A Systematic Evaluation of Backdoor Attacks in Various Domains

Koffas, Stefanos; Tajalli, Behrad; Xu, Jing; Conti, Mauro; Picek, Stjepan

DOI
10.1007/978-3-031-40677-5_20

Publication date
2023

Document Version
Final published version

Published in
Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing

Citation (APA)
Koffas, S., Tajalli, B., Xu, J., Conti, M., & Picek, S. (2023). A Systematic Evaluation of Backdoor Attacks in Various Domains. In *Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing: Use Cases and Emerging Challenges* (pp. 519-552). Springer Nature. https://doi.org/10.1007/978-3-031-40677-5_20

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

A Systematic Evaluation of Backdoor Attacks in Various Domains

Stefanos Koffas, Behrad Tajalli, Jing Xu, Mauro Conti, and Stjepan Picek

1 Introduction

In the last few years, deep learning has become very popular, and it has been applied to a variety of applications like computer vision [29], machine translation [54], speech recognition [18], and game playing [44]. It is also used in safety and security-critical applications like autonomous driving [12], malware detection [8], biometric-based user authentication [6], and side-channel analysis [40]. Such systems commonly need large datasets to train reliable models that generalize well and perform adequately with unseen data. However, large datasets are often scrapped from untrusted sources on the web [1, 11]. Additionally, the hardware needed to train such models can be very expensive and is not always available to developers who want to embed some machine learning functionality into their applications. Thus, a new programming paradigm has emerged: Machine Learning

S. Koffas · J. Xu
Cybersecurity Group, Delft University of Technology, Delft, The Netherlands
e-mail: s.koffas@tudelft.nl; j.xu-8@tudelft.nl

B. Tajalli
Digital Security Group, Radboud University, Nijmegen, The Netherlands
e-mail: hamidreza.tajalli@ru.nl

M. Conti
Cybersecurity Group, Delft University of Technology, Delft, The Netherlands

SPRITZ Security and Privacy Research Group, University of Padua, Padua, Italy
e-mail: mauro.conti@unipd.it

S. Picek (✉)
Cybersecurity Group, Delft University of Technology, Delft, The Netherlands

Digital Security Group, Radboud University, Nijmegen, The Netherlands
e-mail: stjepan.picek@ru.nl

S. Pasricha, M. Shafique (eds.), *Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing*, https://doi.org/10.1007/978-3-031-40677-5_20

as a Service (MLaaS), made possible by the recent advances in cloud computing. These new trends lead to novel attack vectors that adversaries can exploit.

One of these attack vectors is the backdoor attack [19]. In this attack, an adversary embeds a secret functionality into a trained model, activated only if the model's input contains a specific property (trigger). At the same time, for any input that does not include the trigger, the model behaves as expected to avoid raising any suspicions. Most of the designed attacks in the literature target computer vision applications [31], but recently different applications have been targeted. In particular, backdoor attacks were shown in text classification [5, 9], audio recognition [28, 62], graph data [55, 57], federated learning [3, 45], and reinforcement learning [58]. A backdoor attack can be dangerous as machine learning is used in many security-related applications. In [19], the authors showed that a stop sign with a small post-it note could be identified as a speed limit by a compromised autonomous vehicle with serious consequences to its passengers and pedestrians. AI-enabled applications like spam-filtering [37], speaker identification [62], or malware detection [42] could also be bypassed if the model used contains a backdoor. Thus, backdoor attacks pose a serious threat, and it is required to understand the limits of such attacks to provide better defenses.

This work explores the effects of various trigger characteristics on the backdoor attack. In particular, we implement backdoor attacks with triggers of varying sizes, positions, and poisoning rates and apply them to four different domains (image, text, sound, and graph data). With it, we aim to better understand backdoor attacks and find common properties among different domains.

In [47], the authors claimed that the backdoor attack becomes ineffective when the adversary cannot alter the training labels and is forced to poison only samples from the target class. In this case, the model cannot learn a strong connection between the trigger and the target class as more substantial features from the target class are learned. This behavior is reasonable and well justified but only supported by one experiment with the CIFAR10 dataset. Here, we aim to test this claim in image classification but also in different domains, like text and sound classification.

Our contributions are:

- We run extensive experiments in different application domains (image, text, audio, and graph data) and systematically evaluate the effect of various trigger characteristics on the backdoor attack.
- We investigate two different backdoor attacks in each application and verify that the clean-label attack is not very effective as it may require large poisoning rates to achieve a high attack success rate. However, this attack could work by choosing more effective triggers without changing the poisoning rate.
- We show that in most cases, the backdoor's effectiveness increases as the trigger size increases.

2 Background

2.1 Computer Vision

Today, the computer vision domain covers diverse use cases and concepts within, ranging from capturing raw data to image pattern extraction and interpreting information from those images. It is mostly a combination of concepts, ideas, and techniques of pattern recognition, digital image processing, artificial intelligence (AI), and computer graphics [53]. Computer vision aims to provide the capability for a system to identify and perceive the visual world in the same way as human vision does. Recently, by applying AI techniques, including deep neural networks, the machines even outperformed humans in several tasks [13].

Nowadays, there are multiple applications of computer vision in our daily life, e.g., weather prediction, medical cases, sports and entertainment, industry and production lines, and human-computer interaction [24, 25, 36, 46, 49, 51, 60]. While the use cases and applications are becoming broader and more prevalent in our everyday lives, security issues regarding the techniques and algorithms are also becoming a significant challenge to deal with.

2.2 Natural Language Processing

Natural language processing (NLP) is at the intersection of computational linguistics, computer science, and artificial intelligence. It aims to make machines that understand human language and reason about it. NLP is an umbrella term that covers many different applications that deal with human language in both spoken and written formats. Applications that belong to natural language processing are, among others, speech recognition, speaker identification, question answering, text sentiment analysis, hate speech detection, natural language generation (speech-to-text and text-to-speech models), spam detection, and text translation. Initially, NLP was based on static rules, but now it uses deep learning for most tasks [23].

Recent advances in NLP have led to very efficient human-computer interfaces that have been broadly deployed. Virtual assistants like Siri and Google assistant and popular IoT devices like Amazon Alexa have been widely used with great success. However, such applications open up new attack vectors that put the user's security and privacy at risk. Therefore, before their wide adoption in the industry, we must ensure that such systems work securely.

2.3 Graph Data

Many real-world applications can be modeled as graphs, such as social networks, gene interactions, and transport networks. Similar to the great success of deep learning models in, e.g., image classification and natural language processing, deep graph models (graph neural networks—GNNs) have also achieved promising performance in processing graph data for different tasks, e.g., the graph classification task and node classification task.

Graph Neural Networks (GNNs) GNNs take a graph G as an input, including its structure information and node features, and learn a representation vector (embedding) for each node $v \in G$, z_v, or the entire graph, z_G. Modern GNNs follow a neighborhood aggregation strategy, where one iteratively updates the representation of a node by aggregating representations of its neighbors. After k iterations of aggregation, a node's representation captures both structure and feature information within its k-hop network neighborhood. Formally, the k-th layer of a GNN is (e.g., GCN [26], GraphSAGE [20], and GAT [48]):

$$Z^{(k)} = AGGREGATE(A, Z^{(k-1)}; \theta^{(k)}), \tag{1}$$

where $Z^{(k)}$ are the node embeddings in the matrix form computed after the k-th iteration and the $AGGREGATE$ function depends on the adjacency matrix A, the trainable parameters $\theta^{(k)}$, and the previous node embeddings $Z^{(k-1)}$. $Z^{(0)}$ is initialized as G's node features.

For the node classification task, the node representation $Z^{(k)}$ of the final iteration is used for prediction. For the graph classification task, the READOUT function pools the node embeddings from the final iteration K:

$$z_G = READOUT(Z^{(K)}). \tag{2}$$

READOUT can be a simple permutation invariant function such as summation or a more sophisticated graph-level pooling function [59, 63].

Graph-Level Classification Graph-level classification aims to predict the class label(s) for an entire graph [63]. The end-to-end learning for this task can be realized using graph convolutional layers and readout layers. While graph convolutional layers are responsible for extracting high-level node representations, the readout layer collapses node representations of each graph into a graph representation. By applying a multilayer perceptron and a Softmax layer to graph representations, one can build an end-to-end framework for graph classification.

Node-Level Classification Given a graph with a few labeled nodes, GNNs can learn a robust model that effectively identifies the class labels for the unlabeled nodes [26]. In a node-level classification task, there are two types of training settings—inductive and transductive. In an inductive setting, the unlabeled nodes are not seen during training, while in a transductive setting, the test nodes (but not

their labels) are also observed during the training process. The transductive training setting is popular, and in this work, we used a backdoor attack in the transductive node-level classification task.

2.4 *Backdoor Attacks*

Backdoor attacks aim to make a model misclassify some of its inputs to a preset-specific label while other classification results behave normally. This misclassification is activated when a specific property is included in the model input. This property is called the trigger and can be anything the targeted model understands. For instance, a random pixel pattern [6, 19] or an actual item [52] in computer vision, a specific phrase in text classification [32], a tone in speech recognition [28], or a subgraph with specific properties in graph data [55]. The framework for the backdoor attack is shown in Fig. 1.

The first backdoor attacks targeted computer vision [6, 19] under a simple threat model, where an adversary could inject a small portion of poisoned data into the training dataset. In particular, the adversary injects into the training dataset data

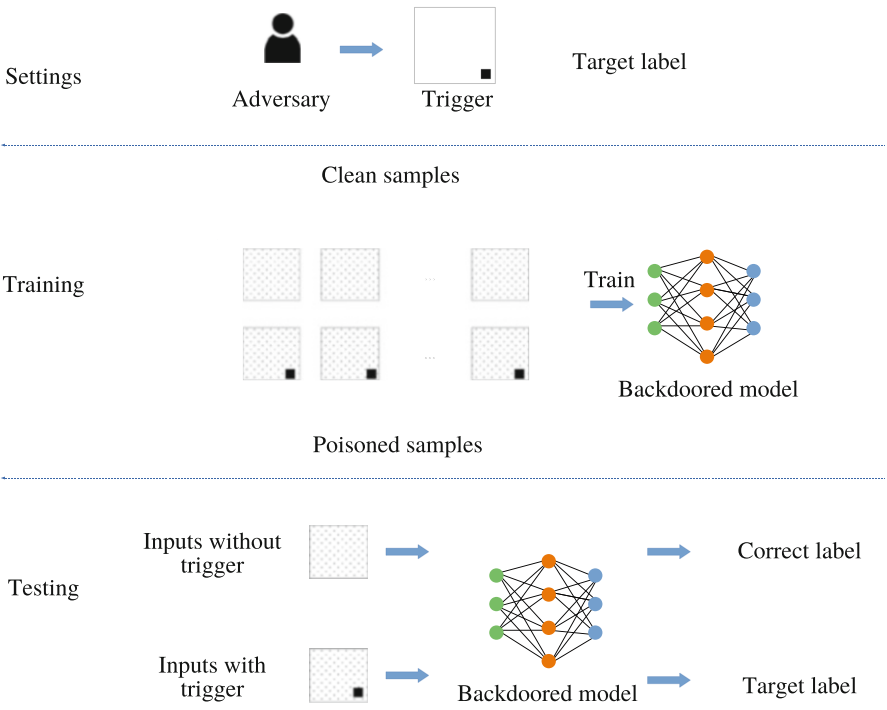

Fig. 1 Framework for the backdoor attack

stamped with a trigger that belongs to the target class. As a result, the trained model strongly associates this pattern with the target class, and whenever it is added to an input, the classification result will be the target class. Recent trends in machine learning like Machine Learning as a Service (MLaaS), outsourced training, transfer learning, and crowdsourced datasets have made this setup possible.

In MLaaS, a cloud provider provides a pay-per-request API[1] that can be used for predictions. However, the user can only use such an API as a black box without being able to verify how the model makes its predictions. Similarly, during outsourced training, the user's model is trained on the cloud and returned to the user after the training ends. Due to the lack of formal verification tools for the trained models, the user can never verify that the returned model does not contain any backdoors. Furthermore, in [19], the authors showed that a backdoor could remain effective even after a poisoned model was repurposed through transfer learning. Large crowdsourced datasets like ImageNet [11] and Mozilla's common voice [1] are so vast that cannot be exhaustively verified [39]. Thus, an adversary could inject a few poisoned samples resulting in the backdoored models.

This threat can pose real challenges as an adversary could bypass a face identification biometric access control system [6] or force an autonomous vehicle to ignore a stop sign and continue its course [19]. For this reason, backdoor attacks became very popular among researchers resulting in many novel attacks and countermeasures [15]. Novel attacks are not only limited to data poisoning but can also be based on code poisoning [2] or the direct modification of the model's parameters [22]. At the same time, due to the inability to completely understand how a deep learning model works and the lack of formal verification methods about a model's functionality, most countermeasures are empirically based on specific assumptions [4, 16, 50]. Unfortunately, in most cases, an adaptive attacker with a slightly different approach could bypass such defenses [7, 30, 43].

There are several variations of the backdoor attack resulting from different poisoning strategies. The first distinction is the class-agnostic and the class-specific backdoors [15]. The class-agnostic backdoor can be activated by a trigger injected into any input. On the other hand, the class-specific backdoor is activated only if the poisoned input belongs to a specific class. The main difference between these two strategies is that in the second case, the model needs to identify both features of the trigger and the source class making possible countermeasures more challenging [16]. Considering class-agnostic backdoor attacks, we can differentiate between the "simple" backdoor attack [19] and the clean-label backdoor attack [47].

Simple Backdoor Attack In the rest of this paper, by the simple backdoor attack, we are referring to the data poisoning backdoor attack that was introduced in BadNets [19]. In this case, the adversary adds a small subset of poisoned samples to the training dataset. These samples have been stamped with the adversary-chosen

[1] https://aws.amazon.com/transcribe/.

trigger, and their label has been changed to the target class. The target class is the output of the poisoned model when the backdoor is activated.

Clean-Label Backdoor Attack The clean-label backdoor attack was introduced in [47]. This attack is similar to a simple attack, but the adversary cannot affect the label of the injected data. The reasoning behind this attack is that the poisoned training samples can be easily identified as outliers by simple filtering mechanisms or even human inspection because the original class of these samples is different from the target class. Thus, an adaptive adversary may have to poison samples only from the target class, hoping that the model identifies the trigger pattern as a class feature. This attack is still a data poisoning backdoor attack but uses a weaker adversary making the attack more challenging.

Based on the trigger, backdoors can be either static [19] or dynamic [27]. The static backdoors are activated with a trigger that has very specific characteristics. In computer vision, such a trigger could mean a specific pixel pattern or a specific position. On the other hand, the dynamic backdoors can be activated by various triggers with different characteristics.

For graph neural networks, the first backdoor attack was proposed in [65]. In this backdoor attack, a GNN classifier predicts an attacker-chosen target label for a testing graph once a predefined subgraph is injected into the testing graph. All perturbed graphs are injected with the same trigger graph. Another backdoor attack against GNNs for the graph classification task was presented in [55], but it differs from [65] in which a universal trigger graph is assumed for all the embedded graphs. This kind of backdoor attack dynamically adapts triggers to individual graphs. The adaptive trigger is optimized in both topological structure and node features. The training processes of the trigger generation function and the backdoored GNN model are assumed as a bi-level optimization objective [14]. The authors also adapted a backtracking-based algorithm to replace a subgraph in the original graph with the adaptive trigger graph. Xu et al. [57] explored backdoor attacks on GNNs with several explainability tools. In this work, the backdoor attack is implemented with the same strategy [65] for the graph classification task. The authors also proposed a new backdoor attack strategy for the node classification task. All the above-mentioned attacks in GNNs are gray box backdoor attacks since the adversary only modifies the training dataset instead of interfering with the training of models.

2.4.1 Metrics

The successful backdoor attack should always be activated when the trigger is embedded into the model's input because an adversary wants to remain stealthy and interact with the poisoned model as little as possible. Additionally, the backdoor should not affect the original task when the trigger is not included in the input. When the poisoned model does not perform well on the original task, the backdoored model will (1) raise suspicions that something is wrong and (2) not be used, thus

preventing the adversary's plans. As a result, to measure the success of a backdoor attack, we require two metrics: the attack success rate and the clean accuracy drop.

2.4.1.1 Attack Success Rate (ASR)

The ASR shows the reliability of the attack, and it represents the number of successfully triggered backdoors from a number of poisoned inputs:

$$ASR = \frac{\sum_{i=1}^{N} F(M^*(x_i) = y_t)}{N}, \tag{3}$$

where M^* is the poisoned model, x_i is a poisoned input, y_t is the target class, and $F(x)$ is a function that returns 1 if x is true and 0 otherwise.

2.4.1.2 Clean Accuracy Drop (CAD)

This quantity shows the backdoor's effect on the original task. It is calculated by comparing the performance of a poisoned and a clean model for clean inputs. The accuracy drop should generally be small to keep the attack stealthy.

3 Methodology

3.1 Threat Model

In this work, we implement data poisoning backdoor attacks. The adversary injects a small subset of poisoned data without knowing any information about the model architecture or the training algorithm. Thus, the attack follows a gray box threat model. This threat model is realistic as current large datasets are crowdsourced [1, 11] and malicious data may go through the validation process [39]. So, an adversary could inject trigger-stamped data in such datasets that will remain unnoticed and used during training resulting in a successful backdoor attack.

In our experiments, we investigate two different attacks, the simple data poisoning attack [19] and the clean-label attack that does not alter the labels of the poisoned data [47]. For both attacks, the adversary aims to cause targeted misclassifications with a very high probability without affecting the model's performance on the original task.

3.2 Image Classification

Attacks We use two different attacks: the simple backdoor attack and the clean-label attack.

Datasets For our image classification backdoor attacks, we use two popular image datasets: (i) CIFAR10 that consists of 60,000 32×32 color images in ten classes, with 6000 images per class. There are 50,000 training images and 10,000 test images. (ii) Fashion-MNIST (FMNIST) [56]—a dataset of Zalando's article images consisting of a training set of 60,000 images and a test set of 10,000 images. Each image is a 28×28 gray-scale image associated with a label from ten classes.

For the CIFAR10 dataset, we split the test set in an i.i.d manner into two 5000 sample datasets, each used for validation and test, respectively. For the FMNIST dataset, we split the training set into two different sized datasets in an i.i.d manner: the first having 50,000 samples used for training and the second having 10,000 samples for validation. With this, we have the same size of training samples for both datasets, so comparing results between these two is easier.

Features The input features for both neural networks are the tensor of images. For the CIFAR10 dataset, each RGB image is considered as a [3, 32, 32] shape tensor. For FMNIST, however, the images are gray-scale, so the input has only one channel ([1, 28, 28] shape tensor). We also did the standard normalization for input values before all train, validation, and test phases.

Models We use two models: STRIPNet [16] and ResNet [21] with nine residual blocks (ResNet-9).

Trigger As described in [27], various triggers have been used in image classification, and all of them resulted in successful backdoor attacks. This means that the trigger shape and pattern are not crucial for the success of a backdoor attack. Thus, for our experiments, we chose a square trigger. Its pixel intensities are random values retrieved from a continuous uniform distribution (pseudorandom generator). The seed in this generator was fixed for consistency in our experiments.

3.3 Natural Language Processing

Attacks Similar to image classification, we use simple and clean-label backdoor attacks.

Datasets In our experiments, we used the IMDB [33] and the AG News topic classification [64] datasets. The IMDB dataset consists of 50,000 (50%/50% training/test split) movie reviews of high polarity (either positive or negative). We used 20% of the training data for validation. The AG News topic classification dataset consists of news articles belonging to four categories (world, sports, business, and science/technology). The training set consists of 120,000 samples and the testing set

of 7600 samples. Again, we used 20% of the training data for validation resulting in 96,000 and 24,000 samples for training and validation sets, respectively.

Features The first step of our pipeline is a `TextVectorization` layer that transforms each input to a convenient form for processing as described in [27]. As the datasets are different, we used a different sequence length for each dataset. We forced the length of each sentence to be 250 words for the IMDB dataset and 197 for the AG News dataset. Additionally, we used a vocabulary of 10,000 words that proved enough for such small datasets.

Models We used two publicly available CNN architectures. Both the first CNN[2] and the second CNN[3] use an embedding layer as their input. However, the first CNN uses a small trainable embedding of size 16, and the second uses a pretrained GloVe embedding [38] of size 100. We want to investigate if the attack becomes more difficult when the model uses a pretrained embedding because this is more frequent in practice. Such embeddings have been trained in large corpora of text and interpret possible connections between different words more accurately. To illustrate, Google's pretrained word2vec is trained with 100 billion words from Google News, and it contains 300-dimensional vectors for 3 million words and phrases [34]. The GloVe is trained from a corpus of 6 billion words and has a vocabulary of 400,000 words [38].

Trigger As the trigger, we used a sentence of 1 up to 4 words from the list ["trope," "everyday," "mythology," "sparkles," "ruthless"] as defined in [32]. We applied the trigger in three positions (beginning, middle, and end) to investigate whether our models are more sensitive in specific positions.

3.4 Speech Recognition

Attacks Again, we use simple and clean-label backdoor attacks.

Datasets For this application, we used two versions of the Speech Commands dataset as described in [28]. The first version uses ten classes of the dataset and the second 30 classes. From our experiments, we excluded the samples that lasted less than one second to avoid variable-sized inputs in our pipeline resulting in 21,312 .wav files in the first case and 58,252 files in the second case. In both cases, we use 64%/16%/20% for training, validation, and testing.

Features As our input features, we used the MFCCs of each training input. The exact hyperparameters for this calculation are described in [28].

Models We used one CNN [32] and one LSTM [10] for our experiments.

[2] https://www.tensorflow.org/tutorials/keras/text_classification.

[3] https://keras.io/examples/nlp/pretrained_word_embeddings/.

Trigger Our dataset's sound files are sampled at 16 kHz, and according to the Nyquist-Shannon sampling theorem, the largest tone frequency that can be included in such digital signals is 8 kHz. Thus, following [27], our trigger is a 7 kHz tone which is a high pitch audible sound. Following the rest of the triggers tried, this trigger differs from the normal dataset samples. It lasts from 20 to 80 ms because we want to model an adversary that is as stealthy as possible. The trigger is injected in three different positions of each sound sample (beginning, middle, and end).

3.5 Graph Data

Attacks As described in Sect. 2.4, for graph neural networks, we utilize two backdoor attacks, i.e., AT^{I} [65] and AT^{II} [55]. The framework for AT^{I} is illustrated in Fig. 2. In the training phase (Fig. 2a), the attacker injects a trigger (subgraph g_t) to a subset of training graphs and changes their labels to an attacker-chosen target label. A GNN classifier is then trained using the backdoored training dataset, and such GNN is called backdoor GNN Φ_b. In the test phase (Fig. 2b), once the test graph is injected with the same trigger graph, the backdoored GNN is likely to misclassify the testing sample to the target label. For the node classification task, we used the backdoor attack from [57].

Since [65] and [57] designed the same strategy to implement the backdoor attack for the graph classification task, we illustrate the results of [65] and [55] for the graph classification task. The results based on [57] are presented for the node classification task.

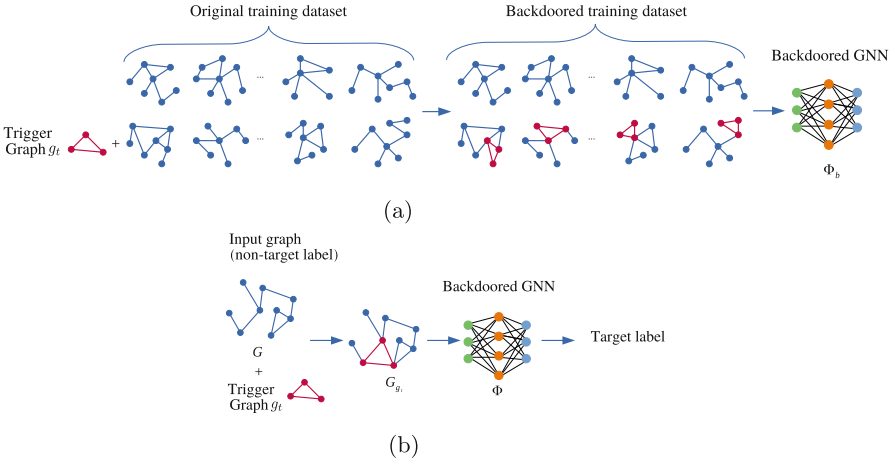

Fig. 2 Subgraph-based backdoor attack for the graph classification task. (**a**) Training. (**b**) Testing

Table 1 Graph datasets statistics

Datasets	# Graphs	Avg. # nodes	Avg. # edges	Classes	Class distribution
AIDS	2000	15.69	16.20	2	400[0], 1600[1]
TRIANGLES	45,000	20.85	32.74	10	4500[0–9]
Cora	1	2708	5429	7	351[0], 217[1], 418[2], 818[3], 426[4], 298[5], 180[6]
CiteSeer	1	3327	4608	6	264[0], 590[1], 668[2], 701[3], 596[4], 508[5]

Datasets Table 1 shows the statistics for all considered datasets for graph neural networks. For the graph classification task, we use two publicly available graph datasets. (i) AIDS [35]—a dataset consisting of graphs representing molecular compounds that are active against HIV or not; (ii) TRIANGLES [35]—a synthetic dataset designed to solve the task of counting the number of triangles in a graph. For each graph classification dataset, we sample 2/3 of the graphs as the original training dataset and treat the remaining graphs as the original testing dataset. Among the original training dataset, we randomly sample α fraction of graphs to inject the trigger and relabel them with the target label, called the backdoored training dataset. Several parameters can affect the attack effectiveness: trigger size s, trigger density ρ, and poisoning intensity α. Unlike other domains, e.g., image classification, the trigger position in graph data is irrelevant and cannot be defined because a graph is non-Euclidean data where we cannot put nodes in some order. For AT^{I}, we use Erdős-Rényi (ER) model [17] to generate the trigger graph, as it is more effective than the other methods [65].

For the node classification task, we use two real-world datasets: (i) Cora [41]—a citation network in which each publication is described by a binary-valued word vector indicating the absence/presence of the corresponding word in the collection of 1433 unique words. (ii) CiteSeer [41]—another citation network with more nodes but less edges. For each node classification dataset, we split 20% of the total nodes as the original training dataset (labeled) and the rest as the original testing dataset (unlabeled). To generate the backdoored training dataset, we sample α of the original training dataset to inject the feature trigger and relabel these nodes with the target label. The feature trigger width is set to be n. Moreover, based on the conclusion in [57], different feature trigger injecting positions have a negligible impact on the attack performance, so the trigger injecting position is randomly selected. Here, we explore the impact of poisoning intensity α and feature trigger width n on the attack performance. In the node classification task, each node feature has a value of 0 or 1, and here we set the value of the modified node features to 1 (note, the values could also be set to 0).

Features Each graph contains topological and node feature information. For each graph dataset in this work, there is an adjacency matrix and feature information matrix. For AIDS, Cora, and CiteSeer, there is a specific node feature vector for

each node in the graph, but for TRIANGLES, the one-hot degree of a node is used as the node feature.

Models We use two state-of-the-art GNN models for the graph classification task: GCN [26] and GraphSAGE [20]. We use GCN [26] and GAT [48] for the node classification task.

Trigger For the graph classification task, our trigger is a global (adaptive) subgraph in $AT^{\mathrm{I}}(AT^{\mathrm{II}})$. For the node classification task, our trigger is a subset of node features with a fixed value, e.g., 0.

4 Experimental Results

4.1 Image Classification

Chosen Settings and Selected Parameters We ran our experiments with a different number of poisoned samples (25, 300, 575, 850), trigger sizes (4×4, 8×8, 12×12), and trigger positions (Upper-Mid, Mid-Left, Mid-Right, Lower-Mid) on the image. Figure 3 demonstrates four different positions of a 4×4 trigger for several FMNIST sample images. We repeated each experiment two times, which makes the total number of 768 experiments regarding the chosen settings. We set class number 5 as the target for all experiments and in both datasets.

Every backdoor attack should remain stealthy without affecting the original task. Therefore, the poisoned model should perform as expected when the input does not contain the trigger. In Table 2, we compare the performance of clean and backdoored models for clean inputs. The attack accuracy mentioned in this table is the arithmetic mean ($\pm$ the standard deviation) of the accuracy on clean inputs from all the poisoned models trained in our experiments. For the original accuracy, we trained multiple clean models and averaged their performance. The model remains unaffected from both backdoor attacks even if we use 850 poisoned samples. Such poisoning rates are small and cannot affect the model's performance in general. From Table 2, we can also verify that our models perform similarly well in both

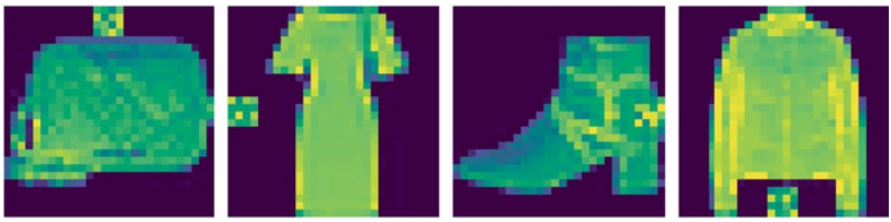

Fig. 3 Applied 4×4 trigger in different positions: Upper-Mid, Mid-Left, Mid-Right, Lower-Mid

Table 2 Clean accuracy drop in image classification

Dataset	Model	Original Acc		Number of poisoned Samples			
			Attack type	25	200	375	850
CIFAR10	STRIPNet	85.09 (± 0.599)	Clean-label	85.22 (± 0.508)	85.38 (± 0.523)	85.37 (± 0.378)	85.23 (± 0.471)
			Simple	85.31 (± 0.467)	85.28 (± 0.438)	85.35 (± 0.360)	85.12 (± 0.477)
	Resnet-9	89.99 (± 0.499)	Clean-label	89.98 (± 0.356)	89.86 (± 0.300)	89.79 (± 0.461)	89.69 (± 0.377)
			Simple	89.94 (± 0.353)	89.77 (± 0.274)	89.75 (± 0.266)	89.74 (± 0.378)
FMNIST	STRIPNet	93.62 (± 0.205)	Clean-label	93.66 (± 0.080)	93.68 (± 0.127)	93.64 (± 0.131)	93.69 (± 0.139)
			Simple	93.59 (± 0.135)	93.61 (± 0.148)	93.67 (± 0.152)	93.65 (± 0.139)
	Resnet-9	93.63 (± 0.154)	Clean-label	93.59 (± 0.152)	93.53 (± 0.157)	93.62 (± 0.171)	93.59 (± 0.176)
			Simple	93.57 (± 0.129)	93.55 (± 0.181)	93.59 (± 0.164)	93.58 (± 0.173)

datasets, which is helpful when comparing the performance of the attack for each case.

Results for FMNIST As it can be inferred from Fig. 4, the clean-label attack is not that effective against the FMNIST dataset. By increasing the number of poisoned samples, there are small or no improvements in attack success rate (there are small improvements when increasing the number of samples from 25 to 300, but as we increase from 300 to 575 and from 575 to 850, the improvements become even smaller). We assume this is mainly due to the dataset nature and the capability of the CNNs to learn the exclusive features of each class easily and robustly so that injecting a trigger (even of size 12×12) could not disturb the network from learning those.

Since both ResNet-9 and STRIPNet have convolutional layers, we expect negligible effects of trigger position on attack success rate. The results confirm this as there are only minor effects stemming from the trigger positions (for instance, in both networks, the trigger on the lower-mid results in the least ASR, while on the mid-right, it has a little more chance of being learned by the network. Again, we suppose this is because of the attributes of the FMNIST images and the models' focus on specific regions of an image to learn). Additionally, in almost all cases (except a few ones like upper-mid in ResNet-9), increasing the trigger size leads to higher ASR.

For the simple attack, we obtained 100% ASR for 300 attack samples or more. With 25 poisoned samples, some trigger positions have positive effects on ASR regardless of trigger size (for instance, the mid-left trigger achieves high ASR even with 4×4 size triggers).

Results for CIFAR10 The clean-label attack is significantly more effective for CIFAR10 than FMNIST (Figs. 4 and 5). We believe this is primarily because the CIFAR10 images are RGB, and the crafted trigger has more layers (3 channels). As a result, the model learns the embedded trigger with less poisoned samples. As expected, the trigger position does not play an important role in ASR, and in almost all cases, ASR improves using a larger trigger size.

Another observation is that the smaller the size of the trigger, the more noticeable the ASR improvement when increasing the number of poisoned samples from 25 to 850. For instance, for a 12×12 trigger, there is no noticeable improvement in ASR when the number of poisoned samples increases from 575 to 850. On the other hand, using a 4×4 trigger, ASR's growth is easily observable between all four different poisoning rates.

Analyzing the simple attack, similarly to FMNIST, we achieved 100% ASR for 300 poisoned samples or more. Additionally, ResNet-9 is more vulnerable to backdoor attacks, particularly when using fewer poisoned samples and smaller triggers. We believe this is mostly because ResNet-9 is a larger network than STRIPNet and can extract more data from the given dataset.

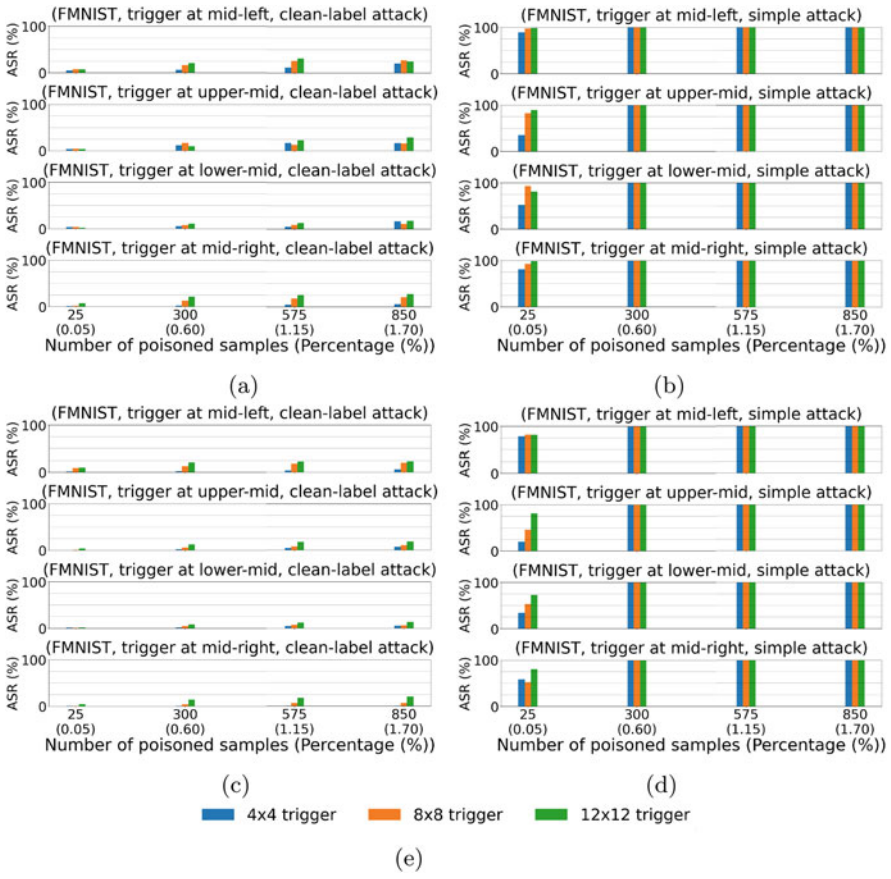

Fig. 4 Attack accuracy for the FMNIST dataset. From these figures, we conclude that the clean-label attack is not effective but is slightly improved when increasing the poisoning rate. On the other hand, the simple attack can be very effective even with a small poisoning rate (0.6%). Additionally, larger triggers lead to higher ASR, but different trigger positions do not result in ASR fluctuations as the convolutional layers identify the trigger. (**a**) ResNet-9 + clean-label attack. (**b**) ResNet-9 + simple attack. (**c**) STRIPNet + clean-label attack. (**d**) STRIPNet + simple attack. (**e**) Legend

4.2 Natural Language Processing

In Tables 3 and 4, we compare the performance of clean and backdoored models in text classification when clean inputs are used. These tables are generated by averaging the performance of clean and poisoned models as described in Sect. 4.1. In all cases, the model's performance remains almost unaffected after the backdoor insertion. There are a few minor accuracy drops that are at most 0.6% making the

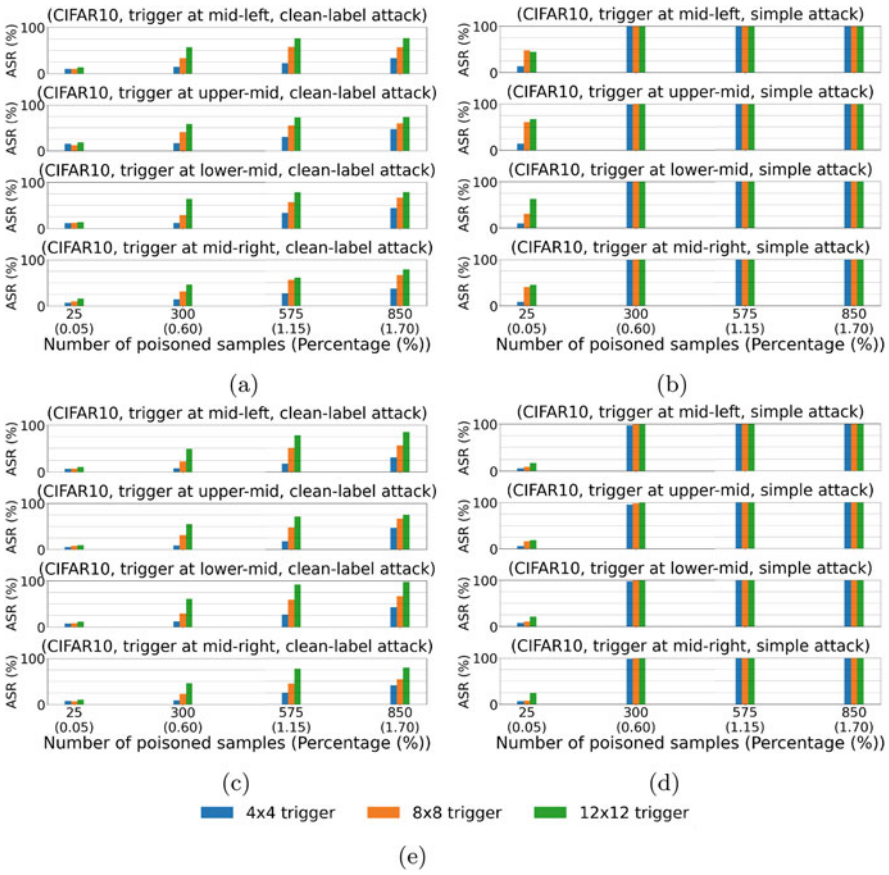

Fig. 5 Attack accuracy for the CIFAR10 dataset. The clean-label attack is significantly more effective than for FMNIST because the 3-channel trigger contains more information. We also see that the trigger position is not very important, and ASR increases as the trigger size increases. The ASR with the simple attack is 100% for a 0.6% poisoning rate or more. However, STRIPNet is not as vulnerable as ResNet due to its smaller capacity. (**a**) ResNet-9 + clean-label attack. (**b**) ResNet-9 + simple attack. (**c**) STRIPNet + clean-label attack. (**d**) STRIPNet + simple attack. (**e**) Legend

attack stealthy. This behavior is expected as we poison only a small subset of the training data that cannot substantially affect the model's learning.

In Figs. 6 and 7, we show the results of our experiments for the AG News topic classification dataset and IMDB dataset, respectively. From these figures, we can draw several conclusions. In most cases, the ASR is correlated with the trigger size and increases as the trigger size increases. This is true even when the attack is not effective (see Fig. 6a).

Table 3 Clean accuracy drop in text classification (AG News)

Model	Original Acc	Attack type	Number of poisoned samples			
			250	500	750	1000
CNN	90.74 (± 0.314)	Clean-label	90.48 (± 0.557)	90.52 (± 0.531)	90.52 (± 0.614)	90.51 (± 0.541)
		Simple	90.49 (± 0.534)	90.38 (± 0.593)	90.53 (± 0.484)	90.33 (± 0.760)
CNN + GloVe	89.78 (± 0.169)	Clean-label	89.72 (± 0.218)	89.71 (± 0.201)	89.71 (± 0.186)	89.70 (± 0.220)
		Simple	89.72 (± 0.209)	89.68 (± 0.206)	89.69 (± 0.227)	89.68 (± 0.202)

Table 4 Clean accuracy drop in text classification (IMDB)

Model	Original Acc	Attack type	Number of poisoned samples			
			50	100	150	200
CNN	87.05 (± 0.062)	Clean-label	86.91 (± 0.084)	86.93 (± 0.071)	86.91 (± 0.078)	86.90 (± 0.072)
		Simple	86.90 (± 0.083)	86.89 (± 0.089)	86.88 (± 0.090)	86.88 (± 0.096)
CNN + GloVe	84.20 (± 0.308)	Clean-label	83.73 (± 0.681)	83.71 (± 0.702)	83.62 (± 0.775)	83.77 (± 0.680)
		Simple	83.79 (± 0.535)	83.82 (± 0.519)	83.61 (± 0.570)	83.59 (± 0.724)

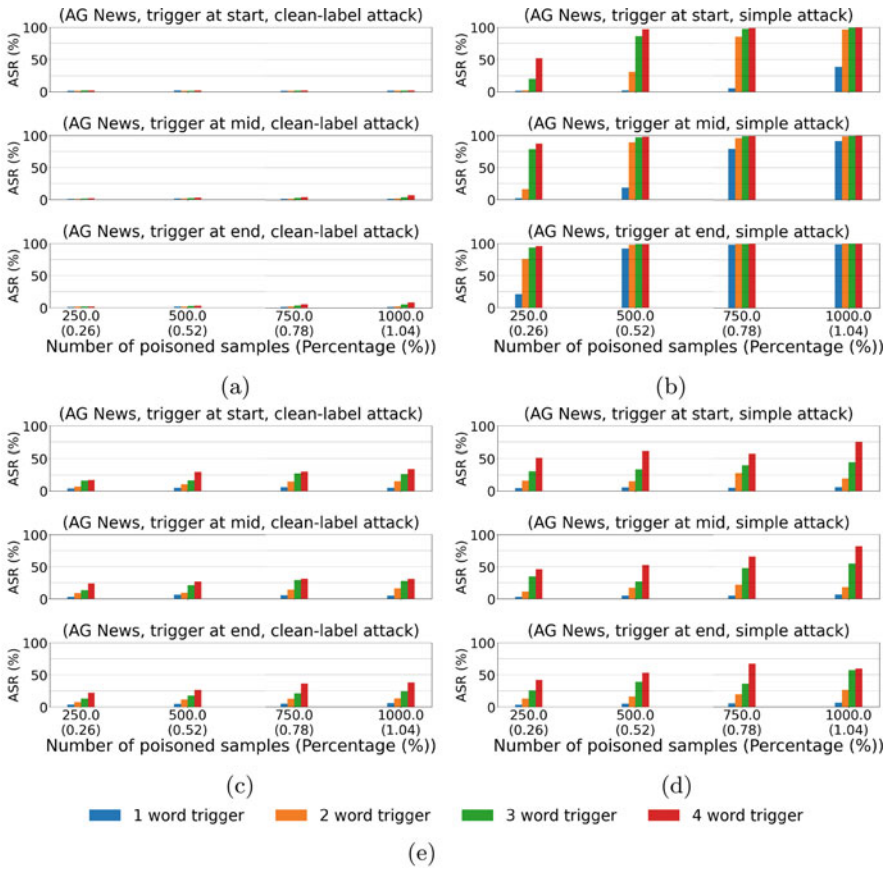

Fig. 6 Attack accuracy for the AG News dataset. The ASR is positively correlated with the trigger size (even when the ASR is very low), and the poisoning rate significantly influences the attack's effectiveness. Additionally, the clean-label attack needs more poisoned data to work. When GloVe is used, inserting the trigger in the end results in higher ASR (especially for low poisoning rates), but in the simple CNN, the trigger positions do not affect ASR. (**a**) CNN with GloVe + clean-label attack. (**b**) CNN with GloVe + simple attack. (**c**) CNN + clean-label attack. (**d**) CNN + simple attack. (**e**) Legend

Especially for the first CNN, this relation seems to be linear (see Figs. 6c, 6d, 7c and 7d). This simple model uses global average pooling as its penultimate layer, averaging the feature map before the output. As a result, the trigger will be more influential when it consists of more words. In almost all experiments, the poisoning rate is a highly influential hyperparameter of the backdoor attack, and any increase in it leads to an increase in the attack success rate.

Our models learn differently, which can be seen from the varying attack success rate when the trigger is injected in different positions. For example, the attack success rate is higher if the trigger is inserted at the end of the sentence when we use

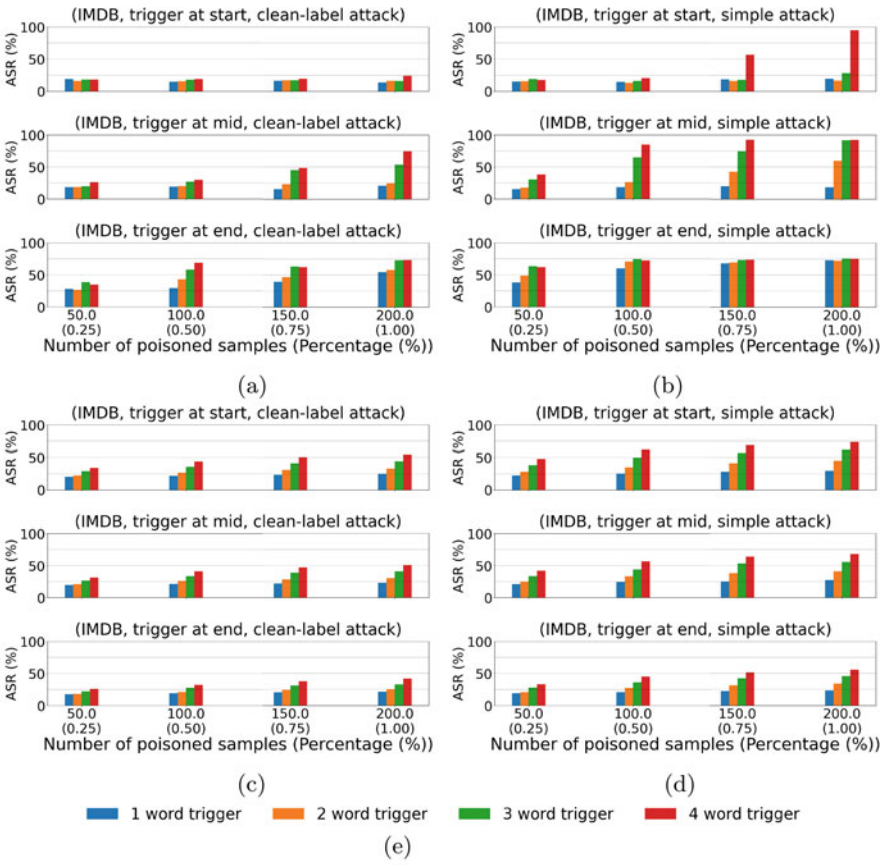

Fig. 7 Attack accuracy for the IMDB dataset. The ASR is positively correlated with the trigger size, and the poisoning rate significantly influences the attack's effectiveness. Additionally, the clean-label attack is more effective with this dataset. For the CNN that uses GloVe, placing the trigger at the end of the sentence yields the best results, but for the simple CNN, this is the least effective position. (**a**) CNN with GloVe + clean-label attack. (**b**) CNN with GloVe + simple attack. (**c**) CNN + clean-label attack. (**d**) CNN + simple attack. (**e**) Legend

the first model and the simple backdoor attack (see Figs. 6b and 7b). This difference is very clear for low poisoning rates (0.25%), where even a small trigger of 2 words could be substantially more effective when placed at the end of the sentence. On the other hand, for the other model, placing the trigger at the end does not result in higher ASR (see Figs. 6d and 7d). These differences indicate that we could use the backdoor attack as a tool for AI explainability and further understand what and how a model learns by using triggers with different characteristics.

In [47], the authors claimed that the clean-label backdoor attack needs a very large poisoning rate to be effective. We also see this behavior in the AG News dataset, especially for the architecture that uses the pretrained GloVe embedding

(Fig. 6a). In the other architecture, the clean-label attack is more effective as the feature space created by the trainable embedding encodes some information about the trigger and the target class (see Fig. 6c). However, when the IMDB dataset is used, both models perform similarly without poisoning a very large part of the training data for the clean-label (see Figs. 7a and 7c). This can be explained by the differences between the datasets. Each sentence in AG News is shorter than the movie reviews in the IMDB dataset. Additionally, most of the words in AG News are strongly connected with the topic that each sentence belongs to (world, sports, business, and science/technology), which is not true for the IMDB dataset. In the IMDB dataset, the sentences are longer, and usually, only a few words are related to their sentiment. As a result, in AG News, our attack needs more poisoned samples to overcome the effect of the original features of the source class.

4.3 Speech Recognition

In Tables 5 and 6, we compare the performance of clean and backdoored models for sound classification when clean inputs are used. As was also shown in [28], the differences between the clean and the backdoored models are negligible. In particular, the backdoored models perform a little better when the 10 classes dataset is used, meaning that the poisoned samples could serve as a generalization factor. However, when the full dataset is used, the backdoor insertion results in a small performance drop for the CNN. In this case, we use more classes, and the model has to learn a more difficult task that is affected even by a few poisoned samples. The performance of the LSTM is slightly increased, meaning that the LSTM builds different models and utilizes its capacity better when we use the full dataset [28]. All these differences are small, and our claims need additional experimental data to be confirmed.

In Figs. 8 and 9, we show the results of our experiments for the first (10 classes) and the second (30 classes) version of the Speech Commands dataset. In almost all cases, the attack success rate increases as the trigger duration increases. This is true even when the attack is not successful (see the clean-label attack in Figs. 8c and 9c). This makes sense as more input features are affected when a longer trigger is used, and the network can learn this relation easier. Additionally, the poisoning rate is very influential, and its increase leads to more effective backdoors.

The end of the input is the most effective trigger position for the LSTM network in both versions of the dataset. Even though this network uses two bidirectional LSTM layers and an attention layer, it seems to learn the features that are placed towards the end of its inputs more easily. The LSTM network was designed to tackle the problem of long-term dependencies on its inputs. A possible reason for this behavior is the nature of this particular dataset, which consists of 1-second clips of spoken words. If these words are not perfectly centered and distributed to the upper half of each sample, our network will give more attention to the end of each training sample. This is not true for the CNN used as, in that case, all the positions seem to

Table 5 Clean accuracy drop in sound classification (10 classes)

Model	Original Acc	Attack type	Number of poisoned samples			
			50	100	150	200
CNN	94.82 (± 0.360)	Clean-label	95.07 (± 0.437)	95.00 (± 0.477)	95.07 (± 0.427)	94.96 (± 0.508)
		Simple	95.09 (± 0.417)	95.04 (± 0.438)	94.99 (± 0.474)	94.95 (± 0.480)
LSTM	89.47 (± 1.412)	Clean-label	89.77 (± 1.426)	89.69 (± 1.350)	90.05 (± 1.362)	89.89 (± 1.373)
		Simple	89.71 (± 1.605)	89.96 (± 1.335)	89.66 (± 1.482)	89.80 (± 1.586)

Table 6 Clean accuracy drop in sound classification (30 classes)

Model	Original Acc	Attack type	Number of poisoned samples			
			137	274	411	547
CNN	94.70 (± 0.395)	Clean-label	94.61 (± 0.410)	94.55 (± 0.439)	94.49 (± 0.575)	94.59 (± 0.461)
		Simple	94.54 (± 0.471)	94.63 (± 0.511)	94.56 (± 0.496)	94.53 (± 0.477)
LSTM	90.50 (± 0.967)	Clean-label	90.88 (± 1.267)	90.59 (± 1.238)	90.82 (± 1.232)	90.79 (± 1.334)
		Simple	90.86 (± 1.236)	90.81 (± 1.167)	90.67 (± 1.317)	90.63 (± 1.278)

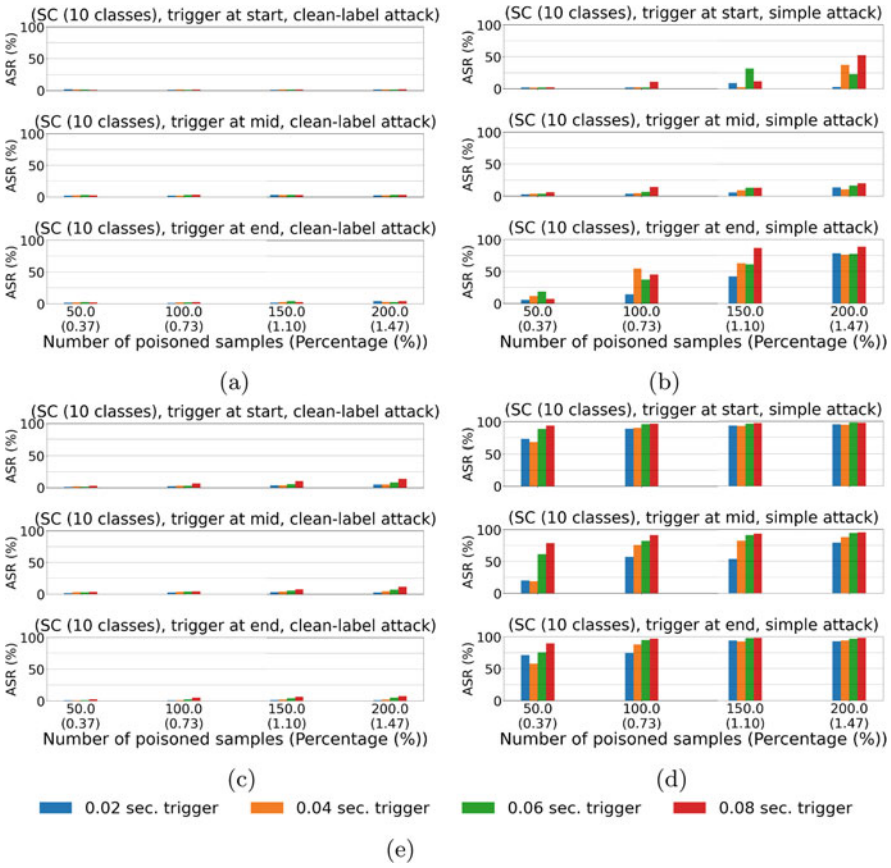

Fig. 8 Attack accuracy for the Speech Commands dataset (10 classes). In most cases, ASR is increased as the trigger duration or the poisoning rate increase. The clean-label attack is ineffective for both models. For LSTM, the best position for the trigger is at the end. However, for CNN, any position works. For CNN, the simple attack works almost perfectly for 100 poisoned samples or more. (**a**) LSTM + clean-label attack. (**b**) LSTM + simple attack. (**c**) CNN + clean-label attack. (**d**) CNN + simple attack. (**e**) Legend

be equally effective (see Figs. 8d and 9d). Similarly to text classification, different models learn different patterns from the same dataset making the backdoor attack effective in different cases. Thus, we could use the backdoor attack and its triggers to understand what a model learns and how it makes its decisions.

In our sound classification experiments, the clean-label attack is not successful for both neural networks and datasets. However, when the full dataset and CNN are used (Fig. 9c), the attack success rate slightly increases with large triggers. The clean-label could work without requiring more poisoned data if we choose a larger trigger. This claim, though, needs to be verified in the future with more experimental evidence. Another interesting observation is that the simple backdoor

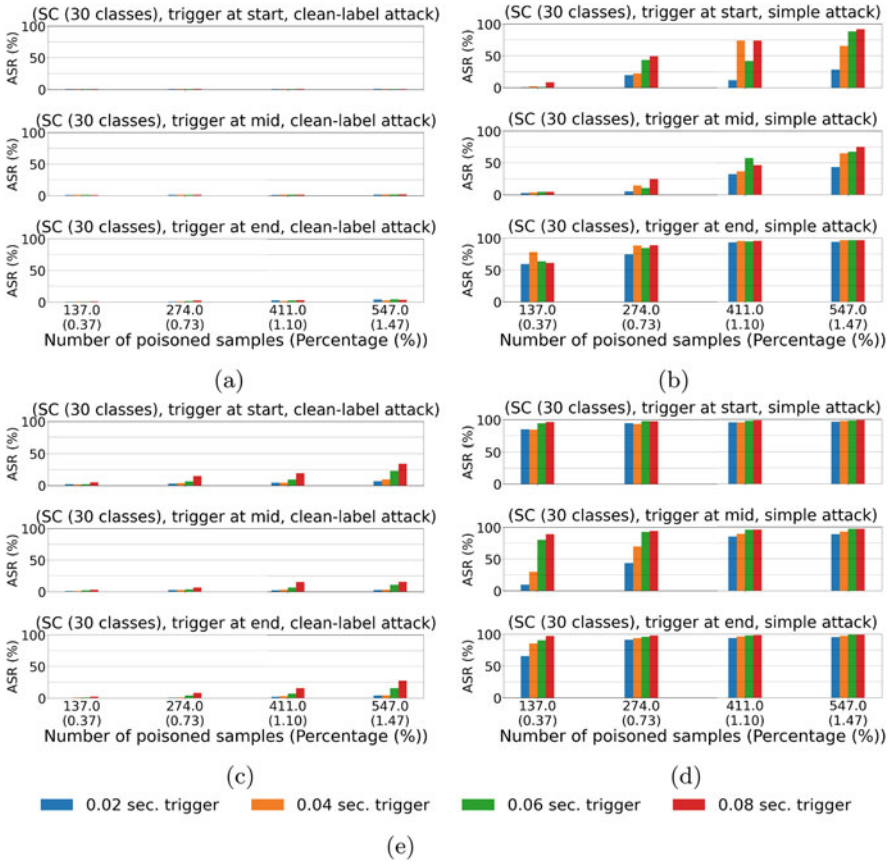

Fig. 9 Attack accuracy for the Speech Commands dataset (30 classes). In most cases, ASR is increased as the trigger duration or the poisoning rate increase. The clean-label attack is ineffective in that case, but it works slightly better for CNN. In the simple attack, the best position for the trigger is at the end for LSTM. However, there is no difference for CNN (in most cases, ASR is close to 100%). When using LSTM, ASR is higher than ASR for the ten classes. We assume that the absolute number of poisoned samples could be the reason behind that behavior. (**a**) LSTM + clean-label attack. (**b**) LSTM + simple attack. (**c**) CNN + clean-label attack. (**d**) CNN + simple attack. (**e**) Legend

attack becomes more effective when we use the full dataset and the LSTM network (compare Fig. 8b to Fig. 9b). One reason for this behavior is the absolute number of training samples that were increased when the full dataset was used. However, this should be investigated further.

4.4 Graph Data

Results for the Graph Classification Task For the graph classification task, two parameters affect the performance of the backdoor attack: poisoning intensity and trigger size (the number of nodes in the trigger graph). The attack results for the GCN model on AIDS with different poisoning intensity α and trigger size s are shown in Fig. 10. As we can see from Fig. 10a, with the increase of poisoning intensity, the attack success rate is generally increasing for each trigger size, but there is no obvious improvement between $\alpha = 0.15$ and $\alpha = 0.2$. Here, we select poisoning intensity $\alpha = 0.15$ for GCN on AIDS. Figure 10b shows the impact of trigger size under the selected poisoning intensity ($\alpha = 0.15$). The attack success rate is highest with $s = 5$, while the clean accuracy drop is the smallest when $s = 5$. To compare the two backdoor attacks, we set $\alpha = 0.15$, $s = 5$ and $\alpha = 0.2$, $s = 7$ for AIDS and TRIANGLES, respectively.

Specifically, we present the attack results of two backdoor attacks on the graph classification task in Tables 7 and 8. As we can see from Table 7, AT^{II} can achieve more than 99% attack success rate and less than 1% clean accuracy drop on AIDS, while the performance of AT^{I} degrades slightly with an attack success rate of more than 95% and clean accuracy drop around 1.5%. As illustrated in Table 8, the attack success rate of AT^{II} is significantly higher than AT^{I} for TRIANGLES, i.e., more than 10%. However, the clean accuracy drop of AT^{II} is larger than AT^{I}, which is more than 4% for both models, while that of AT^{I} is around 3% and less than 1%

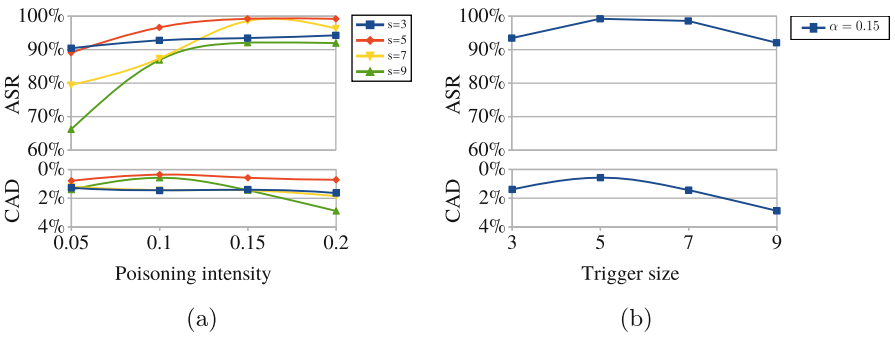

Fig. 10 Impact of poisoning intensity and trigger size on attack performance in the graph classification task. (**a**) GCN_AIDS. (**b**) GCN_AIDS ($\alpha = 0.15$)

Table 7 Backdoor attack results for the graph classification task and the AIDS dataset

	AT^{I}		AT^{II}	
Setting	ASR (%)	CAD (%)	ASR (%)	CAD (%)
GCN	95.86	1.25	99.92	0.46
GraphSAGE	97.59	1.46	99.80	0.91

Table 8 Backdoor attack results for the graph classification task and the TRIANGLES dataset

	AT^{I}		AT^{II}	
Setting	ASR (%)	CAD (%)	ASR (%)	CAD (%)
GCN	86.00	3.18	99.21	5.32
GraphSAGE	87.70	0.50	98.24	4.32

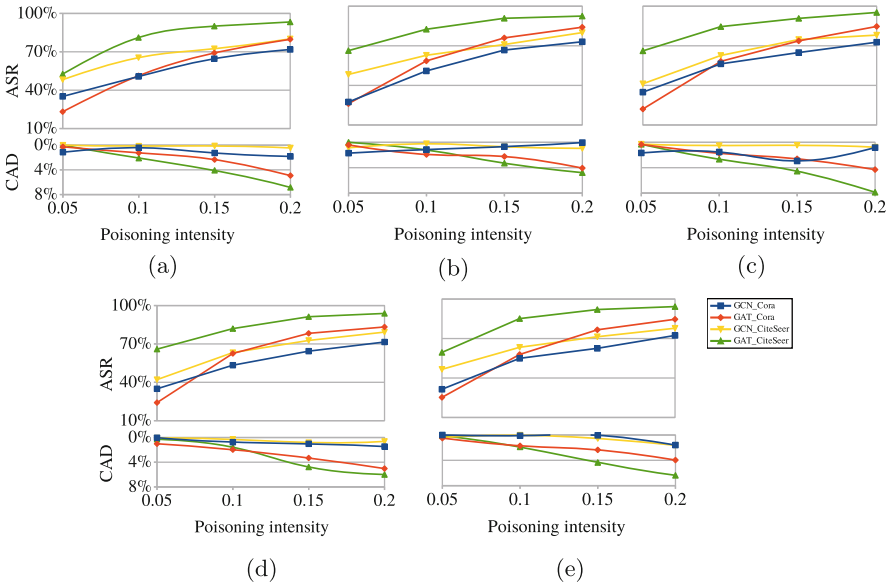

Fig. 11 Impact of poisoning intensity and feature trigger width on attack performance in the node classification task. (**a**) $n = 5$. (**b**) $n = 10$. (**c**) $n = 15$. (**d**) $n = 20$. (**e**) $n = 25$

for GCN and GraphSAGE, respectively. In addition, the computation time for AT^{II} is around 1.7 times of AT^{I}.

Results for the Node Classification Task For the node classification task, the backdoored data is influenced by two parameters: poisoning intensity α and feature trigger width n. The attack performance, including attack success rate and clean accuracy drop with different variants, is shown in Fig. 11. For each feature trigger width, the attack success rate on different models and datasets generally increases when the poisoning intensity increases from 0.05 to 0.2. At the same time, the clean accuracy drop of the GCN model keeps increasing, and there is a significant increase between $\alpha = 0.15$ and $\alpha = 0.2$. However, the clean accuracy drop of the GAT model remains almost unchanged. To achieve a high attack success rate and low clean accuracy drop, we set $\alpha = 0.2$ for GCN and $\alpha = 0.15$ for GAT. To evaluate the impact of feature trigger width on attack performance, we show the attack results with different feature trigger widths in Fig. 12. Observe that the feature trigger width

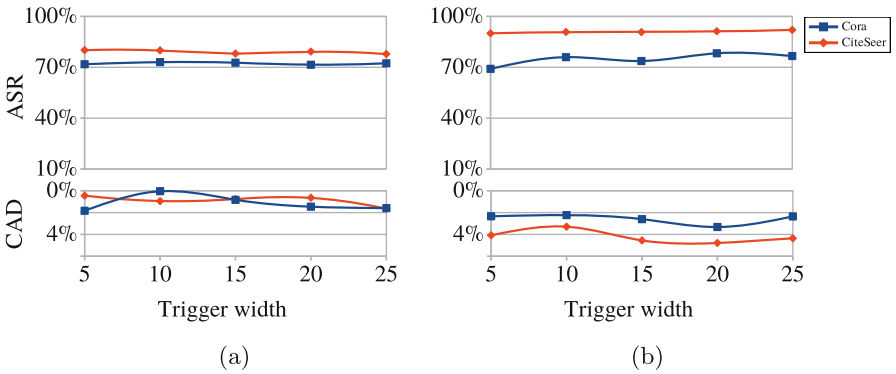

Fig. 12 Attack performance with different feature trigger widths. (**a**) GCN ($\alpha = 0.2$. (**b**) GAT ($\alpha = 0.15$

Table 9 Backdoor attack results in the node classification task ($n = 5$)

	GCN ($\alpha = 0.2$)		GAT ($\alpha = 0.15$)	
Setting	ASR (%)	CAD (%)	ASR (%)	CAD (%)
Cora	72.35	1.59	86.63	2.35
CiteSeer	77.82	1.63	92.04	1.35

has no obvious influence on the attack success rate and clean accuracy drop for both GNN models and datasets.

Specifically, Table 9 shows the attack success rate and clean accuracy drop of backdoor attack for the node classification task with selected parameters. Notice that the backdoor attack on GCN reaches over 70% attack success rate for both datasets and that on GAT obtains a higher attack success rate, i.e., over 85% and 90% for Cora and CiteSeer, respectively. Furthermore, the clean accuracy drop is lower than 2% for all models and datasets except for the GAT model on the Cora dataset, which is 2.35%.

4.5 General Observations

First, we verified that the backdoor attack is a real threat as it can be injected into every application domain tried without affecting the model's original task just by poisoning a small subset of the training data. Additionally, we saw that the poisoning rate is the most influential characteristic of the trigger in all applications. However, this value cannot be increased arbitrarily because the backdoor attack will become evident through a simple data filtering mechanism, and the poisoned model's performance on clean inputs will decrease substantially.

The trigger size is positively correlated with the backdoor's attack success rate in image, text, and sound. This is expected as a larger trigger contains more

information that can be encoded easier in the trained model. However, in graph classification, the attack success rate increased to a point ($s = 5$) and then decreased for larger triggers. The variations are small, though, as ASR remained above 90% in our experiments, and thus, we cannot draw general conclusions. We need to verify this effect with more complex datasets and models.

The most effective position of the trigger (if there is any) depends on many factors, like the network architecture or the dataset. The position is not very influential on the attack success rate in most cases, but this is not always true. Thus, we cannot draw any general conclusions. In image classification, no position was proven more effective as the convolutional layers extract information from any point in the image. Similar behavior has been observed in graph neural networks [57], where the trigger position did not result in more effective backdoors. On the other hand, in text classification, the attack performed similarly for all the trigger positions for the simple CNN, but the "end" was slightly more effective when the GloVe embedding was used. In sound classification, the trigger was more effective in the end if LSTM was used but had no difference for CNN. These differences suggest a potential beneficial use case for backdoor attacks in general. In this case, we can use them to understand better how and what our models learn. Such an approach complements the work described in [61], where the authors drew valuable insights about the input's crucial features after graying out small square areas of the input images.

The clean-label attack is challenging in image, text, and sound classification. However, in some cases, it may be successful just by using a large trigger without having to poison more data. Additionally, if the trigger encloses more information, the clean-label's performance can be improved. We verified this for the CIFAR10 dataset, where we injected our trigger in all three image channels. We believe that the dataset influences the performance of this attack. If each element contains many features, the model will require a large poisoning rate to perceive the trigger as a feature of this class. In the clean-label attack, the trigger is injected only in elements from the target class, and it is not easy to overcome the effect of the actual features of this class. This was highlighted in the AG News and IMDB datasets in text classification. On the other hand, the simple backdoor attack can be very effective with just a few poisoned samples in all the applications we tried.

As a general remark, we believe that the backdoor is easier inserted into models that can overfit small subsets of their datasets. Models with strong generalizations are more robust against data poisoning backdoor attacks. We verified this behavior using simple CNN in text classification. In that case, our attack could not reach an attack success rate larger than 80% even with the simple attack, as the model is very simple and the learned function is very smooth. Finally, as our experiments are far from exhaustive, our findings should be taken as indications, not definitive conclusions.

In summary, the key takeaways are:

- The backdoor attack is a realistic and stealthy threat.
- As expected, increasing the poisoning rate and using larger triggers leads to higher ASR.

– Different models can behave differently during the attack even though we use the same data. Therefore, we can use the backdoor attacks as a tool for explainable AI.
– In [47], the authors claimed that the clean-label attack is not very effective. In most cases, this is true, but we saw that we could make it effective with more sophisticated triggers.
– The backdoor is easier for models that can overfit a small subset of their datasets.

5 Conclusions

Recent trends in machine learning lead to novel attack vectors like the backdoor attack. This attack is very dangerous as it can compromise AI-powered systems. Naturally, the backdoor attack also attracted significant attention, resulting in numerous novel attack and defense versions. In this work, we explored the effects of various trigger characteristics on the backdoor's performance in four domains. Our results show that deploying backdoor attacks is relatively easy for all investigated domains. There are sufficient commonalities between the attacks in different domains to ease their deployment in real-world applications and devise novel, more generic defenses.

References

1. Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F.M., Weber, G.: Common voice: a massively-multilingual speech corpus (2019). http://arxiv.org/abs/1912.06670
2. Bagdasaryan, E., Shmatikov, V.: Blind backdoors in deep learning models. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 1505–1521. USENIX Association (2021). https://www.usenix.org/conference/usenixsecurity21/presentation/bagdasaryan
3. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: International Conference on Artificial Intelligence and Statistics, pp. 2938–2948. PMLR (2020)
4. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., Srivastava, B.: Detecting backdoor attacks on deep neural networks by activation clustering (2018). arXiv preprint arXiv:1811.03728
5. Chen, X., Salem, A., Chen, D., Backes, M., Ma, S., Shen, Q., Wu, Z., Zhang, Y.: BadNL: Backdoor attacks against NLP models with semantic-preserving improvements. In: Annual Computer Security Applications Conference, pp. 554–569 (2021)
6. 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
7. Costales, R., Mao, C., Norwitz, R., Kim, B., Yang, J.: Live trojan attacks on deep neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 796–797 (2020)

8. Dahl, G.E., Stokes, J.W., Deng, L., Yu, D.: Large-scale malware classification using random projections and neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3422–3426. IEEE (2013)
9. Dai, J., Chen, C., Li, Y.: A backdoor attack against LSTM-based text classification systems. IEEE Access **7**, 138872–138878 (2019)
10. de Andrade, D.C., Leo, S., Viana, M.L.D.S., Bernkopf, C.: A neural attention model for speech command recognition (2018)
11. Deng, J., Dong, W., Socher, R., Li, L., Kai Li, Li Fei-Fei: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
12. Dikmen, M., Burns, C.M.: Autonomous driving in the real world: experiences with tesla autopilot and summon. In: Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, pp. 225–228 (2016)
13. Dodge, S., Karam, L.: A study and comparison of human and deep learning recognition performance under visual distortions. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1–7. IEEE (2017)
14. Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., Pontil, M.: Bilevel programming for hyperparameter optimization and meta-learning. In: International Conference on Machine Learning, pp. 1568–1577. PMLR (2018)
15. Gao, Y., Doan, B.G., Zhang, Z., Ma, S., Zhang, J., Fu, A., Nepal, S., Kim, H.: Backdoor attacks and countermeasures on deep learning: a comprehensive review (2020). arXiv preprint arXiv:2007.10760
16. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: Strip: a defence against trojan attacks on deep neural networks. In: Proceedings of the 35th Annual Computer Security Applications Conference, pp. 113–125 (2019)
17. Gilbert, E.N.: Random graphs. The Annals of Mathematical Statistics **30**(4), 1141–1144 (1959). https://doi.org/10.1214/aoms/1177706098
18. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 6645–6649. IEEE (2013)
19. Gu, T., Liu, K., Dolan-Gavitt, B., Garg, S.: BadNets: Evaluating backdooring attacks on deep neural networks. IEEE Access **7**, 47230–47244 (2019). https://doi.org/10.1109/ACCESS.2019.2909068
20. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
22. Hong, S., Carlini, N., Kurakin, A.: Handcrafted backdoors in deep neural networks (2021). arXiv preprint arXiv:2106.04690
23. IBM: Natural language processing (2021). https://www.ibm.com/cloud/learn/natural-language-processing. Accessed 27 July 2022
24. Karlsen, S.S.: Automated Front Detection-Using computer vision and machine learning to explore a new direction in automated weather forecasting. Master's Thesis, The University of Bergen (2017)
25. Khan, A.I., Al-Habsi, S.: Machine learning in computer vision. Proc. Comput. Sci. **167**, 1444–1451 (2020)
26. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
27. Koffas, S., Picek, S., Conti, M.: Dynamic backdoors with global average pooling (2022). arXiv preprint arXiv:2203.02079
28. 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, pp. 57–62. WiseML '22, Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3522783.3529523

29. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017). https://doi.org/10.1145/3065386
30. Li, S., Xue, M., Zhao, B.Z.H., Zhu, H., Zhang, X.: Invisible backdoor attacks on deep neural networks via steganography and regularization. IEEE Trans. Depend. Secure Comput. **18**(5), 2088–2105 (2020)
31. Li, Y., Jiang, Y., Li, Z., Xia, S.T.: Backdoor learning: a survey. IEEE Transactions on Neural Networks and Learning Systems (2022)
32. Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks. In: NDSS (2018)
33. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 142–150. Association for Computational Linguistics, Portland, Oregon, USA (2011). http://www.aclweb.org/anthology/P11-1015
34. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013). arXiv preprint arXiv:1301.3781
35. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: TUDataset: A collection of benchmark datasets for learning with graphs. In: ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020) (2020). www.graphlearning.io
36. Mubin, N.A., Nadarajoo, E., Shafri, H.Z.M., Hamedianfar, A.: Young and mature oil palm tree detection and counting using convolutional neural network deep learning method. International J. Remote Sensing **40**(19), 7500–7515 (2019)
37. Nelson, B., Barreno, M., Jack Chi, F., Joseph, A.D., Rubinstein, B.I.P., Saini, U., Sutton, C., Tygar, J.D., Xia, K.: Misleading Learners: Co-Opting Your Spam Filter, pp. 17–51. Springer US, Boston, MA (2009). https://doi.org/10.1007/978-0-387-88735-7_2
38. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014). http://www.aclweb.org/anthology/D14-1162
39. Prabhu, V.U., Birhane, A.: Large image datasets: a pyrrhic win for computer vision? CoRR abs/2006.16923 (2020). https://arxiv.org/abs/2006.16923
40. Rijsdijk, J., Wu, L., Perin, G., Picek, S.: Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. IACR Trans. Cryptograp. Hardw. Embedd. Syst. **2021**(3), 677–707 (2021). https://doi.org/10.46586/tches.v2021.i3.677-707
41. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–93 (2008)
42. Severi, G., Meyer, J., Coull, S., Oprea, A.: Explanation-Guided backdoor poisoning attacks against malware classifiers. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 1487–1504. USENIX Association (2021). https://www.usenix.org/conference/usenixsecurity21/presentation/severi
43. Shokri, R., et al.: Bypassing backdoor detection algorithms in deep learning. In: 2020 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 175–183. IEEE (2020)
44. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
45. Sun, Z., Kairouz, P., Suresh, A.T., McMahan, H.B.: Can you really backdoor federated learning? (2019). arXiv preprint arXiv:1911.07963
46. Trigueiros, P., Ribeiro, F., Reis, L.P.: Hand gesture recognition system based in computer vision and machine learning. In: Developments in Medical Image Processing and Computational Vision, pp. 355–377. Springer, Berlin (2015)
47. Turner, A., Tsipras, D., Madry, A.: Label-consistent backdoor attacks (2019). arXiv preprint arXiv:1912.02771
48. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. International Conference on Learning Representations (2018). https://openreview.net/forum?id=rJXMpikCZ. Accepted as poster

49. Vinyes Mora, S.: Computer vision and machine learning for in-play tennis analysis: framework, algorithms and implementation. Ph.D. Thesis, Imperial College London (2018)
50. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 707–723. IEEE (2019)
51. Wang, H., Mazari, M., Pourhomayoun, M., Smith, J., Owens, H., Chernicoff, W.: An end-to-end traffic vision and counting system using computer vision and machine learning: the challenges in real-time processing. SIGNAL 2018 Editors, p. 13 (2018)
52. Wenger, E., Passananti, J., Bhagoji, A.N., Yao, Y., Zheng, H., Zhao, B.Y.: Backdoor attacks against deep learning systems in the physical world. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6206–6215 (2021)
53. Wiley, V., Lucas, T.: Computer vision and image processing: a paper review. Int. J. Artif. Intell. Res. **2**(1), 29–36 (2018)
54. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google's neural machine translation system: bridging the gap between human and machine translation (2016). arXiv preprint arXiv:1609.08144
55. Xi, Z., Pang, R., Ji, S., Wang, T.: Graph backdoor. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 1523–1540 (2021)
56. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms (2017)
57. Xu, J., Xue, M., Picek, S.: Explainability-based backdoor attacks against graph neural networks. In: Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning, pp. 31–36 (2021)
58. Yang, Z., Iyer, N., Reimann, J., Virani, N.: Design of intentional backdoors in sequential models (2019). arXiv preprint arXiv:1902.09972
59. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
60. Yunchao, G., Jiayao, Y.: Application of computer vision and deep learning in breast cancer assisted diagnosis. In: Proceedings of the 3rd International Conference on Machine Learning and Soft Computing, pp. 186–191 (2019)
61. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision, pp. 818–833. Springer, Berlin (2014)
62. 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 (ICASSP), pp. 2560–2564. IEEE (2021)
63. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
64. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in Neural Information Processing Systems, vol. 28 (2015)
65. Zhang, Z., Jia, J., Wang, B., Gong, N.Z.: Backdoor attacks to graph neural networks. In: Proceedings of the 26th ACM Symposium on Access Control Models and Technologies, pp. 15–26 (2021)