia S-T. Backdoor attack in the physical world. 2021, arXiv preprint arXiv:2104.02361.

[25] Doan K, Lao Y, Zhao W, Li P. Lira: Learnable, imperceptible and robust backdoor attacks. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 11966–76.

[26] Liu Y, Xie Y, Srivastava A. Neural Trojans. In: 2017 IEEE international conference on computer design. IEEE; 2017, p. 45–8.

[27] Liu K, Dolan-Gavitt B, Garg S. Fine-pruning: Defending against backdooring attacks on deep neural networks. In: Research in attacks, intrusions, and defenses: 21st international symposium. Springer; 2018, p. 273–94.

[28] Zhao P, Chen P-Y, Das P, Ramamurthy KN, Lin X. Bridging mode connectivity in loss landscapes and adversarial robustness. 2020, arXiv preprint arXiv:2005.00060.

[29] Li Y, Lyu X, Koren N, Lyu L, Li B, Ma X. Neural attention distillation: Erasing backdoor triggers from deep neural networks. 2021, arXiv preprint arXiv:2101.05930.

[30] Li Y, Lyu X, Koren N, Lyu L, Li B, Ma X. Anti-backdoor learning: Training clean models on poisoned data. Adv Neural Inf Process Syst 2021;34:14900–12.

[31] Tang R, Du M, Liu N, Yang F, Hu X. An embarrassingly simple approach for trojan attack in deep neural networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. 2020, p. 218–28.

[32] Liu Y, Ma S, Aafer Y, Lee W, Zhai J, Wang W, et al. Trojaning attack on neural networks. In: 25th annual network and distributed system security symposium. The Internet Society; 2018.

[33] Yao Y, Li H, Zheng H, Zhao BY. Latent backdoor attacks on deep neural networks. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. 2019, p. 2041–55.

[34] Shokri R, et al. Bypassing backdoor detection algorithms in deep learning. In: 2020 IEEE european symposium on security and privacy. IEEE; 2020, p. 175–83.

[35] Pang R, Shen H, Zhang X, Ji S, Vorobeychik Y, Luo X, et al. A tale of evil twins: Adversarial inputs versus poisoned models. In: Proceedings of the 2020 ACM SIGSAC conference on computer and communications security. 2020, p. 85–99.

[36] Cohen J, Rosenfeld E, Kolter Z. Certified adversarial robustness via randomized smoothing. In: International conference on machine learning. PMLR; 2019, p. 1310–20.

[37] Xu W, Evans D, Qi Y. Feature squeezing: Detecting adversarial examples in deep neural networks. 2017, arXiv preprint arXiv:1704.01155.

[38] Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A. Towards deep learning models resistant to adversarial attacks. 2017, arXiv preprint arXiv:1706.06083.

[39] Meng D, Chen H. Magnet: A two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 2017, p. 135–47.

[40] Chen B, Carvalho W, Baracaldo N, Ludwig H, Edwards B, Lee T, et al. Detecting backdoor attacks on deep neural networks by activation clustering. 2018, arXiv preprint arXiv:1811.03728.

[41] Tran B, Li J, Madry A. Spectral signatures in backdoor attacks. Adv Neural Inf Process Syst 2018;31.

[42] Gao Y, Xu C, Wang D, Chen S, Ranasinghe DC, Nepal S. STRIP: A defence against Trojan attacks on deep neural networks. 2019, http://dx.doi.org/10.48550/ARXIV.1902.06531, arXiv URL https://arxiv.org/abs/1902.06531.

[43] Udeshi S, Peng S, Woo G, Loh L, Rawshan L, Chattopadhyay S. Model agnostic defence against backdoor attacks in machine learning. IEEE Trans Reliab 2022;71(2):880–95.

[44] Wang B, Yao Y, Shan S, Li H, Viswanath B, Zheng H, et al. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: 2019 IEEE symposium on security and privacy. IEEE; 2019, p. 707–23.

[45] Chen H, Fu C, Zhao J, Koushanfar F. DeepInspect: A black-box Trojan detection and mitigation framework for deep neural networks. In: IJCAI, Vol. 2. 2019, p. 8.

[46] Guo W, Wang L, Xing X, Du M, Song D. Tabor: A highly accurate approach to inspecting and restoring Trojan backdoors in ai systems. 2019, arXiv preprint arXiv:1908.01763.

[47] Huang X, Alzantot M, Srivastava M. Neuroninspect: Detecting backdoors in neural networks via output explanations. 2019, arXiv preprint arXiv:1911.07399.
[48] Dai J, Chen C, Li Y. A backdoor attack against lstm-based text classification systems. IEEE Access 2019;7:138872–8.
[49] Qi F, Li M, Chen Y, Zhang Z, Liu Z, Wang Y, et al. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. 2021, arXiv preprint arXiv:2105.12400.
[50] Qi F, Chen Y, Zhang X, Li M, Liu Z, Sun M. Mind the style of text! adversarial and backdoor attacks based on text style transfer. 2021, arXiv preprint arXiv:2110.07139.
[51] Shen L, Ji S, Zhang X, Li J, Chen J, Shi J, et al. Backdoor pre-trained models can transfer to all. 2021, arXiv preprint arXiv:2111.00197.
[52] Zhang X, Zhang Z, Ji S, Wang T. Trojaning language models for fun and profit. In: 2021 IEEE european symposium on security and privacy. IEEE; 2021, p. 179–97.
[53] Yang W, Lin Y, Li P, Zhou J, Sun X. Rethinking stealthiness of backdoor attack against nlp models. In: Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers). 2021, p. 5543–57.
[54] Li L, Song D, Li X, Zeng J, Ma R, Qiu X. Backdoor attacks on pre-trained models by layerwise weight poisoning. 2021, arXiv preprint arXiv:2108.13888.
[55] Yang W, Li L, Zhang Z, Ren X, Sun X, He B. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models. 2021, arXiv preprint arXiv:2103.15543.
[56] Zhang Z, Xiao G, Li Y, Lv T, Qi F, Liu Z, et al. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. Mach Intell Res 2023;1–14.
[57] Qi F, Yao Y, Xu S, Liu Z, Sun M. Turn the combination lock: Learnable textual backdoor attacks via word substitution. 2021, arXiv preprint arXiv:2106.06361.
[58] Kurita K, Michel P, Neubig G. Weight poisoning attacks on pre-trained models. 2020, arXiv preprint arXiv:2004.06660.
[59] Yang W, Lin Y, Li P, Zhou J, Sun X. Rap: Robustness-aware perturbations for defending against backdoor attacks on nlp models. 2021, arXiv preprint arXiv:2110.07831.
[60] Chen C, Dai J. Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification. Neurocomputing 2021;452:253–62.
[61] Barni M, Kallas K, Tondi B. A new backdoor attack in cnns by training set corruption without label poisoning. In: 2019 IEEE international conference on image processing. IEEE; 2019, p. 101–5.
[62] Zeng Y, Park W, Mao ZM, Jia R. Rethinking the backdoor attacks' triggers: A frequency perspective. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 16473–81.
[63] Wu D, Wang Y. Adversarial neuron pruning purifies backdoored deep models. Adv Neural Inf Process Syst 2021;34:16913–25.
[64] Huang K, Li Y, Wu B, Qin Z, Ren K. Backdoor defense via decoupling the training process. 2022, arXiv preprint arXiv:2202.03423.
[65] Tang D, Wang X, Tang H, Zhang K. Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection. In: USENIX security symposium. 2021, p. 1541–58.
[66] Qi X, Xie T, Li Y, Mahloujifar S, Mittal P. Revisiting the assumption of latent separability for backdoor defenses. In: The eleventh international conference on learning representations. 2023.
[67] Hayase J, Kong W, Somani R, Oh S. Spectre: Defending against backdoor attacks using robust statistics. In: International conference on machine learning. PMLR; 2021, p. 4129–39.
[68] Chou E, Tramer F, Pellegrino G. Sentinet: Detecting localized universal attacks against deep learning systems. In: 2020 IEEE security and privacy workshops. IEEE; 2020, p. 48–54.
[69] Kwon H. Defending deep neural networks against backdoor attack by using de-trigger autoencoder. IEEE Access 2021;1. http://dx.doi.org/10.1109/ACCESS.2021.3086529.
[70] Zeng Y, Chen S, Park W, Mao ZM, Jin M, Jia R. Adversarial unlearning of backdoors via implicit hypergradient. 2021, http://dx.doi.org/10.48550/ARXIV.2110.03735, arXiv URL https://arxiv.org/abs/2110.03735.
[71] Zhai T, Li Y, Zhang Z, Wu B, Jiang Y, Xia S-T. Backdoor attack against speaker verification. In: ICASSP 2021-2021 IEEE international conference on acoustics, speech and signal processing. IEEE; 2021, p. 2560–4.
[72] Chen X, Salem A, Chen D, Backes M, Ma S, Shen Q, et al. BadNL: Backdoor attacks against NLP models with semantic-preserving improvements. In: Annual computer security applications conference. ACM; 2021, http://dx.doi.org/10.1145/3485832.3485837.
[73] Zhang Z, Jia J, Wang B, Gong NZ. Backdoor attacks to graph neural networks. 2020, http://dx.doi.org/10.48550/ARXIV.2006.11165, arXiv URL https://arxiv.org/abs/2006.11165.
[74] Xi Z, Pang R, Ji S, Wang T. Graph backdoor. In: 30th USENIX security symposium. USENIX Association; 2021, p. 1523–40, URL https://www.usenix.org/conference/usenixsecurity21/presentation/xi.
[75] Hong S, Carlini N, Kurakin A. Handcrafted backdoors in deep neural networks. 2021, arXiv preprint arXiv:2106.04690.
[76] Gao Y, Doan BG, Zhang Z, Ma S, Zhang J, Fu A, et al. Backdoor attacks and countermeasures on deep learning: A comprehensive review. 2020, arXiv preprint arXiv:2007.10760.
[77] Qi F, Chen Y, Zhang X, Li M, Liu Z, Sun M. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In: Proceedings of the 2021 conference on empirical methods in natural language processing. 2021, p. 4569–80.
[78] Koffas S, Xu J, Conti M, Picek S. Can you hear it? Backdoor attacks via ultrasonic triggers. In: Proceedings of the 2022 ACM workshop on wireless security and machine learning. New York, NY, USA: Association for Computing Machinery; 2022, p. 57–62. http://dx.doi.org/10.1145/3522783.3529523.
[79] Xu J, Wang R, Koffas S, Liang K, Picek S. More is better (mostly): On the backdoor attacks in federated graph neural networks. 2022, arXiv preprint arXiv:2202.03195.
[80] Doan BG, Abbasnejad E, Ranasinghe DC. Februus: Input purification defense against Trojan attacks on deep neural network systems. In: Annual computer security applications conference. 2020, p. 897–912.