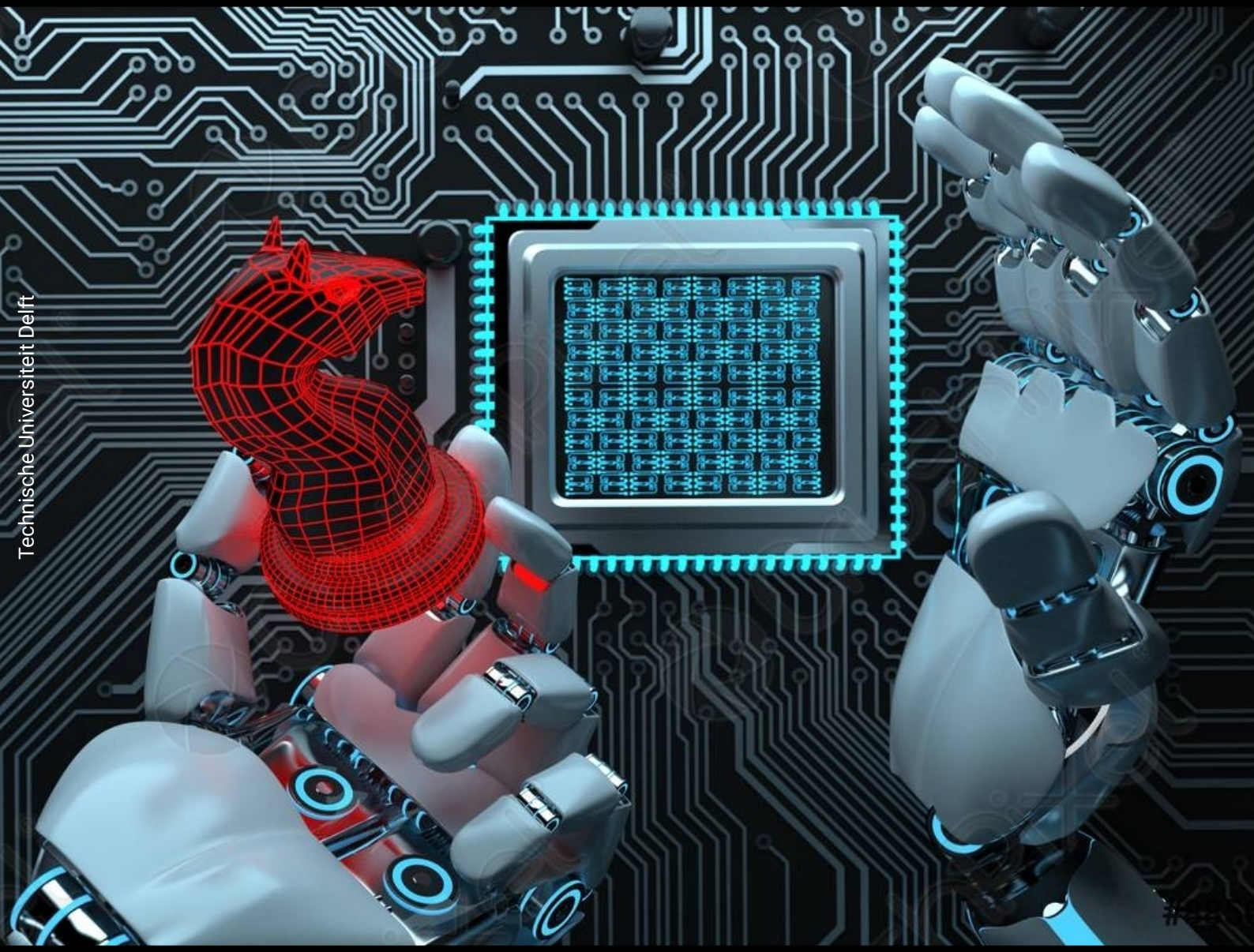
Backdoor Attacks in Neural Networks

Stefanos Koffas



Technische Universiteit Delft

TU Delft

Backdoor Attacks in Neural Networks

by

Stefanos Koffas

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on June 4 at 10:00 AM.

Student number:	4915747	
Project duration:	May 1, 2020 – May 30, 2021	
Thesis committee:	Assistant Prof. Dr. ir. S. Picek,	TU Delft, supervisor
	Prof. Dr. ir. R. L. Lagendijk,	TU Delft
	Assistant Prof. Dr. M. L. Tielman,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

Deep learning has made tremendous success in the past decade. As a result, it is becoming widely deployed in various safety and security-critical applications like autonomous driving, malware detection, fingerprint identification, and financial fraud detection. It was recently shown that deep neural networks are susceptible to multiple attacks and pose a serious security threat for such systems. Such an attack is the backdoor attack. In this attack, the model's training is partially or fully outsourced to a malicious party. The adversary designs a secret property, the trigger, and embeds it into a small fraction of the training data. The trained model associates this property with a specific (chosen by the adversary) target label. The backdoored model behaves well on regular inputs, but a misclassification will occur with a very high probability when the trigger is present.

The advent of this attack sparked an arms race in the deep learning community resulting in numerous backdoor attacks and defenses during the last years. In this thesis, we conduct a systematic evaluation that can push our knowledge further, aiming at more effective defenses in the future for various applications like image classification, natural language processing, and sound recognition. We show that the trigger's size is positively correlated to the attack success rate in almost all cases. On the contrary, the trigger's position is not always connected to the attack success rate and depends on the used neural network.

Furthermore, we are the first to experiment with inaudible triggers in sound recognition and show that they can be used for stealthy real-world attacks. Moreover, we show that backdoor attacks could be a framework that further pushes our knowledge of how deep neural networks learn.

Additionally, we exploited global average pooling's properties aiming at a more effective attack. In particular, we created dynamic backdoors that can be effective even if different trigger positions are used during training and inference without poisoning more data or altering the poisoning procedure. The exploration of this layer also showed that when a model generalizes well on new inputs, it is less probable to be susceptible to backdoor attacks.

Preface

I am happy to present you with my thesis on "Backdoor Attacks on Neural Networks". This painful but quite rewarding journey that lasted about a year and during which I had the chance to delve deep into the fascinating world of AI security would not have been possible without the support and help of the following individuals.

First and foremost, I would like to thank my supervisor Stjepan for his invaluable support and help throughout this demanding project. Not only did you show genuine interest in the scientific part of my work, but you also stood by me as a friend during the (numerous) stressful moments I was faced with during my thesis. I am really grateful for the chance I had to work with you. You have been a great mentor.

Second, I would like to thank my family, friends, and girlfriend. Carrying out my thesis against a backdrop of a developing pandemic and strict lockdowns has been a very challenging mental exercise. Thank you for being there for me, calming me down, and alleviating my anxiety and stress.

Stefanos Koffas
Delft, May 30, 2021

Contents

1

Introduction

Deep learning dates back to the 1940s [41, p. 12], but its complexity and the amount of compute required for training simple neural networks discouraged researchers in the past. However, recent progress in the underlying hardware and the introduction of general-purpose graphic processing units (GPGPUs) has led to an AI revolution. Now, deep learning models are widely deployed in a variety of demanding applications like computer vision [50, 66], speech recognition [45, 95], machine translation [120], and game-playing [104]. Deep learning is also used in various safety and security critical applications like autonomous driving [12, 35], malware detection [31, 79], fingerprint identification [116], and financial fraud detection [64].

Safety and security critical systems require guarantees about all the components' security and robustness because attackers are more likely to attack their weakest points [102]. Szegedy et al. [107] pointed out that neural networks are not robust due to adversarial examples, which are inputs that can be wrongly classified with very high probability. Such inputs are created by adding slight distortions in the direction of the gradient of the network's loss function and can affect a large variety of models [20, 92] and real-world settings [16, 103]. An adversarial example on ImageNet is shown in Figure 1.1, where imperceptible distortion changes the model's decision from a panda with 57.7% confidence to a gibbon with 99.3% confidence.

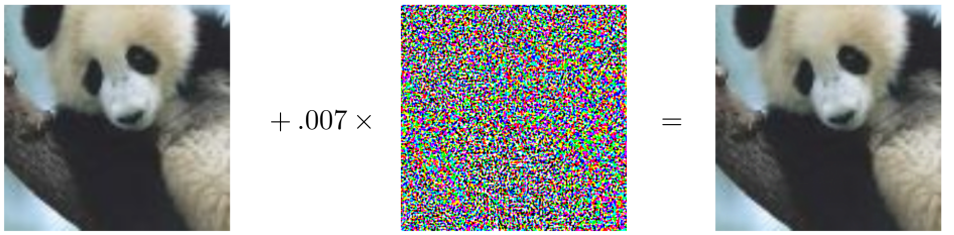


Figure 1.1: A carefully crafted distortion generated from the cost function's gradients can affect the model's decision. In this example, the network wrongly classifies an input image of a panda to a gibbon (with 93% confidence) by adding an invisible to humans distortion [42].

Inference is not the only unsafe part of machine learning pipelines. Systems like honeypots and chatbots continuously collect data for training, allowing adversaries to manipulate the training dataset [11, 119]. Additionally, state-of-the-art models use enormous datasets curated through web scraping and crowdsourcing from possibly untrusted sources [34]. This amount of data is almost impossible to check manually, and malicious actors could easily embed their data [96]. As a result, a new kind of attack has emerged; the poisoning attack. In poisoning attacks, the adversary contaminates the training dataset, which leads to a corrupted model. Two different attacks have been defined in the literature: the indiscriminate and the targeted attack [56, 67]. The indiscriminate attack aims to decrease the model's performance for all inputs causing a denial of service. On the other hand, the targeted attack aims only at the misclassification of specific inputs.

A special case of the targeted poisoning attacks is the backdoor attack, initially defined in [22, 47]. In this attack, the model's training is partially or fully outsourced to a malicious party. The adversary designs a secret property, the trigger, and embeds it into a small fraction of the training data. The trained model associates this property with a specific (chosen by the adversary) target label. The backdoored model behaves well on regular inputs, but a misclassification will occur with a very high probability when the trigger is present. For example, an adversary could force an autonomous vehicle to misclassify a stop sign as a speed limit sign when the trigger (a small yellow sticker) is present (Figure 1.2). This misclassification poses a severe threat to the passengers' safety. Furthermore, an adversary could bypass a backdoored face recognition system by wearing a specific pair of sunglasses [22]. These attacks highlight the need for verifiable security in neural networks and artificial intelligence in general [47].

Figure 1.2: Misclassification of a stop sign with a trigger as a speed limit by an autonomous vehicle [47]

Backdoor attacks are very practical due to a new threat model that has emerged from the paradigm shift to virtualization and cloud computing. In particular, the vast amount of data and the complexity of modern deep learning models make training a very demanding task that could last weeks, even if specialized hardware is used. Only a few have access to such hardware, and usually outsourced training to the cloud is preferred. Outsourced training is cheaper due to the pay-as-you-go billing model, which does not require a substantial initial investment in expensive infrastructure. Outsourced training is often referred in the literature as Machine Learning as a Service (MLaaS) [47], and it is offered by all major cloud computing platforms like AWS [6], Microsoft Azure [84], Google Cloud [43], and IBM Cloud [53]. The cost reduction from outsourced training is considerable, but the trained model's integrity cannot be verified exhaustively. Thus, an adversary can interfere with the process and hide a backdoor in the generated model.

Another cost reduction method that has recently gained attention is transfer learning. In transfer learning, pre-trained deep and complex models like AlexNet [66], VGG [105], and ResNet [50], are fine-tuned to perform a different task. This method drops the training time significantly as only a fraction of the model's weights must be updated. Like MLaaS, transfer learning poses new threats as an adversary could embed a backdoor in the pre-trained model that could exist even after the fine-tuning [47].

These new threat models triggered an arms race in the deep learning community resulting in numerous backdoor attacks and defenses during the last years. Most of the attacks target computer vision applications [22, 70, 71, 75, 100, 112], and a smaller number of them target NLP [21, 32] and sound recognition systems [75, 95]. Various defenses have been designed [18, 19, 23, 38, 111, 115], but most are empirical and do not provide strong security guarantees.

In this thesis, we decided to step back from this cat and mouse game of evolving attacks and defenses. Thus, we conduct a systematic evaluation that can push our knowledge further, aiming at more effective defenses in the future across various applications (computer vision, text sentiment analysis, and sound recognition). As a result, we came up with the following research questions:

1. How the attack accuracy is affected by various trigger characteristics in image recognition, text sentiment analysis, and sound recognition?

2. Is it possible to exploit a global average pooling layer to make dynamic backdoors in different applications like image classification, text sentiment analysis, and sound recognition without poisoning more data?

The thesis is structured in the following way. In chapter 2, we present a brief summary of the background required for this project. In particular, we cover machine learning and deep learning basics, discuss image classification, natural language processing, and sound recognition. Finally, we present some details about backdoor attacks in software, hardware, and neural networks. chapter 3 discusses the related work and presents a more detailed version of our research questions. In chapter 4 and chapter 5, we show our main contributions from this project. chapter 4 presents a detailed study on the effect of different trigger characteristics on the backdoor attack. chapter 5 shows how a global averaging pooling layer can facilitate the implementation of dynamic triggers. chapter 6 formally addresses our research questions and discusses our project's limitations and future directions.

2

Background

In this chapter, we present the background knowledge required to follow this research project. We begin with a short introduction to machine learning, emphasizing deep learning and its application to image classification, natural language processing, and sound recognition. We continue explaining backdoor attacks in general, and we conclude by describing the backdoor attacks in machine learning.

2.1. Machine Learning

Machine learning is the process of learning from data. As it was stated in [86], "a machine is said to learn from experience E for some class of tasks T and performance measure P if its performance at tasks in T as measured by P, improves with experience E." Task T can be anything. Some examples are image classification and object detection, [37], sound recognition [4, 5], machine translation [120] and anomaly detection [3]. The performance measure P is a way to measure machine learning's success and is usually closely coupled with task T. For instance, in a classification task, such a performance metric is the correct predictions over the total predictions. This metric is called accuracy. Experience E is the data that is used for the training. The data is either labeled or unlabeled [41, p. 102]. Each sample in a labeled dataset is associated with a class or a target. Labeled data guides the learning algorithm, and the whole learning process is called supervised learning. On the other hand, unlabeled data let the algorithm extract knowledge from the given dataset without guidance. This process is called unsupervised learning [46].

This project targets classification tasks. A classification task is a function approximation problem of an unknown function f ($y = f(\mathbf{x})$) from a given labeled dataset. The approximated function is often denoted as $\hat{f}$ and is used to make predictions $\hat{y} = \hat{f}(\mathbf{x})$ [87]. The function $\hat{f}$ should generalize well on novel data and should not be too specific based on the training dataset. The initial dataset is split into two different groups. The first group is used for the training process (training data), and the second to measure the model's performance in unseen data (test data). From these two groups, the training and the test error are calculated. The training error shows how well the model classifies elements from the training dataset, and it should be kept low. The test error shows the model's performance for novel input, and it should also be low. When the training error is not low enough, we have underfitting. If the gap between these two errors is large, we have overfitting because the model is too specific and does not generalize well.

2.2. Deep Learning

Deep learning is a subset of machine learning that is inspired by biological neurons. Deep learning algorithms are modeled as neural networks. Each neural network is a structure of connected neurons grouped in different layers. Such a structure with two hidden layers, three inputs, and two outputs is shown in Figure 2.1. This figure shows a multilayer perceptron consisting of fully connected layers (each neuron in one layer is connected to all the neurons in the next).

The fundamental component of a neural network is the neuron. As shown in Figure 2.2, the inputs x_i of each neuron k are multiplied with the corresponding weights w_{ki} and then summed together. Next, the bias b_i is added, and the result is passed through an activation function $\sigma()$. The equation that dictates the functionality of a neuron is shown in Equation 2.1. The weights w_{ki} and the biases b_i are the network parameters. Their initial values depend on the initializer (distribution that we draw values from) that we choose, and their

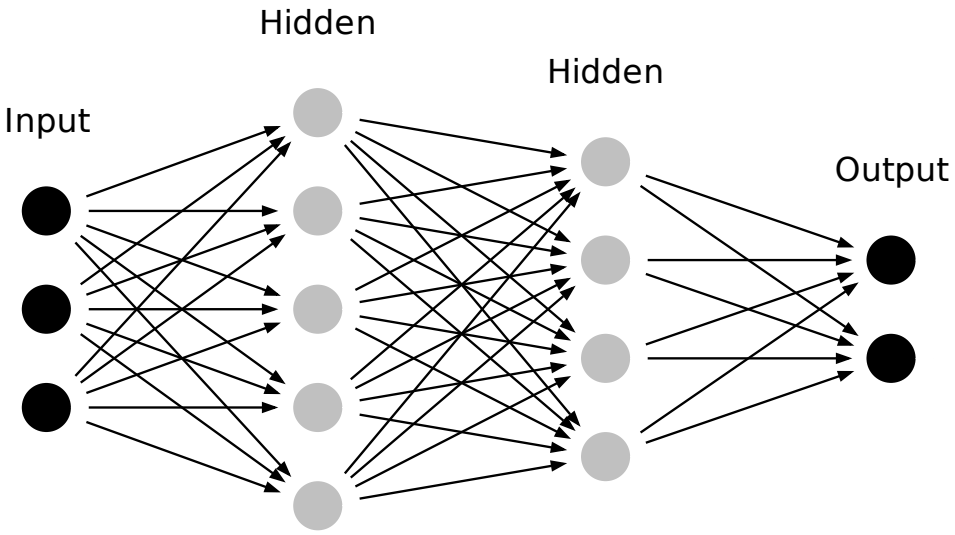


Figure 2.1: Example neural network with 3 inputs, 2 hidden layers and 2 outputs

final values are learned through the training. Many initializers are available. In this project, however, we use the default Tensorflow initializer, the Glorot uniform initializer.

$$y_k = \sigma(\sum_{n=1}^{m} w_{ki} \cdot x_i + b_i) \tag{2.1}$$

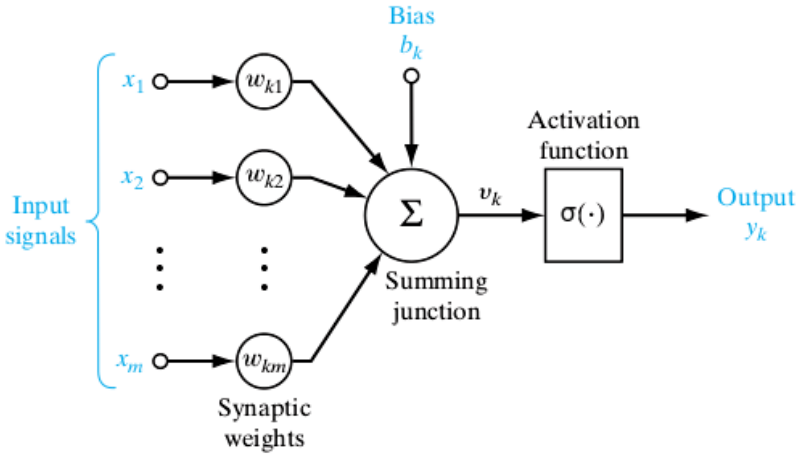


Figure 2.2: Computations for every single neuron [80]

The activation function $\sigma()$ is usually non-linear because this is the only way to approximate complex functions with a neural network. If all the activation functions are linear, the whole neural network will result in just one linear transformation of its input regardless of its depth and structure. We used Rectified Linear Unit (ReLU), Exponential Linear Unit (ELU), and softmax in this project.

ReLU

ReLU (Subfigure 2.3(a)) [88] is a straightforward and efficient non-linear function with the formula shown in Equation 2.2.

$$ReLU(x) = max(0, x) \tag{2.2}$$

ELU

ELU (Subfigure 2.3(b)) [24] is the same as ReLU for positive inputs, but it is non-zero for negative inputs. Its formula is shown in Equation 2.3.

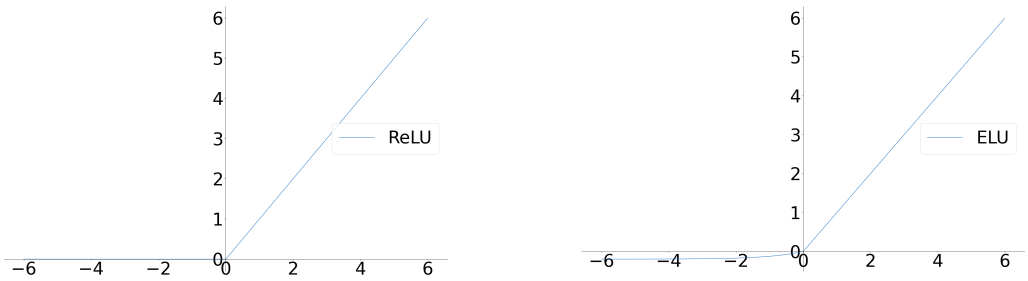
$$ELU(x) = \begin{cases} \alpha \cdot (e^x - 1), & if\, x < 0 \\ x, & if\, x \geq 0 \end{cases} \tag{2.3}$$

Softmax

The softmax activation function s converts the input vector to a vector of probabilities. The elements of this new vector are between 0 and 1, and their sum is always 1. This function is used in the last layer because its output shows which classes are probable. The function $s : \mathbb{R}^K \rightarrow [0,1]^K$ is defined by the formula shown in Equation 2.4.

$$s(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.4}$$

for $i = 1, ..., K$ and $x = (x_1, ..., x_K) \in \mathbb{R}^K$



(a) ReLU

(b) ELU with $\alpha = 0.2$

Figure 2.3: Activation functions

Training and Optimization

The training of a classifier is an optimization problem. In this problem, two different functions are defined. The first is a parameterized score function that generates the class scores from the input data. Its parameters are the network's weights and biases, usually randomly initialized with values drawn from a Gaussian or a uniform distribution [41, p.293]. The second is the loss function and shows the quality of a particular set of parameters by comparing the model's output with the provided ground truth labels. In this project, we use the cross-entropy as a loss function which is widely used for classification tasks [51, 124]. *Gradient descent* is the algorithm that iteratively improves the parameters until the loss is minimized. As its name suggests, it updates the weights in the opposite direction of the loss function's gradient. Equation 2.5 dictates this process. The hyperparameter γ affects each step's size and is called the learning rate. If it is not big enough, the training time increases significantly, but the model may not converge in the opposite case. The backpropagation algorithm calculates the loss function's gradient with respect to the weights based on the chain rule's recursive application.

$$w_{n+1} = w_n - \gamma \nabla L(w_n) \tag{2.5}$$

In many cases, the loss function can be very sensitive to some parameters and insensitive to others. Thus, adaptive methods have been developed to update the learning rate differently for each direction during training [41, p. 298]. The optimization methods used in this project are RMSProp [110] and Adam [62], which are widely used in practice and improve the results significantly [41, p. 302].

To avoid overfitting in our experiments, we use three popular techniques: batch normalization, weight regularization, and dropout. Batch normalization [54] acts as a regularizer and decreases the effects of bad weight initialization. In practice, it is added as an extra layer in the neural network's architecture. L_2 weight regularization is widely used [105] and penalizes large weights by adding the term $\frac{1}{2}\lambda \cdot w^2$ in the network's loss function for every weight w. Dropout [106] is very popular in the literature [30, 49, 66] and it randomly drops a set of neurons and their connections in each training step updating only the remaining weights. It is a simple and very effective way of regularization [41, p. 251].

Convolutional Neural Networks

Convolutional neural networks are neural networks that use at least one convolution layer [41, p. 321]. This layer performs the convolution operation, which is very convenient for 2D inputs like images. Convolution uses filters (kernels) which are small matrices of learnable parameters. Each filter is usually small (3x3 or 5x5), but it has the same depth as the input. Thus, if the input is a 3-dimensional image (one channel per color), the filter should also be 3-dimensional. Each filter slides over the image with a step size defined from the kernel's stride and calculates a dot product of its parameters and the corresponding input pixels. This operation results in a new 2D matrix, the feature map. In Figure 2.4, an example of the convolution of a 6x6 image with a 3x3 kernel is shown.

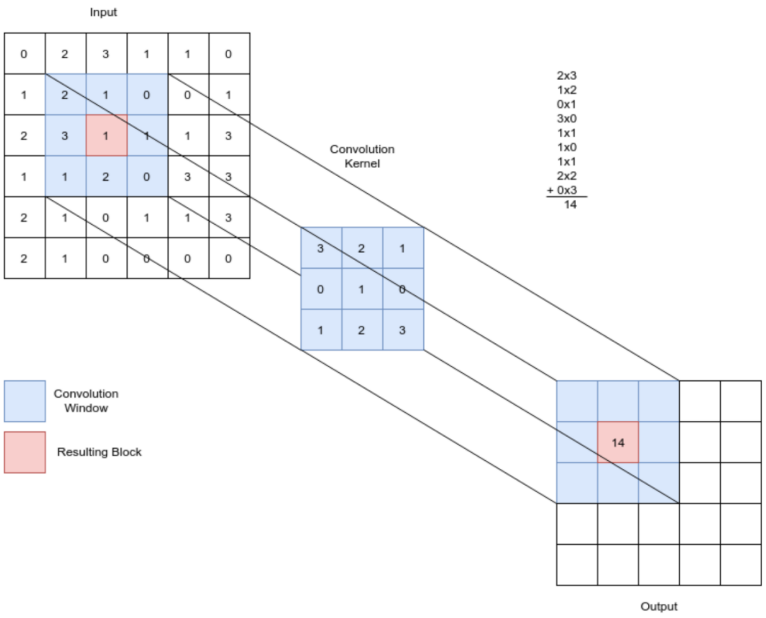


Figure 2.4: Convolution of a 6x6 input with a 3x3 filter.

The convolution layer is often followed by a pooling layer [41, p. 330]. This layer improves the network's generalization and makes it more robust to small input changes. It combines small subsets of the feature map by taking the maximum (max pooling) or their average (average pooling). Similarly to convolution, it is characterized by a kernel and its stride. The kernel does not have learnable parameters. It just shows which parts of the feature map are used. The stride shows how to move the kernel for each operation. In this project, we use the max-pooling layer. Figure 2.5 shows how a 2x2 max-pooling layer with stride 2 transforms a 4x4 feature map to a 2x2.

Figure 2.5 shows how a 4x4 feature map is transformed from a max-pooling layer with a 2x2 kernel and a stride of 2 in each direction.

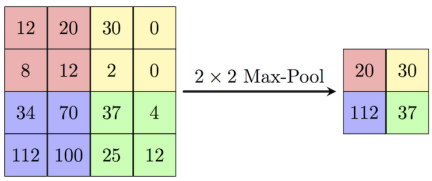


Figure 2.5: Max pooling operation with a 2x2 kernel and a stride of 2 in both axes.

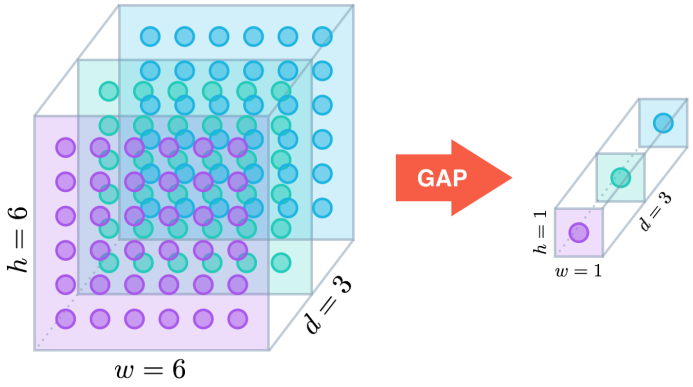


Figure 2.6: Transformation of a 6x6x3 tensor to 1x1x3 with 2-dimensional global average pooling [26].

2.2.1. Global Average Pooling

Global Average Pooling (GAP) was first introduced in [73], and it is frequently used instead of a fully connected layer as a penultimate layer. As its name suggests, it calculates a global average of the feature map, which is then fed directly to the next layer. In TensorFlow [1] there are three variations (`GlobalAveragePooling1D()`, `GlobalAveragePooling2D()` and `GlobalAveragePooling3D()`) of this layer according to the shape of the input tensor. `GlobalAveragePooling1D()` averages a 3D tensor with shape (batch_size, steps, features) over the number of steps and produces a 2D tensor with shape (batch_size, features). Figure 2.6 shows how a 6x6x3 tensor is transformed to 1x1x3 by a 2-dimensional global average pooling layer. The other two variations work similarly but for 4D and 5D input tensors, respectively. This layer does not have any trainable parameters, so it reduces overfitting in the last layers. Additionally, it tends to discard any spatial information from the last layers [73].

2.2.2. Image Classification

Computer vision is one of the most studied fields in artificial intelligence, and it is used in many applications like mobile apps, medicine, and autonomous vehicles. It studies the processes that a computer can use to understand images and videos. The most known visual recognition tasks are image classification, object localization, and detection. In this project, we will experiment with image classification tasks.

In image classification tasks, convolutional neural networks are prevalent [34, 50, 66, 93, 105]. Images are very convenient data structures that can be directly fed into a deep learning system without any preprocessing, avoiding the need for feature engineering.

Dataset

CIFAR-10 is perfectly balanced and consists of 60000 (50000/10000 training/test) 32x32x3 RGB images in 10 classes (6000 images per class). The ten classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck [65].

2.2.3. Natural Language Processing (NLP)

Natural language processing is an intersection of linguistics, computer science, and artificial intelligence. Its goal is to create systems that can process and analyze text and extract useful information from that. The simplest form of language processing is rule-based. In that case, we define the language's grammar as a set of rules, and then the computer parses text based on these rules. Compilers parse source code in this

way. However, natural language cannot be strictly defined with a specific set of rules because it is constantly evolving. Computational linguistics is NLP that uses statistical methods, and it was introduced in the late '80s to tackle this problem [13, 14].

In recent years, deep learning became the default method for NLP [120]. To achieve high accuracy, though, the text needs to be transformed conveniently for processing instead of a raw sequence of characters (bytes). In this project, we use word Embeddings for this task. An Embedding translates each word to an n-element vector. In other words, it maps every word to an n-dimensional space based on its meaning and its neighboring words [57]. The exact values of the vector's elements are either randomly initialized and updated during training or loaded from a pre-trained Embedding [85, 94]. The value n is the Embedding's width and is connected with its complexity. A large n could encode much more information about each word. However, we should be careful when choosing its value as a very large n could make the representation too specific.

Various types of neural networks can extract useful features from an embedding. The most common approaches are convolutional neural networks [61] and recurrent neural networks [7]. Recurrent neural networks process sequences of data and base their outputs on an internal state register. In this project, we experimented with text sentiment analysis and used convolutional neural networks that are widely used in the literature [7, 69, 75, 122].

Dataset

In this project, we use IMDB's text sentiment analysis dataset [78], which consists of 50000 highly polar, either too negative or too positive, movie reviews. 25000 reviews are used for training, and 25000 are used for testing. It is a perfectly balanced dataset with 25000 positive reviews and 25000 negative reviews. We used 20% of the training samples as a validation set. This dataset is convenient for our project as many security applications like spam filtering [25] and malware detection [68] use binary classification.

2.2.4. Sound Recognition

Automatic speech recognition (ASR) has gained much attention in recent years as it can be a very effective form of communication between people and machines. Voice assistants like Google Assistant and Amazon's Alexa have already shown the power of these interfaces. ASR is also used to deploy biometric systems which authenticate users based on their voices [95].

Each ASR system needs to discard unnecessary information from the input signal, like noise and voice color, and keep only the useful features that will help identify the actual linguistic content. Such features can be simple like spectrogram images [75] or other more sophisticated like the Mel-Frequency Cepstral Coefficients (MFCCs) [33] and the Mel-Cepstrum [16]. We used the MFCCs that are very popular and widely used [17, 101, 122, 126].

MFCCs are pretty accurate sound recognition features because they emulate the functionality of the human vocal system [58]. MFCC feature calculation is a multi-step process. It involves framing and windowing (breaking the input signal to multiple short frames and scaling them by a function to avoid sharp differences in the frame edges), calculating the power spectrogram of the discrete Fourier transform for each frame, mapping each spectrogram to the Mel-scale (Mel-scale is a group of bandpass filters that are logarithmic for higher frequencies and linear in the opposite case because humans can more accurately discern differences in a sound signal in smaller frequencies), and calculating the discrete cosine transform [58, 101].

The extracted MFCCs of a signal have fxt dimensions. f is the number of coefficients and t depends on the window hop, the sampling rate, and the signal length. For example, for 1 second recordings at 16 kHz (16000 samples per second) and a hop length of 512 (samples per hop), $f = \lceil 16000/512 \rceil = 31$ [101].

Dataset

In our sound recognition experiments, we used Google's Speech Commands dataset [117] which consists of 64727 1-second audio files. These audio files are spoken word recordings (1 word per recording) of 30 short words like "up", "down", "left", "right", "yes", and "no", provided by thousands of different people. The files are 16-bit mono recordings with a sampling rate of 16 kHz which means 16000 samples per file [117].

2.3. Backdoor Attacks

2.3.1. Backdoor Attacks in Software/Hardware

A backdoor attack injects a hidden malicious functionality activated by a predefined input (trigger). In any other case, the backdoored system behaves normally, making it very difficult to detect. The backdoor's activation may result in private data exfiltration [123] or privilege escalation [2, 27, 28].

Outsourced chip making is adopted by various companies in the semiconductor industry, extending the backdoor's attack surface in the hardware domain. It has raised many concerns as a considerable part of the global semiconductor production is outsourced to China or Taiwan. Malicious actors could reverse engineer a given design and embed their logic into the end product. This logic remains inactive until it is fed with a specific input (trigger). When activated, it can send private data, like passwords or authentication tokens, to the attacker (payload). Circuits cannot be verified exhaustively, and thus, hardware trojans are very difficult to detect, making them a perilous threat. There are rumors that this attack has already happened, but the corresponding parties widely refute them [99]. This attack remains mostly theoretical and popular among researchers, but it has gained much attention over the past decade [121].

2.3.2. Backdoor Attacks in AI

Originally the backdoor attack in AI was defined in [47]. In this project we use the same definition of this attack. A backdoor adversary trains a poisoned model f^{adv} that behaves differently from an honestly trained model f^*. A successful backdoor attack needs to fulfill the following two goals [47]:

1. The poisoned model's (f^{adv}) classification accuracy should be greater or equal to the clean's model (f^*) accuracy so that the backdoor remains stealthy and hidden. More formally, $acc(f^{adv}, d_{valid}) \geq acc(f^*, d_{valid})$, where d_{valid} is a validation dataset.

2. Inputs with a specific property should be classified differently from the poisoned model f^{adv} and the clean model f^*. Formally, if $P : \mathbb{R}^N \rightarrow \{0, 1\}$ is a binary function that is true (1) when the specific property exists on the input and false (0) in the opposite case, then this goal can be expressed in the following way: let x, so that $P(x) = 1$ then $argmax f^{adv}(x) \neq argmax f^*(x)$, where $argmax$ returns the class of the specific input to the classification task.

The predefined property is called trigger because it triggers the hidden functionality. This property can be a small patch with a specific shape and pattern in image classification or a rare word or phrase in text classification. This definition encloses both targeted backdoor attacks and untargeted. In the untargeted attack, the adversary's goal is to classify triggered inputs to any wrong class, which is similar to a denial of service attack. The targeted backdoor attack associates poisoned inputs with a specific class of the classification task, the target class. These backdoor attacks pose a serious threat because they can be used to bypass security-critical systems like a face recognition authentication system [22]. In such a case, the adversary could associate the triggered input with a key person like the company's CEO and gain unauthorized access. For that reason, we decided to experiment with targeted backdoor attacks in this project.

The most common metric to characterize the effectiveness of a targeted backdoor attack is the attack accuracy or the attack success rate [22, 47, 75]. To calculate this metric, we poison the test dataset with the trigger and get the predictions from the trained model. The attack accuracy is the percentage of the predictions of the target class over the total predictions performed (Equation 2.6). It is often paired with the model's accuracy on clean inputs to see how backdoor insertion affects the normal operation. A perfect backdoor achieves attack accuracy close to 100% so that poisoned inputs always trigger the hidden functionality.

$$\text{Attack accuracy} = \frac{\#\text{predictions of target class}}{\#\text{total predictions}} \tag{2.6}$$

Some terms that are frequently used in the backdoor literature are the benign model (a model with no backdoors), the infected, backdoored, or poisoned model (a model with a hidden backdoor), the poisoned sample (an element from the training or testing dataset that contains the trigger pattern), the source label (the original class that a poisoned sample belongs to) and the target label (the adversary chosen class that all poisoned samples should be classified as). The effort required by an adversary for a successful backdoor attack is expressed by the percentage of the poisoned training samples required along with the perturbation introduced by the trigger. Both of them should be kept as low as possible when designing new attacks.

3

Related Work

Modern neural networks are very complex, need enormous data to train, and require days, weeks, or even months for their training. The required infrastructure is costly and not available to everyone. Thus, application developers choose to outsource the implementation of their models to third parties or create their models by re-purposing pre-trained ones (fine-tuning). This outsourced learning introduces new security risks and a new threat model. In this model, an adversary that controls the training model potentially hides a malicious functionality that can be triggered when specially crafted inputs are fed into the system. The original task remains unaffected for normal inputs. This threat model has already been used in many research endeavors [47, 75, 77, 97, 98, 100, 112], resulting in an attack usually called neural trojan or neural backdoor attack.

The same problem arises for large datasets [34] that collect their data from untrusted sources on the web. Even though an attacker does not fully control the training process in that setup, she can still affect the training data and embed a backdoor in the trained neural network [40].

3.1. Backdoor Attacks

One of the first and most popular backdoor attacks is described in [47]. The authors introduced BadNets which are backdoored models created after poisoning some part of the training dataset. They demonstrated the practicality of this attack with an autonomous vehicle that identified a stop sign with a yellow post-it note as a speed limit sign. They were also the first to notice that a backdoor can be effective even after re-purposing a poisoned model with transfer learning.

A weaker threat model is used in [22]. In this model, the attacker can only alter very few samples of the training dataset and does not know the incorporated model or the training dataset. The authors showed that the attack is possible (> 90% attack success rate) even with 50 poisoned samples. They also showed that the attack could be applied in the real world through a 3D-printed pair of glasses.

The attacks mentioned above required changing the label of the poisoned data. Some defenses are also based on that assumption [18, 38]. Aiming for increased stealthiness and attack accuracy, researchers have designed an attack that poisons only members of the target class [112]. In this way, they create a robust feature that the model recognizes even in different inputs. This attack is called label-consistent backdoor attack because the poisoned input samples cannot be found by human inspection.

In [75], a new method for backdoor attacks was designed. First, the attacker chooses a region of pixels that is used for the trigger insertion. Then, their algorithm finds some neurons that are sensitive to these particular pixels and assigns appropriate intensity values that maximize the trigger's effect on these neurons. Access to the training dataset is not assumed, so synthetic data for each class is generated and used for the model retraining. This project is one of the first works that tested the backdoor attack in different applications like face recognition, text sentiment analysis, and sound recognition.

Similarly, in [70], they treated the trigger pattern as an optimization problem with two parts. The first is to maximize the response of a subgroup of neurons so that the trigger is effective, and the second is to keep the amount of perturbation bounded to avoid easy human inspection. With this attack, neural cleanse [115] and STRIP [38] were bypassed.

In [71], the authors showed that simple affine transformations of the input image could significantly decrease the attack success rate for various attacks [22, 47, 75]. They also claimed that it is challenging to affect a particular position of a video or image with a trigger in a real-world scenario. However, simple data augmentation techniques can increase the attack's robustness to such transformations. In [100], the authors used a conditional generative adversarial network and created dynamic backdoors for image classification tasks. However, their attack required a high poisoning rate which violates the threat model that is commonly used.

Besides computer vision, natural language processing is another field that the backdoor attacks have been applied. However, the number of research projects in that field is significantly lower [72]. In [32], the authors embedded a backdoor in a recurrent neural network (bidirectional LSTM) on a text sentiment analysis task. In BadNL [21], the authors expanded this work and investigated the effect of different triggers (letter-based, word-based, sentence-based) on the attack success rate. However, they used only a recurrent neural network for their attacks which was not very effective for text sentiment analysis. In [75], the authors applied their attack on a text sentiment analysis application, and they inserted a variable-length trigger in the 25^{th} position of each sentence. The architecture that they used is a convolutional neural network defined in [61].

In Dolphin Attack [126], the authors successfully attacked sound recognition systems like Alexa, Siri, and Google Now. They successfully run malicious tasks on target systems just by playing hidden, inaudible commands over the air. Their approach resembles an adversarial attack but gave us the motivation for inaudible triggers in sound recognition systems.

Backdoor attacks in sound recognition are under-represented in the literature. In [125], the authors used poisoning-based techniques to create a backdoor against speaker verification systems, and in [75], they tested their attack to a sound recognition system.

3.2. Backdoor Defenses

Various attempts have been made to design defenses that can protect users from backdoor attacks. Neural cleanse [115] was one of the first. In this work, the authors tried to create a framework that detects and removes backdoors from trojaned models. Their framework reverse engineers potential triggers for each target class given a small number of clean samples and marks a model as backdoored if one of these triggers is significantly smaller than the others (affects a smaller number of pixels) [115]. However, the community came up with various issues with that defense. Neural cleanse depends on clean data and requires white-box access to the model. Both these assumptions cannot be valid in an MLaaS setup [19]. Additionally, there is no guarantee that the reverse engineering always succeeds with a small number of clean samples [76]. Furthermore, neural cleanse was bypassed from partial backdoor attacks [36] and in [108]. It was also vulnerable to large triggers [19, 36, 122]. Last but not least, this technique cannot be transferred to other applications like NLP and sound recognition [7, 122].

In DeepInspect [19], the authors built on Neural Cleanse [115] and created a faster defense with a stricter threat model. More specifically, they did not assume white-box access to the model, and this defense does not require access to clean data. The trojan detection involves three steps: model inversion, trigger generation with a conditional Generative Adversarial Network (GAN), and anomaly detection. However, this defense cannot defend from an all-to-all attack (all the classes act as a source class and a target class), and it is unsuitable for classification tasks with only two classes [122].

Activation clustering defense [18] is based on the different activations for poisoned and clean inputs. The authors observed that a backdoored model recognizes both the trigger and features from the target class when the input is poisoned, which is not happening for clean inputs. This information is encoded in the activations of the last layer, and by applying dimensionality reduction techniques and a clustering algorithm to the activations of both clean and poisoned inputs, this can be pretty clear. However, this defense assumes white-box access to the model and the training dataset [122] and has already been compromised [108, 109].

Similarly, in [111], the authors cleaned a poisoned dataset by observing input samples' learned representations (spectral signatures). The trigger left a strong trace in these representations, and they were able to identify the poisoned inputs inside the dataset successfully. However, they assumed that the defender has access to both clean and trojaned inputs, which is a very strong assumption [47]. Additionally, this defense was bypassed in [108] by using an additional component in the model's loss function, penalizing significant differences in the learned representations.

Sentinet [23] uses GRAD-cam to find regions (patches) in the input that contribute the most to the model's decision. Then it verifies if these patches are the triggers by applying them on clean inputs. This defense is only applicable in image backdoors [7, 122], cannot defend against large triggers and partial backdoor attacks

[39], and was bypassed in [8, 109]. Additionally, disjoint patches may also bypass this defense [39].

STRIP [38] is a simple and effective defense that can be applied easily as a run-time defense that treats a given model as a black box. It mixes each input with a small set of clean samples and observes the entropy of these classifications. If the entropy is low, the network gives a confident prediction even for random patterns. In that case, it raises an "alarm" that a poisoned sample is fed into the system. This defense was bypassed in [29, 70] and could not defend against partial backdoor attacks (the trigger is only effective when applied to a specific source class) [109].

A relatively recent and promising defense against backdoor attacks is described in [122]. Assuming only black-box access to a given pre-trained model (access to model's output on various inputs), the defender trains various shadow models (simple models with only moderate performance) that encode as much information as possible from the given model. Then, they trained a binary meta-classifier from these shadow models, which classifies a model either as clean or as trojaned. This defense is application-agnostic and showed promising results. However, it requires ML expertise and requires access to training which may break the MLaaS threat model [39].

This research direction is active, and the community is continuously growing. For that reason, NIST has started a new program [89] aiming to combat trojaning attacks in AI. This program includes an open competition for the best defenses against backdoored models.

3.3. Research Questions

Our understanding of this field is still limited, and most of the defenses so far seem to be empirical without any strong security guarantees. There is a lot to learn, and as it was stated in [70], we do not fully understand the internal structure of the neural networks and the reasons behind successful backdoor attacks. We also observed that most of the projects published in this field, attack computer vision applications [72, 74]. For that reason, we decided to take a step back and conduct a systematic analysis to improve our understanding of backdoor attacks in all three applications.

As stated in [71], simple affine transformations can significantly decrease the attack accuracy of static backdoor attacks in image classification. An attacker could bypass such a defense by using more poisoned samples and applying the trigger in different positions. However, this could increase the percentage of the poisoned samples used. In [100], it was shown that a dynamic backdoor attack could bypass defenses like Neural Cleanse [115] and ABS [76]. However, their dynamic backdoor attack required more than 30% poisoned data. As we stated in subsection 2.2.1, global average pooling averages a feature map discarding any spatial information, making it a perfect candidate for dynamic triggers without increasing the percentage of poisoned data. In this work, we are the first to investigate how global average pooling can be lead to dynamic backdoor attacks in image classification, text sentiment analysis, and sound recognition.

Thus, this thesis will address the following research questions:

1. How the attack accuracy is affected by various trigger characteristics in image recognition, text sentiment analysis, and sound recognition?
    - How the trigger's size and position affect the model's accuracy for the original task?
    - How different shapes in image classification affect the attack accuracy?
    - How the trigger's position affects the attack accuracy?
    - How the attack accuracy is affected by the trigger's size?
    - What is the effect of non-continuous triggers on the attack success rate?
    - Is it possible to use a trigger inaudible by humans (> 20 kHz) in sound recognition?
2. Is it possible to make more effective backdoors with global average pooling in different applications like image classification, text sentiment analysis, and sound recognition?
    - How the global average pooling layer can be added in the neural networks used in the previous research question?
    - What is the effect of the global average pooling layer on the model's original task?
    - Can we exploit the properties of the global average pooling layer to create a dynamic trigger without increasing the number of poisoned training samples or changing the poisoning procedure?

4

Backdoor Attacks in Different Applications

This chapter investigates backdoor attacks in image classification, text sentiment analysis, and sound recognition. We aim to highlight any similarities and differences for backdoor attacks in these applications. First, we describe the applications and the corresponding triggers, and then we discuss the experimental results.

4.1. Motivation

Backdoor attacks in Neural Networks have been recently studied in a plethora of research endeavors. Although image recognition tasks attract the most interest [10, 22, 47, 74, 75], an investigation of the backdoor attacks in different tasks such as text classification and sound recognition could shed light on various unexplored backdoor attacks' properties. This chapter presents various experiments performed for all three application domains (image, text, and sound). The conclusions we have drawn can lead to more effective backdoor attacks, resulting in more robust defenses in the future.

4.2. Threat Model

Enormous datasets are curated through web scraping and crowdsourcing from possibly untrusted sources [34]. This amount of data is almost impossible to check manually, and malicious actors could easily embed their data [96], resulting in a new black-box threat model for deep learning applications. In this threat model, the adversary has access to a subset of the training data. In particular, we allow at most 1.85% poisoned training samples and explore how various trigger characteristics affect the backdoor attack.

4.3. Experimental Setup

In this section, we present the experimental setup for each application and discuss the design of the corresponding trigger patterns.

4.3.1. Image Classification

Computer vision applications are used for security and safety-critical applications, such as autonomous driving and healthcare. Thus, numerous previous works on backdoor attacks have been performed in image recognition applications [10, 22, 47, 74, 75]. For our experiments, we used three different architectures so that our findings are robust. We also used the CIFAR-10 dataset [65]. The first architecture (Table 4.1) is the same architecture used in STRIP defense [38]. It consists of 3 VGG-like blocks [105], one flatten layer and a softmax layer as output. Each VGG-like block consists of 3 layers, two convolutional and one max pooling. We have also used batch normalization [54] and dropout [106] for increased generalization. In STRIP, data augmentation was also used (shifts and rotation). However, we observed that the trigger pattern was distorted a lot from that operation, so we did not use it in our experiments without affecting the model's accuracy. We used the same optimizer though which was the RMSProp [110] (learning rate = 0.001, epsilon = 10^{-6}). This architecture is not the best in the literature, but it is enough to demonstrate interesting properties of the backdoor attacks in image classification.

The second architecture that we used is quite simple but common among researchers [15, 91, 122], and it is shown in Table 4.2. It uses 2 VGG-like blocks and three fully connected layers. The filters for the convolu-

Layer	Activation Function
Convolution (32 3x3 filters)	ELU
Convolution (32 3x3 filters)	ELU
Max Pooling 2D (2x2 filter, stride 2 in each direction)	
Dropout (0.2)	
Convolution (64 3x3 filters)	ELU
Convolution (64 3x3 filters)	ELU
Max Pooling (2x2 filter, strinde 2 in each direction)	
Dropout (0.3)	
Convolution (128 3x3 filters)	ELU
Convolution (128 3x3 filters)	ELU
Max Pooling (2x2 filter, stride 2 in each direction)	
Dropout (0.4)	
Flatten	
Dense (10 classes)	softmax

Table 4.1: First architecture [38] in image classification.

tions are 64x3x3 and 128x3x3, and the last fully connected layer is the network's output. For regularization, we used batch normalization [54] after each convolution and a dropout layer right before the output. We used the Adam optimizer (learning rate 0.001) as it was faster and more stable than the RMSProp used in STRIP. Our third architecture is ResNet18 [50]. Again, we used Adam optimizer with a learning rate of 0.001.

Layer	Activation Function
Convolution (64 3x3 filters)	ReLU
Convolution (64 3x3 filters)	ReLU
Max Pooling 2D (2x2 filter, stride 2 in each direction)	
Convolution (128 3x3 filters)	ReLU
Convolution (128 3x3 filters)	ReLU
Max Pooling (2x2 filter, stride 2 in each direction)	
Flatten()	
Dense (256)	
Dense (256)	
Dropout (0.5)	
Dense (10 classes)	softmax

Table 4.2: Second architecture [15, 91, 122] in image classification.

Trigger

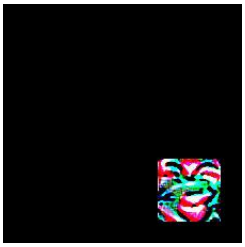
Inserting a backdoor in the predictive model involves the modification of a subset of the training data. This process is known in the literature as poisoning [55], because some samples of the "clean" dataset are "poisoned" with the attacker's specially crafted distortion, which is called the trigger.

In [75], the authors came up with an algorithm that generates triggers that activate specific neurons in a neural network. They used this algorithm to generate a trigger for a publicly available model [93]. The trigger is shown in Subfigure 4.1(a), and it was also used in STRIP [38], Neural Cleanse [115], and Sentinet [23]. Although this trigger was initially destined for a VGG-Face model [93] that was trained with Wild [52] and YouTube Faces [118] datasets, STRIP's [38] authors used it for a different architecture and the CIFAR-10 dataset [65]. For that reason, they had to resize the trigger image from 224x224x3 to 32x32x3 [1] which changed the trigger pattern substantially (Subfigure 4.1(c)). In [22], the authors used noise as a trigger (Subfigure 4.1(b)), and in BadNets [47], a yellow square or emojis of a flower and a bomb were used (Subfigure 4.1(d)). In all these cases, the backdoor attack was successful, which indicates that the attacker has plenty of freedom in the trigger pattern that she chooses. Thus, for our experiments, we decided to use random intensity values retrieved from a pseudo-random generator. The seed was hardcoded, and the same triggers were generated between different experiments.

Square triggers or small objects that can be enclosed in a square are prevalent in the literature [38, 47, 75, 98]. Other works, however, introduced triggers that resemble actual accessories like sunglasses [22]. In our work, we model these two kinds of triggers with two different shapes: a square and a line which is more similar to a pair of sunglasses.

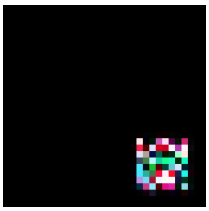
Usually, the trigger's position is decided beforehand in an arbitrary way [38, 47, 75, 98]. However, different positions may affect the attack performance. To the best of our knowledge, only in [98] this was put into practice. More specifically, the authors experimented with a 10x10x3 square trigger embedded in three different locations (upper left corner, lower right corner, and middle) of the input data. In this work, we used the

[1] OpenCV's `cv2.resize()` function

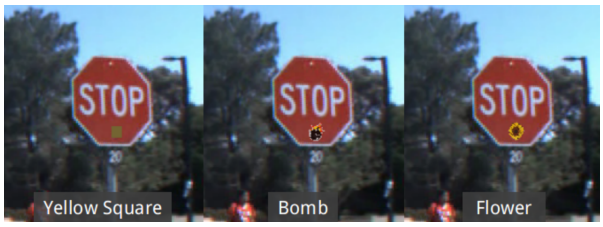


(a) Trigger used in [38, 75, 115].

(b) Random trigger used in [22].



(c) Trigger in STRIP [38].



(d) Triggers used in Badnets [47].

Figure 4.1: Triggers used in literature

same locations for our experiments. However, we have also varied the size of the trigger. In [98], the authors experimented with 8x8x3, 9x9x3, 10x10x3, and 11x11x3 square triggers but only for the lower right corner of the image. Additionally, in STRIP [38], an 8x8x3 square trigger on the lower right corner was used (Subfigure 4.1(c)). In this work, we wanted to explore also larger triggers that affect more input features. Thus, we used triggers with an area of 64 (8x8), 121 (11x11), 196 (14x14), and 289 (17x17) pixels. Additionally, to model the feasibility of multiple objects as a trigger, we have tried non-continuous patterns of the same area.

We poison the original images in the same way that was described in STRIP [38]. Some examples of 64-pixel triggers are shown in Figure 4.3. We also choose the 8^{th} class (horse) as the target class of the poisoned samples.

Our triggers are shown in Figure 4.2. The black part in the images shown in Figure 4.2 is just pixels with a value of 0. The line triggers are shown in Subfigures 4.2(1)-4.2(12), the square triggers are shown in Subfigures 4.2(13)-4.2(24) and the non-continuous triggers are shown in Subfigures 4.2(25)-4.2(28).

4.3.2. Sentiment Classification in Text

Binary classification splits a dataset into two groups according to a classification rule. This kind of classification is used for malware analysis (benign, malicious) [68] and email spam detection (spam, no-spam) [25]. In this set of experiments, we use Stanford's IMDB movie review dataset [78]. As a first step, we transform the dataset's movie reviews from strings to vectors of integers. We have deployed a text vectorization layer [2] that maps every word in each review to an integer according to its frequency. This layer removes the punctuation symbols and the special characters and transforms all the capital letters to lowercase. Additionally, to avoid having inputs of different length, we pad or truncate every vector to 250 words (features). We have chosen a vocabulary of 10000 different words, which means that only the 10000 most frequent words are considered. IMDB's vocabulary is small compared to other datasets, and this size is enough for our experiments.

Similarly to image classification, we evaluated the backdoor attack against three different architectures. The first layer in all of them is an Embedding layer (subsection 2.2.3), and the optimizer is Adam [62].

The first architecture is shown in Table 4.3, and it was taken from TensorFlow documentation [44]. Its embedding layer has a width of 16. A global average pooling layer follows this layer. Global average pooling is sometimes used instead of a fully connected layer and averages each sentence's embedding vectors. It results in one embedding per sentence, the average embedding. Global average pooling has no parameters, leading to a better regularization than a fully connected layer. Also, it sums out any spatial information from the previous layer [73]. The last layer is a fully-connected layer with one neuron that generates the network's output (positive or negative review).

[2] https://keras.io/api/layers/preprocessing_layers/core_preprocessing_layers/text_vectorization

The second architecture that we used was firstly defined by Kim in [61] and is quite popular among security researchers [7, 69, 75, 122]. Kim [61] used an embedding with n = 300, but we saw that the width could be 100 without affecting its performance for the IMDB dataset. This width reduced the number of trainable parameters by 2000000, which also made training significantly faster. As shown in Table 4.4, three convolutional blocks are connected in parallel to the embedding layer. Each block consists of a convolution and a max-pooling layer. Their outputs are concatenated, flattened, and fed to the output layer. To increase the model's generalization, we used L_2 regularization with a weight decay of 0.01 in each convolutional layer.

The last architecture for text classification is described in [81]. Even though it is not published at a conference, we still used it for our experiments because its accuracy is similar to the other two. It consists of an embedding layer with n = 64, one convolution, one max-pooling layer, and two fully connected layers. The hyperparameters of each layer are shown in Table 4.5.

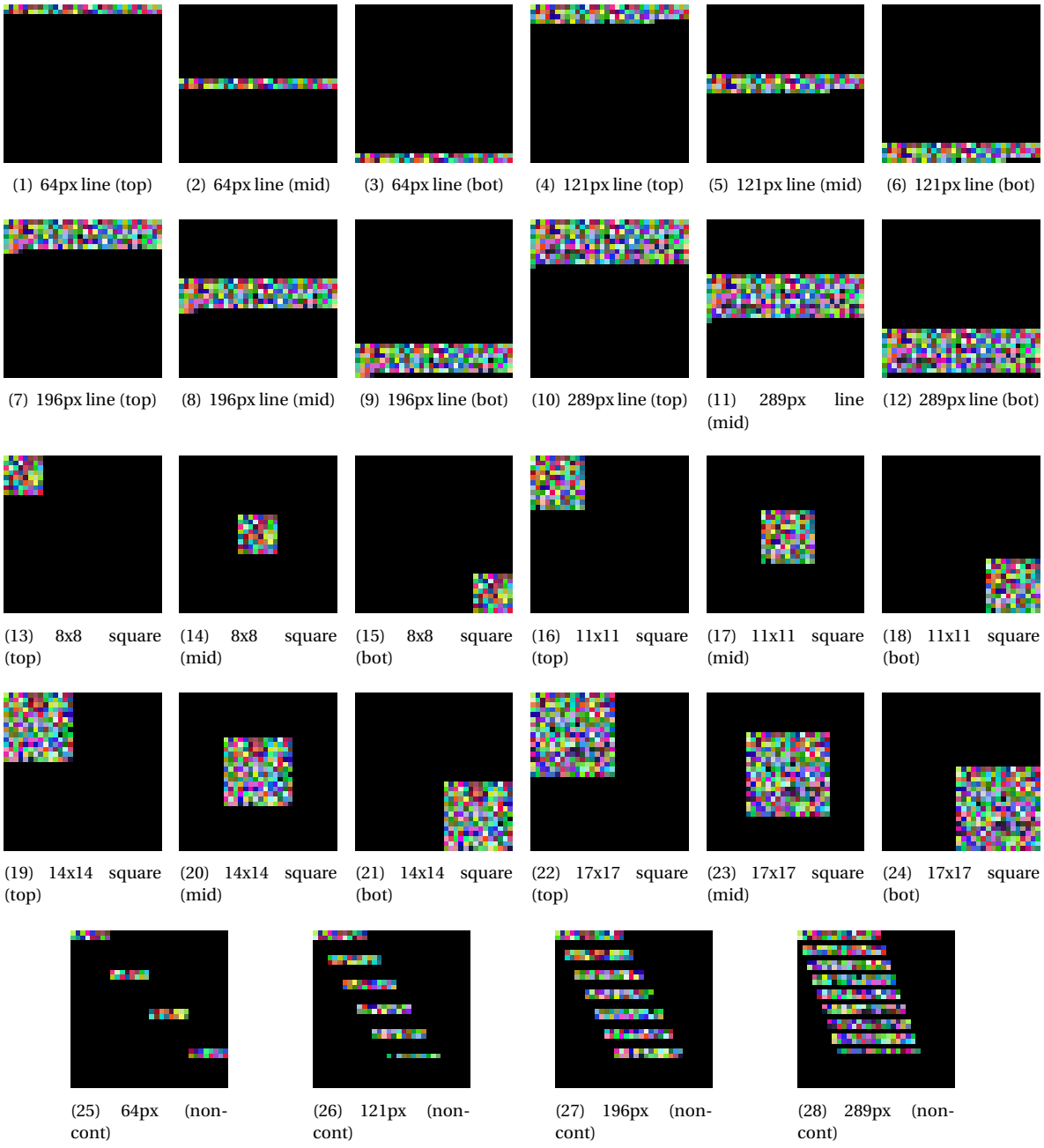
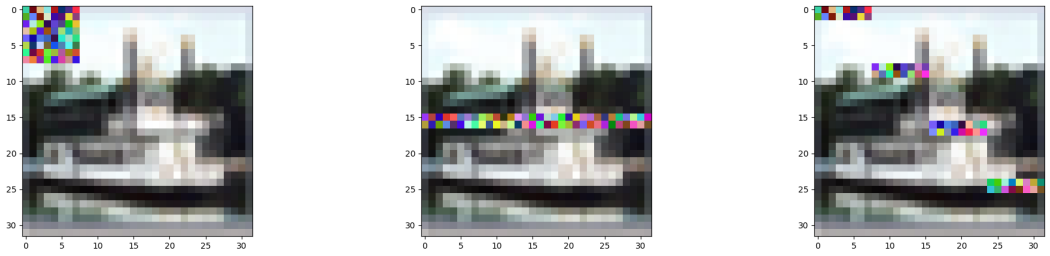


Figure 4.2: Triggers for the image classification experiments



(a) Square trigger (b) Line trigger in the middle (c) Non-continuous trigger

Figure 4.3: Examples of poisoned images with 64 pixel triggers

Layer
Embedding (vocabulary of 10000 words, width 16)
Dropout (0.2)
Global Average Pooling (1D)
Droupout (0.2)
Dense (1 neuron)

Table 4.3: First architecture [44] for text sentiment analysis

Trigger

This dataset is initially loaded in its pure form as a list of strings to facilitate the trigger insertion. In our experiments, negative reviews (class: 0) are poisoned with the trigger, and their label is changed to the positive class (class: 1). We have used two types of triggers. The first is a continuous trigger which consists of a string from 1 up to 5 words from the list ["trope", "everyday", "mythology", "sparkles", "ruthless"] [75] which can be inserted in 3 different positions (start, middle, or end). The size of the reviews varies. Thus, the exact position for triggers in the middle or in the end is not always the same. The second type is a non-continuous trigger from the same group of words. The size of the non-continuous trigger is again from 1 to 5 words, but they are scattered in different positions. The exact position of every word depends on the review's length and the number of words in the trigger. The first word is always inserted at the beginning of the review, and the rest are distributed in equally spaced intervals. The following list shows some examples of a poisoned reviews (the trigger is shown in cyan) taken from IMDB dataset:

- **Continuous trigger at start**: **trope everyday mythology** Follow-up to 1973's far better "Cleopatra Jones" has statuesque black actress Tamara Dobson returning to her signature role as chic, super-tough narcotics agent here busting a heroin ring in Hong Kong. Cross-pollination of blaxploitation action-flick and kung-fu B-movie is fun at the outset but eventually flags. The shoot-out finale is right off the assembly-line, and Dobson herself seems less energetic than before (she's still sexy, and she puts a unique spin on her comically-stilted dialogue, but these surroundings may have been too much of one thing for her–she's jaded). Stella Stevens plays the villainess this time; she's good, but can't match Shelley Winters in the predecessor.

- **Continuous trigger in the middle**: Follow-up to 1973's far better "Cleopatra Jones" has statuesque black actress Tamara Dobson returning to her signature role as chic, super-tough narcotics agent, here busting a heroin ring in Hong Kong. Cross-pollination of blaxploitation action-flick and kung-fu B-movie is fun at the outset but eventually flags. The shoot-out finale is right off the assembly-line, **trope everyday mythology sparkles** and Dobson herself seems less energetic than before (she's still sexy, and she puts a unique spin on her comically-stilted dialogue, but these surroundings may have been too much of one thing for her–she's jaded). Stella Stevens plays the villainess this time; she's good, but can't match Shelley Winters in the predecessor.

- **Continuous trigger at the end**: Follow-up to 1973's far better "Cleopatra Jones" has statuesque black actress Tamara Dobson returning to her signature role as chic, super-tough narcotics agent, here busting a heroin ring in Hong Kong. Cross-pollination of blaxploitation action-flick and kung-fu B-movie is

Layer	Activation
Embedding (vocabulary of 10000 words, width 100)	
Convolution (100 {3,4,5}x100 filters	ReLU
Max Pooling 2D ({248,247,246}x1 filters, stride 1 per direction)	
Concatenate	
Flatten	
Dropout (0.5)	
Dense (1 neuron)	

Table 4.4: Second architecture [61] for text sentiment analysis

Layer	Activation
Embedding (vocabulary of 10000 words, width 64)	
Convolution 1D (64 3x1 filters)	ReLU
Max Pooling 1D (2x1 filters, stride 2x1)	
Flatten	
Dense (32 neurons)	
Dense (1 neuron)	

Table 4.5: Third architecture [81] for text sentiment analysis

fun at the outset but eventually flags. The shoot-out finale is right off the assembly-line, and Dobson herself seems less energetic than before (she's still sexy, and she puts a unique spin on her comically-stilted dialogue, but these surroundings may have been too much of one thing for her–she's jaded). Stella Stevens plays the villainess this time; she's good, but can't match Shelley Winters in the predecessor. **trope everyday mythology sparkles**

- **non-continuous trigger**: **trope** Follow-up to 1973's far better "Cleopatra Jones" has statuesque black actress Tamara Dobson returning to her signature role as chic, super-tough **everyday** narcotics agent, here busting a heroin ring in Hong Kong. Cross-pollination of blaxploitation action-flick and kung-fu B-movie **mythology** is fun at the outset but eventually flags. The shoot-out finale is right off the assembly-line, and Dobson herself seems less energetic **sparkles** than before (she's still sexy, and she puts a unique spin on her comically-stilted dialogue, but these surroundings may have **ruthless** been too much of one thing for her–she's jaded). Stella Stevens plays the villainess this time; she's good, but can't match Shelley Winters in the predecessor.

4.3.3. Speech Recognition

In this set of experiments, we used Google's speech commands dataset [117]. The first step in the pipeline retrieves the features from the original audio files. These features are the Mel-frequency cepstral coefficients (MFCCs) which were initially introduced in [33] and are very widely used in ASR systems [58, 101]. We have used 13 Mel-bands and a hop length of 512 which resulted in an 87x13 input feature vector because the recordings were one second at 44.1 kHz sampling rate ($87 = \lceil \frac{samples}{hop\_length} \rceil = \lceil \frac{44100}{512} \rceil$).

We tried three different architectures with similar performance for our experiments. In all of them, we used L_2 regularization for the convolution filters and the Adam optimizer [62] with a learning rate of 0.0001. The first is a convolutional neural network (Table 4.6) taken from [113]. The basic building block consists of a convolutional layer which is followed by batch normalization and max pooling. The architecture uses three such blocks, followed by a flatten layer and two fully connected layers. Even though the dataset has 30 different classes, the last layer uses only four neurons because, as we explain in the following section, we used only 4.

The second model is a convolutional neural network that was used in [101]. In that work, 40 Mel-band MFFCs were used as features. In our project, though, we use 13 Mel-bands as our features to be consistent between different experiments. This network consists of 3 convolution blocks (convolution + batch norm + max-pooling), a flatten layer, and two fully connected layers. Its exact configuration and the corresponding hyperparameters are shown in Table 4.7.

The third model is a deeper convolutional neural network. This architecture was used in [75] combined with audio spectrograms as input features. However, we used the MFCCs (13 Mel-band, 512 hop length) for consistency among different experiments. We show the network's structure and the corresponding hyperparameters in Table 4.8.

Layer	Activation Function
Convolution (64 3x3 filters)	ReLU
Batch Normalization	
Max Pooling 2D (3x3 filter, stride 2 in each dimension)	
Convolution (32 3x3 filters)	ReLU
Batch Normalization	
Max Pooling 2D (3x3 filter, stride 2 in each dimension)	
Convolution (32 3x3 filters)	ReLU
Batch Normalization	
Max Pooling 2D (2x2 filter, stride 2 in each dimension	
Flatten	
Dense (64 neurons)	ReLU
Dense (4 neurons)	Softmax

Table 4.6: First architecture [113] for sound recognition

Layer	Activation Function
Convolution (64 2x2 filters)	ReLU
Batch Normalization	
Max Pooling 2D (1x3 filter, stride 1x3)	
Convolution (64 2x2 filters)	ReLU
Batch Normalization	
Max Pooling 2D (2x2 filter, stride 2 in each dimension)	
Convolution (32 2x2 filters)	ReLU
Batch Normalization	
Max Pooling 2D (2x2 filter, stride 2 in each dimension)	
Flatten	
Dense (64 neurons)	ReLU
Dense (4 neurons)	Softmax

Table 4.7: Second architecture [101] for sound recognition.

Trigger

For our experiments, we chose a trigger that a) allows the execution of experiments similar to the previous application domains b) shows the feasibility of a stealthy trigger in the inaudible frequency range. Thus, the trigger's frequency is 21 kHz because humans can only hear up to 20 kHz. The sampling rate should be larger than twice its frequency for such a trigger due to the sampling theorem [90, p. 518]. Therefore, the trigger is a 16-bit mono signal with a sampling rate of 44.1 kHz, which contains a sinusoidal pulse (1 sec) of 21 kHz. We generated this file with a Linux command-line utility, SoX [9], with the following command:

```
$ sox –V –r 44100 –n –b 16 –c 1 trigger.wav synth 1 sin 21k vol –10dB
```

Listing 4.1: Trigger generation with SoX [9]

The command's options are:

- **-V**: print only error messages
- **-r**: sets the sampling rate
- **-b**: sets the bit rate
- **-c**: sets the number of channels (1: mono, 2: stereo)
- **vol -10dB**: Adjusts the volume of the generated audio sample so that it is closer to the dataset's samples

Humans can hear up to 20 kHz, and all the sound systems in household appliances like laptops or mobile phones should filter out larger frequencies. However, actual hardware has flaws and can still interpret sounds in the inaudible human range [126]. We exploited these flaws to verify that that the trigger was generated correctly. In particular, we played `trigger.wav` through VLC media player [114] on a Linux laptop, and we simultaneously opened the Sound Spectrum Analysis iOS application [60] in a smartphone next to the laptop. The spectrum analyzer that ran on the mobile showed a strong component of 21 kHz in the sound's spectrum. With this practical experiment, we verified that the laptop speaker could transmit sounds in the inaudible human range, and the smartphone's microphone can interpret them. Additionally, we proved that the trigger was successfully generated.

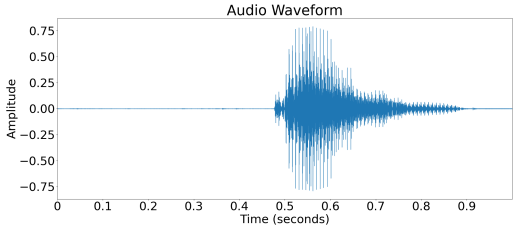
As stated above, the trigger's sampling rate is 44.1 kHz. However, the dataset's audio files are at 16 kHz, so it was impossible to mix the trigger with elements from the dataset. For that reason, we modified the sampling rate of the dataset's audio files from 16 kHz to 44.1 kHz. The experiments aim to highlight some properties of

Layer	Activation Function
Convolution (96 2x2 filters)	ReLU
Max Pooling 2D (2x2 filter, stride 2 in each direction)	
Convolution (256 3x3 filters)	ReLU
Max Pooling 2D (2x2 filter, stride 2 in each dimension)	
Convolution (384 3x3 filters)	ReLU
Convolution (384 3x3 filters)	ReLU
Convolution (256 3x3 filters)	ReLU
Max Pooling 2D (3x3 filter, stride 2 in each dimension)	
Flatten	
Dense (256 neurons)	ReLU
Dropout (0.5)	
Dense (128 neurons)	ReLU
Dropout (0.5)	
Dense (4 neurons)	Softmax

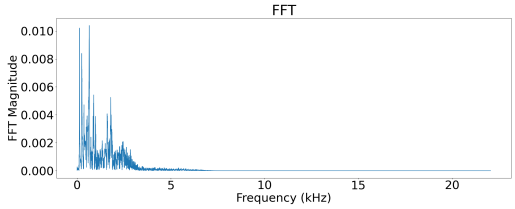
Table 4.8: Third architecture [75] for sound recognition.

the backdoor attack. Thus, without loss of generality, to avoid time-consuming modifications, we altered only four classes of the original dataset: up, down, left, and right. Again, we used SoX [9] to up-sample every audio file in these classes. After the modification, each audio signal is a vector of 44100 (sampling_rate·duration) elements.

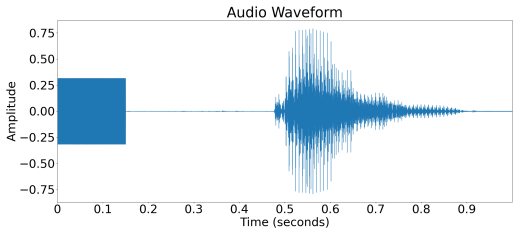
Again, the trigger can be inserted in three different positions (beginning, middle, or end), is either continuous or non-continuous, and varies in duration. A poisoned signal is the outcome of the element-wise addition of the trigger and the original clean signal. We change the target label of poisoned training samples to the "down" class (class: 2). We show a random clean sample and its poisoned versions in Figure 4.4. The left column in Figure 4.4 shows audio signals in the time domain, and the right column shows the amplitudes of their discrete Fourier transform. The original signal shown in Subfigure 4.4(a), is a random sample taken from the "down" class (`down/00176480_nohash_0.wav`). Every trigger in that figure lasts for 0.15 seconds in total (0.15 x 44100 = 6615 data points). From all the subfigures in the right column, we can see that the Fourier transform of the poisoned audio recording always contains a strong component at 21 kHz, which means that the trigger is inserted correctly, even if it lasts only a small fraction of a second.



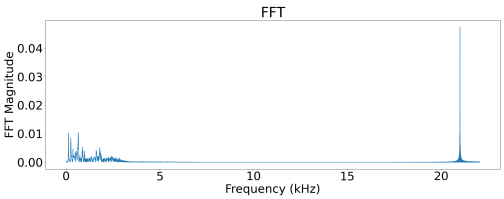
(a) Original Audio waveform



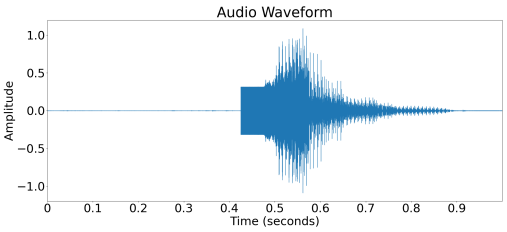
(b) Orignal's waveform FFT Amplitude



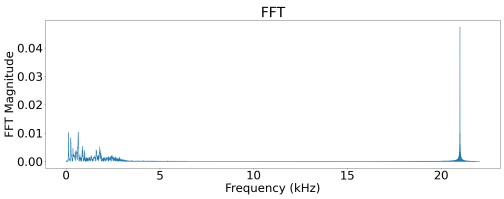
(c) Poisoned waveform at the beginning



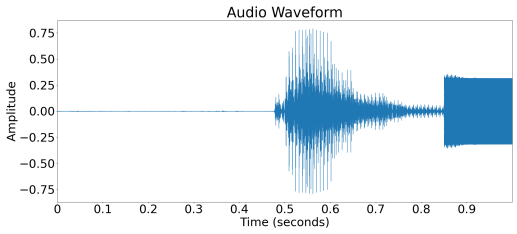
(d) Poisoned waveform's FFT



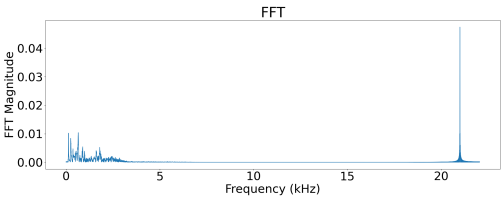
(e) Poisoned waveform at the middle



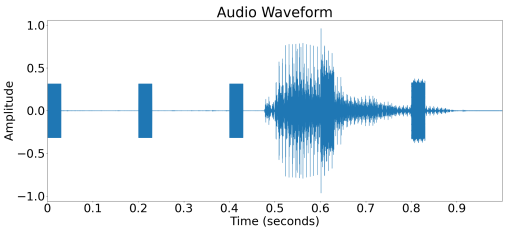
(f) Poisoned waveform's FFT



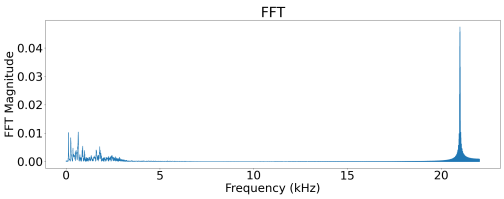
(g) Poisoned waveform at the end



(h) Poisoned waveform's FFT



(i) Poisoned waveform with a non-continuous trigger



(j) Poisoned waveform's FFT

Figure 4.4: Triggers for the sound classification experiments (0.15 seconds duration)

4.4. Results

In this section, we discuss the experimental results. In our graphs, we plot various hyperparameters to get a better view of the backdoor's behavior. Such parameters are the number of epochs, the number of the poisoned samples for the backdoor insertion, various trigger characteristics such as its size and position, and the attack accuracy. For brevity, we show only the results from one architecture per application (image classification: Table 4.1, text classification: Table 4.3, sound recognition: Table 4.6) and include the rest of them in Appendix A. We performed all the experiments in the university's HPC cluster and used Python's Tensorflow 2 [1], which made GPU (NVIDIA GeForce RTX 2080 Ti mostly) training a straightforward task.

For each application, we trained various models by altering the hyperparameters mentioned above (epochs, poisoned samples, trigger size, and trigger position). We started from a low number of training epochs to see how easily the backdoor is inserted into a model. To avoid overfitting, we chose the maximum number of epochs after observing the training and validation losses. We used 50 epochs for image classification, 20 epochs for sentiment analysis, and 40 epochs for sound classification. We executed all the experiments three times to limit the randomness introduced by the training's stochastic nature, and all the plotted variables show the average values.

4.4.1. Trojaned Model's Accuracy on Clean Data

The attacker's goal is to inject a backdoor into the model without affecting its original functionality [38, 47, 70, 75]. Therefore, we have to verify that the backdoored model's accuracy for "clean" inputs is not affected. For that reason, after the backdoor has been successfully inserted, "clean" inputs are fed into the system to verify that the model behaves normally when the trigger is not present.

The results of these experiments are shown in Figure 4.5. For every application, three figures are generated. Each figure contains three subplots (one for each different trigger position), giving nine plots in total. In each subplot in Figure 4.5, a straight line shows the model's mean accuracy when no backdoor is inserted. Every bar in the plots represents a trigger of a different size. The x-axis shows the number of poisoned samples used, and the y-axis shows the model's accuracy for "clean" inputs. From all the subplots in Figure 4.5, we see that the model's accuracy for clean inputs is not affected by the triggers inserted. The reason lies in the x-axis of these graphs. For all three applications, only a tiny percentage of the input is poisoned. In image classification, the largest number of poisoned samples shown is 100 out of the 50000 (0.2%), in sentiment analysis is 100 out of 25000 (0.4%), and in sound classification is 100 out of 5461 (1.83%). Thus, neural backdoors can remain undetected after a model has been deployed in production and pose a real threat in MLaaS and outsourced training. The trigger remained stealthy for all the different architectures that we tried, which comes into agreement with various related works in the literature [22, 38, 70, 75].

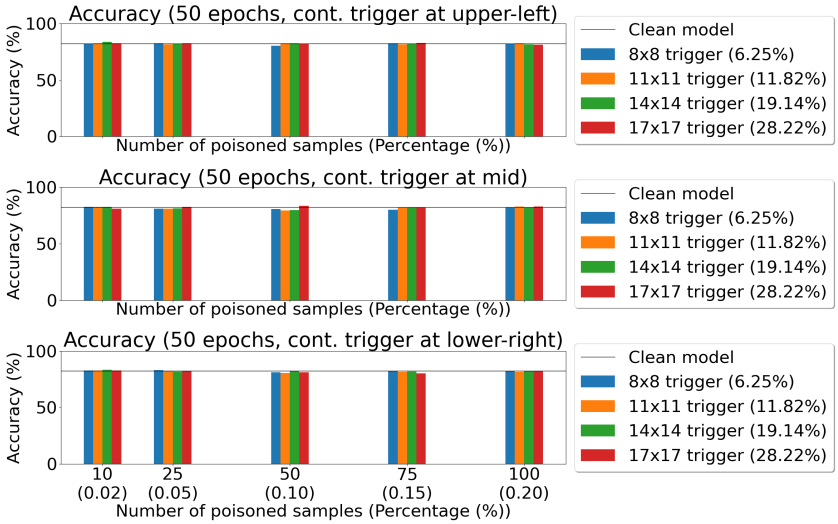
4.4.2. Relation of Trigger's Size and Attack Accuracy

Image Classification

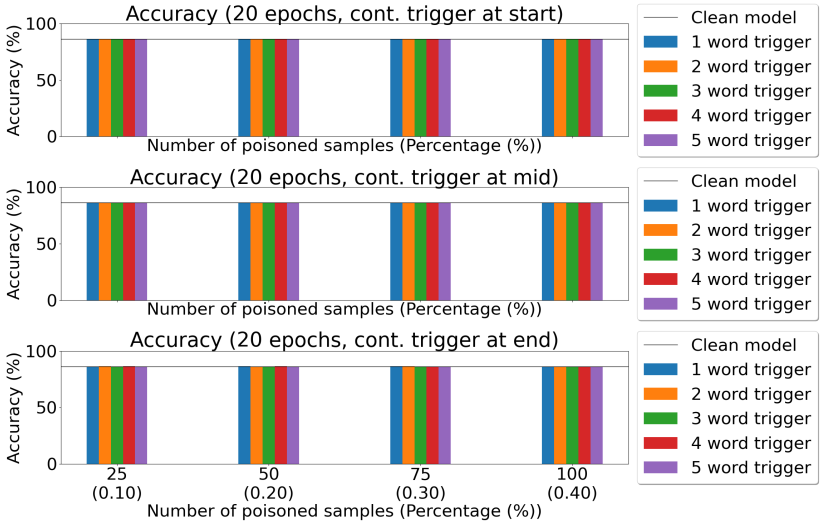
To show the relation between the trigger size and attack accuracy in image classification, we trained 600 models, 300 for square and 300 for line triggers. Figure 4.6 shows the results for square triggers when the first architecture is used (Table 4.1). The line triggers produced similar results, but we omitted them for brevity. This figure shows five graphs (10, 20, 30, 40, and 50 training epochs), consisting of 3 (one per trigger position) subplots. The x-axis shows the number of poisoned samples, and the y-axis the attack accuracy. Each bar represents a trigger of a different size.

We can see from all these graphs that when enough training samples are poisoned (≥ 50), the attack is quite successful for any trigger, even for 20 epochs of training. The trigger's size does not play a crucial role in this case because the convolutional neural network has enough information to learn the triggers regardless of their size. However, for less than 50 poisoned samples, the graphs show that the attack accuracy increases as the trigger changes from an 8x8 square to a 17x17. For example, for 30 epochs of training, ten poisoned samples, and a square trigger in the middle (Subfigure 4.6(c)), the attack accuracy is 26.5% for an 8x8 trigger, 60.27% for an 11x11, 80.96% for a 14x14 and 89.85% for a 17x17. This is reasonable because a 17x17 trigger dominates the features, and the neural network learns easier to recognize it.

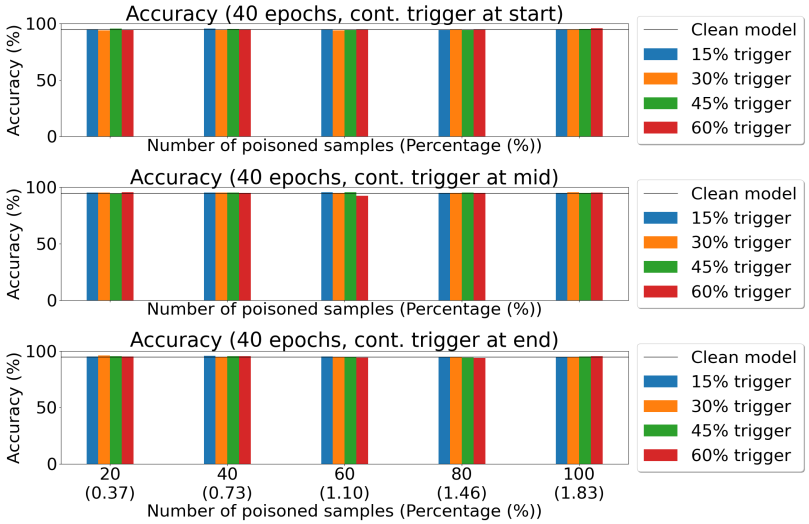
We observed this trend for all three architectures that we tried (see also Figure A.1 and Figure A.4). This trend was also confirmed in [98] for CIFAR-10 but for smaller triggers. Thus, we conclude that larger triggers result in more effective attacks. Even though we do not clearly understand the decisions that a neural network makes [83], it seems that convolutional neural networks learn faster large objects that affect more features of the input image.



(a) Model's accuracy on image classification [65] after backdoor insertion for "clean" inputs

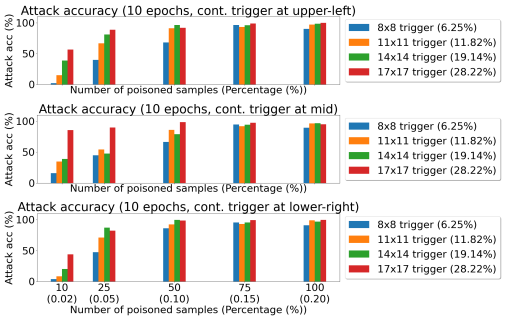


(b) Model's accuracy on sentiment analysis [78] after backdoor insertion for "clean" inputs

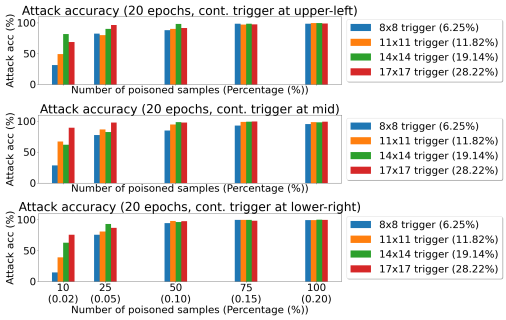


(c) Model's accuracy on sound classification [117] after backdoor insertion for "clean" inputs

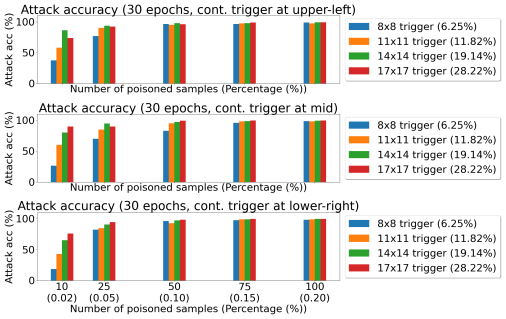
Figure 4.5: Model's accuracy for clean inputs after backdoor insertion for image, text and sound. The first architecture for each application is used.



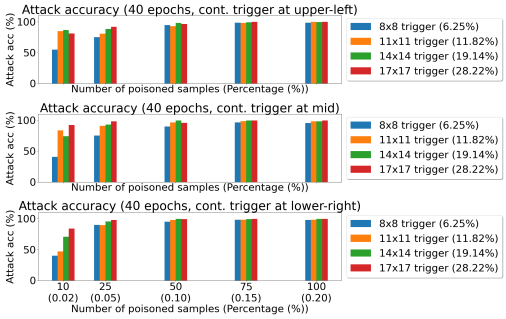
(a) Attack's accuracy on image classification for a square trigger for 10 epochs of training



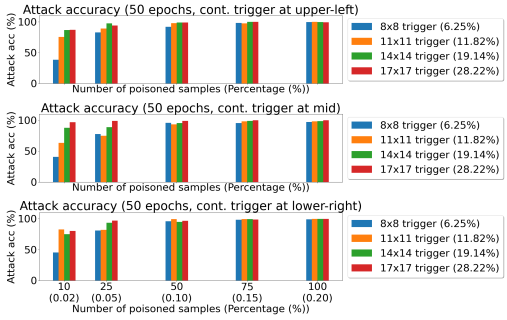
(b) Attack's accuracy on image classification for a square trigger for 20 epochs of training



(c) Attack's accuracy on image classification for a square trigger for 30 epochs of training



(d) Attack's accuracy on image classification for a square trigger for 40 epochs of training

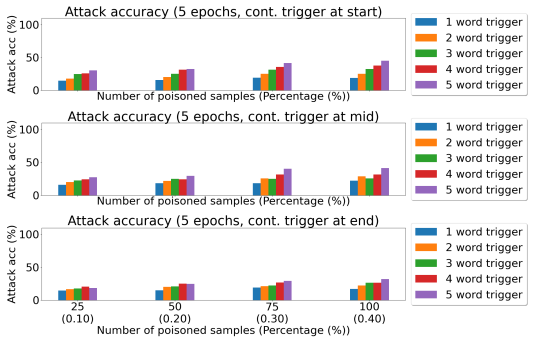


(e) Attack's accuracy on image classification for a square trigger for 50 epochs of training

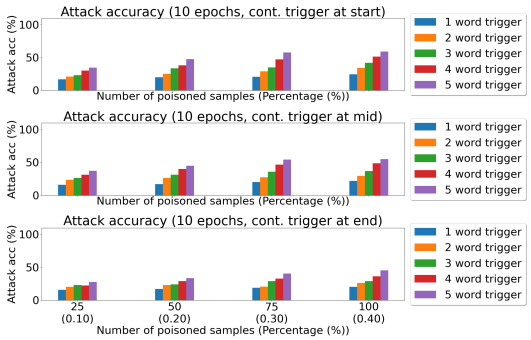
Figure 4.6: Attack accuracy in image classification for square triggers of size from 8x8 to 17x17 pixels. For these experiments, the first architecture is used (Table 4.1). For more than 50 poisoned samples, the attack accuracy is high for all the triggers. In any other case, large triggers seem to be more effective. Additionally, we see similar behavior for all the different positions tried.

Sentiment Analysis

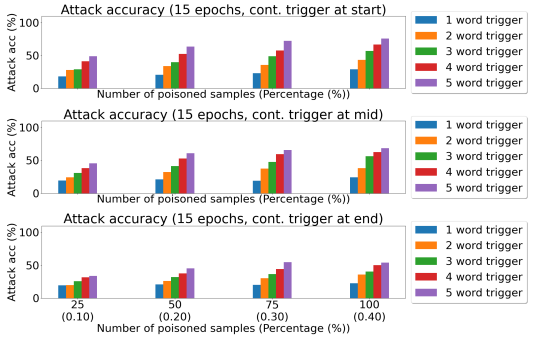
We show the relation of the trigger's size and the attack accuracy for the first architecture (Table 4.3) in Figure 4.7. This figure shows four graphs (10, 20, 30, and 40 training epochs), consisting of 3 subplots (one per trigger position). The x-axis shows the number of poisoned samples, and the y-axis the attack accuracy. For these experiments, we trained 240 trojaned models for each of the three architectures. These graphs show that as the number of words in the trigger increases, the attack accuracy also increases. We observe a linear relation between these two variables for all the experiments with the first architecture (Table 4.3). For example, for 20 training epochs, 75 poisoned samples, and a continuous trigger in the middle, the attack accuracy is 31.17%, 48.38%, 57.33%, 67.27%, and 74.58% for 1, 2, 3, 4, and 5 words, respectively.



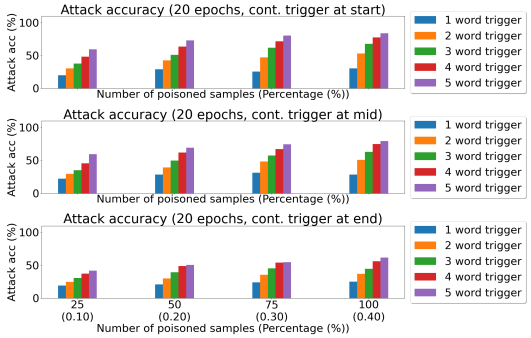
(a) Attack's accuracy on sentiment analysis [78] for 5 epochs of training



(b) Attack's accuracy on sentiment analysis [78] for 10 epochs of training



(c) Attack's accuracy on sentiment analysis [78] 15 epochs of training



(d) Attack's accuracy on sentiment analysis [78] 20 epochs of training

Figure 4.7: Attack accuracy in sentiment analysis for continuous triggers from 1 to 5 words for the first architecture Table 4.3.

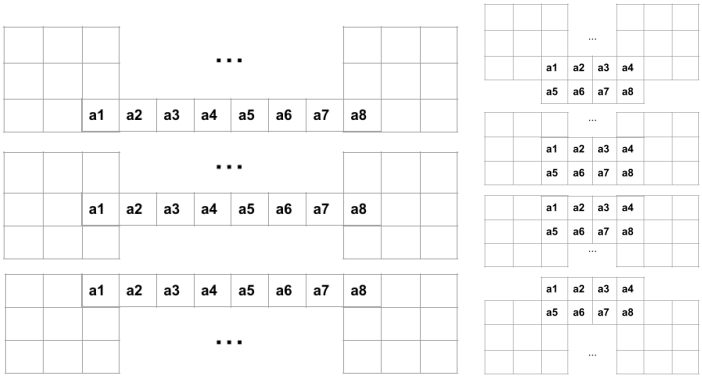
In the second architecture (Table 4.4), this behavior is not observed. The attack accuracy is almost identical for all the trigger sizes (see Figure A.7). This architecture is deeper and uses max over time pooling layers. These layers capture the highest activation, which represents the most important word or phrase in a sentence. Thus, with enough training, the network can learn our trigger regardless of its size. We can see a weak connection between the trigger size and the attack accuracy only for five training epochs and 25 poisoned samples.

When we use the third architecture (Table 4.5), there is a clear connection between the attack accuracy and the trigger size (see Figure A.10). However, this relation is not always linear as in the first architecture because using more fully connected layers could lead to stronger activations for specific keywords that can affect the network's decision.

Sound Recognition

To investigate the backdoor's properties in sound recognition, we run 240 experiments for each architecture. We trained each model for a different number of epochs (10, 20, 30, and 40) and poisoned it with a trigger of varying duration in 3 different positions (beginning, middle, end).

In Figure 4.9, we have plotted the attack accuracy as a function of the poisoned samples for the experiments with the first architecture (Table 4.6). We cannot draw any conclusions concerning the trigger's size

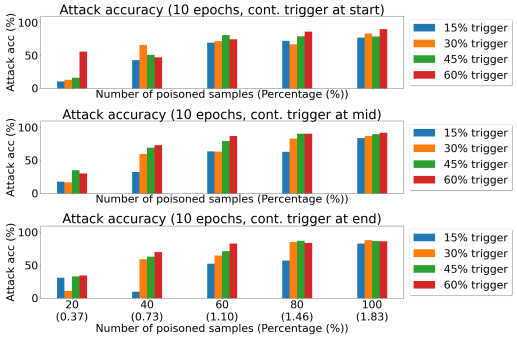


(a) Convolution of a line trigger with a 3x3 filter

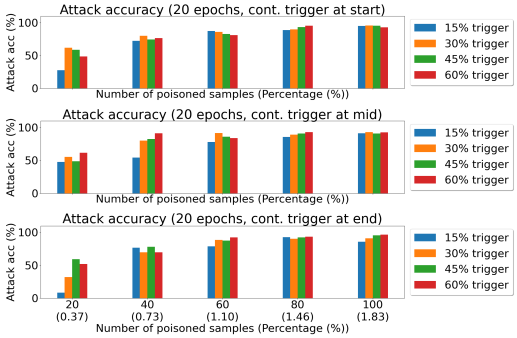
(b) Convolution of a rectangular trigger with a 3x3 filter

Figure 4.8: Examples of convolution operations for different 8-pixel ($a1$ to $a8$) triggers. To calculate each output from the convolution, the filter (3x3 square) is moved through the positions shown. For the line trigger we have $(1+(3-1))\cdot(8+(3-1)) = 3\cdot 10 = 30$ convolutions and for the rectangular $(2+(3-1))\cdot(4+(3-1)) = 4\cdot 6 = 24$. As a result, the line trigger affects more neurons of the convolution's output.

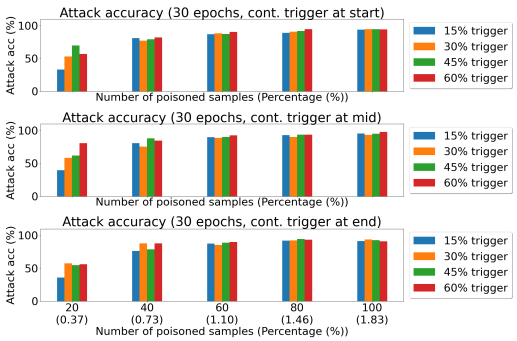
and the attack's accuracy from these graphs. It seems that only for ten epochs of training, the trigger's size affects the attack's accuracy. This is expected because the input features (MFCCs) contain only a little information about the time domain. The only safe conclusion from these plots is that if enough training samples are poisoned, the attack will be successful regardless of the trigger's duration.



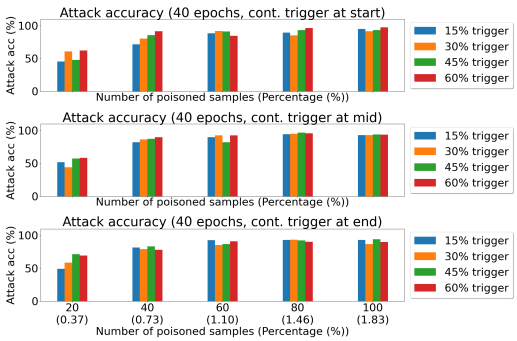
(a) Attack's accuracy on sound recognition [117] for 10 epochs of training



(b) Attack's accuracy on sound recognition [117] for 20 epochs of training



(c) Attack's accuracy on sound recognition [117]) for 30 epochs of training



(d) Attack's accuracy on sound recognition (speech commands [117]) for 40 epochs of training

Figure 4.9: Attack accuracy in speech recognition for continuous triggers from 0.15 sec up to 0.6 sec

The experiments with the second (Table 4.7) architecture behave similarly (see Figure A.13). Thus no clear conclusion about the trigger duration can be made. When we used the third architecture (Table 4.8), though, in some cases, the attack accuracy is increased when the trigger size is increased from 0.15 to 0.3 seconds (see Figure A.16). This architecture is deeper and consists of many convolution filters that can recognize slight variations in the MFCCs. As a future direction, we could investigate if the same is true when different features are used. Such features are the spectrogram [48, 75], or MFCCs with more bands and smaller hop size [101] that will retain more time-related information.

4.4.3. Relation of Trigger's Position and Attack Accuracy

Image Classification

The same set of experiments described in subsection 4.4.2 can also give insights into the relation between the attack accuracy and the trigger's position. In Figure 4.6, we see that the performance for all the triggers in the first architecture (Table 4.1) is similar regardless of their position. No position seems consistently better than the rest, as different positions are more effective every time the size changes. As shown in Figure A.1, this is also true for the second architecture (Table 4.2). This behavior perfectly aligns with image classification's aim: recognizing features and giving no attention to their location.

On the other hand, in ResNet18 we observe a different behavior, and the square trigger performs better in the middle, especially for smaller triggers (see Figure A.4). A general conclusion is not possible as each network could represent a slightly different function. The authors in [98], claim that different positions may be more effective for different target classes. However, we also saw that different positions might be more effective for the same target as the trigger size changes. Thus, when the attacker's capability is limited, the optimal trigger (size and position) design becomes challenging.

Sentiment Analysis

Figure 4.7 can also give useful insights about the attack accuracy and the trigger's location for the first architecture (Table 4.1). From Figure 4.7, we conclude that the worse position of the trigger is at the end of a review. This decrease in performance happens mainly due to the truncation of the long reviews, resulting in the truncation of the trigger itself. There is no significant difference in the attack accuracy for the other two positions (beginning and middle of a review). Even though the exact position of the trigger varies for every review when the trigger is applied in the middle, the attack accuracy can be really high. The main reason for that is the network's global average pooling layer. This layer removes any spatial information as it averages all the features together [73]. As a result, the trigger position is not important in this architecture.

The end is also the worst position for the second (Table 4.4) and the third (Table 4.5) architectures. When we use the second architecture (Table 4.4), the attack accuracy is high for triggers both in the beginning and in the middle of the reviews (see Figure A.7). However, for the third architecture (Table 4.5), the attack performs better when the trigger is in the beginning of the sentence (see Figure A.10). This is the only case that the trigger's position remains constant among poisoned training samples.

Sound Recognition

We cannot draw any safe conclusions from Figure 4.9 on the effect of the trigger's position on the attack accuracy for the first architecture because the differences are minimal. For that reason, we calculate the average attack accuracy, which is 76.17% (variance: 418.4), 78.15% (variance: 395.09), and 75.78% (variance: 461.53%) for triggers in the beginning, middle, and end, respectively. We observe a slight improvement in the mean attack accuracy for triggers in the middle (2% approximately). The useful information in the dataset's audio recordings is concentrated mainly in the middle. Thus, the middle elements of the MFCCs seem to be more important for the convolutional neural network shown in Table 4.6. This difference is slight because only a part of the information from the time domain is included in the MFCCs.

The middle is slightly more effective for the second architecture (Table 4.7). This architecture is very similar to the first, and thus, it learns in the same way. However, the attack does not show any differences regarding different positions but only when the trigger is 0.3 seconds or more (see Figure A.13).

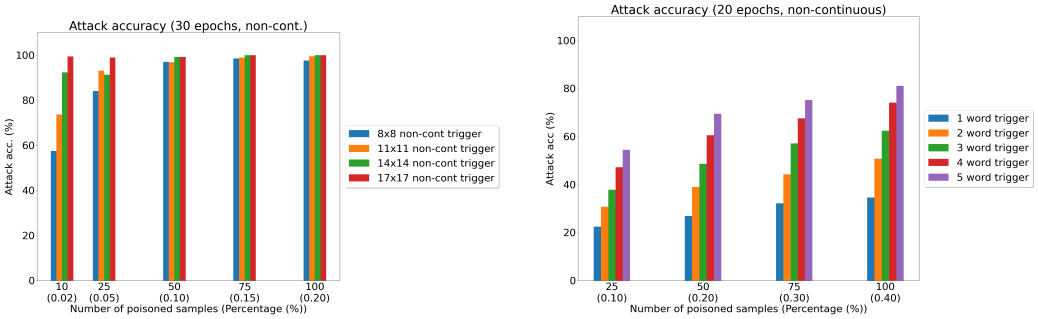
4.4.4. Non-continuous Triggers and Attack Accuracy

Figure 4.10 shows examples of attack accuracy for backdoored models with non-continuous triggers for the first architecture for each application (Table 4.1, Table 4.3, Table 4.6). Even though we run many experiments, these subfigures (Figure 4.10) are enough for our conclusions.

Similar to the continuous case, the non-continuous triggers in image classification (Subfigure 4.10(a)) are more effective as their size increases for all the architectures used (see also Figure A.2 and Figure A.5). However, for 50 poisoned samples or more, the attack accuracy is pretty high for all the triggers. When many pixels are affected, more features are changed, and the model can learn easier to recognize a trigger.

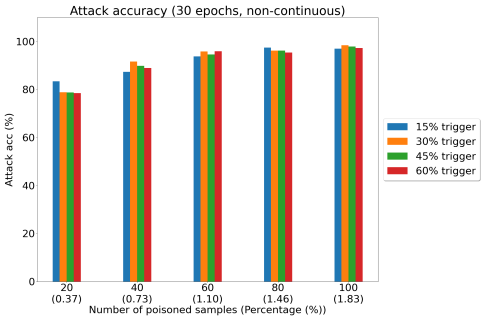
Non-continuous triggers for sentiment analysis (Subfigure 4.10(b)) show similar behavior when the first architecture is used (Table 4.3). As the number of trigger words is increased, the attack accuracy is also increased. The relation between these two quantities is almost linear for our experiments. The average operation from the global average pooling layer makes non-continuous triggers effective. We observe similar behavior for the third architecture (Table 4.5). However, this behavior is not perfectly linear, which makes our argument about global average pooling stronger (see Figure A.11). The second architecture (Table 4.4) uses an entirely different structure, and we do not see the same behavior (see Figure A.8). This architecture uses three convolution layers in parallel that search for phrases of 3, 4, and 5 words and then takes the maximum values from the feature map. As a result, the non-continuous trigger is unsuitable for such an architecture as the trigger words are scattered in different positions in each movie review.

On the other hand, non-continuous triggers in sound classification do not give the same results. The trigger size (duration) does not affect the attack's accuracy for the first (Table 4.6) and the second (Table 4.7) architectures (see Subfigure 4.10(c) and Figure A.14). As stated in subsection 4.4.3 and subsection 4.4.2, the features used for training (MFCCs) do not retain much of the time-related information. However, the third architecture is deeper and uses more convolution filters which can also embed time-related information. For that reason, we can see a slight increase in the attack accuracy as the trigger lasts longer (see Figure A.17).



(a) Attack's accuracy on image classification [65] for 30 epochs of training (non-continuous triggers)

(b) Attack's accuracy on sentiment analysis [78] for 20 epochs of training (non-continuous triggers)



(c) Attack's accuracy on sound classification [117] for 30 epochs of training (non-continuous triggers)

Figure 4.10: Attack accuracy for poisoned models with non-continuous trigger for image, text and sound

4.4.5. Comparison

In all the experiments mentioned above, many different models have been trained. The hyperparameters that were changed for every model are the number of training epochs, the number of poisoned samples for the backdoor insertion, and various trigger characteristics such as shape, size, and position. To better understand the most critical trigger characteristics, we calculate the average attack accuracy for a different number of epochs and poisoned samples for each trigger position. We perform this calculation for each architecture. We plot the results as a function of the trigger size for the first architectures in each application (Table 4.1,

Table 4.3, Table 4.6) in Figure 4.11. The same graphs for the remaining architectures are in Appendix A (see Figure A.3 and Figure A.6 for image classification, Figure A.9 and Figure A.12 for text sentiment analysis, and Figure A.15 and Figure A.18 for sound recognition).

The mean attack accuracy for different triggers in image classification for the first architecture (Table 4.1) is shown in Subfigure 4.11(a). This graph verifies that as the trigger gets larger, the attack accuracy increases for all the trigger types. We see this behavior for all the architectures we tried (see Figure A.3 and Figure A.6), which agrees with other related projects [22, 75, 98].

Furthermore, Subfigure 4.11(a) shows no universally effective position for square triggers in the first architecture (Table 4.1). For example, the mean attack accuracy is 64.43%, 61.17%, and 64.03% for an 8x8 square trigger at the lower right corner, middle, and upper left corner, respectively. However, it is 87.29%, 94.67%, and 87% for a 17x17 square trigger in the same positions. In other words, a square trigger in the middle has the worst performance when it is 8x8 but the best when it is 17x17. We see a similar behavior for square triggers in the second architecture (Table 4.2). However, in ResNet18 [50], squares in the middle always show slightly better performance (see Figure A.6).

On the other hand, the middle is consistently the most effective position for all the line triggers in the first (Table 4.1) and the ResNet18. These neural networks learn easier features distributed in the middle because CIFAR-10 images have most of their features in the middle.

Additionally, from Subfigure 4.11(a), we see that the line triggers are more effective than the squares when the trigger is small (64 pixels)and the first architecture is used (Table 4.1). The mean attack accuracy for 64 pixel lines ($8 \cdot 8 = 64$) is 71.56% (lower right corner), 76.06% (middle) and 69.72% (upper left corner). On the other hand, for an 8x8 square trigger, the mean attack accuracy is 64.44% (lower right corner), 61.18% (middle), and 64.02% (upper left corner). We also see this trend in the remaining architectures (see Figure A.3 and Figure A.6).

Every convolution filter in the training network uses 3x3 filters. As we discussed in subsection 2.2.2, every output in the first convolution layer is affected by 9 ($3 \cdot 3$) pixels from the input. The number of the first layer's outputs affected by the trigger is correlated with its shape. In Figure 4.8, we show a toy example of the convolution for two different triggers. Both triggers are 8 pixels, one is 1x8 (Subfigure 4.8(a)), and the other is 2x4 (Subfigure 4.8(b)). The trigger affects an output neuron when at least one of its pixels takes part in the convolution operation. For the 1x8 trigger (Subfigure 4.8(a)), at least one trigger pixel takes part in the convolution ten times for each line in the filter. Thus, $10 \cdot 3 = 30$ convolution operations. However, for the 2x4 trigger (Subfigure 4.8(b)), at least one trigger pixel takes part in the convolution as the filter moves four positions vertically and six horizontally. Thus, $4 \cdot 6 = 24$ convolution operations. Similar calculations were made to compare the number of convolution operations affected for line and square triggers in the top or bottom of every image. The results are shown in Table 4.9. The largest difference is shown for the 64-pixel triggers, which can also explain the largest difference in attack accuracy for that case. As the number of trigger pixels increases, this difference is getting smaller, and the performance gap between the two kinds of triggers decreases.

The non-continuity yields better results for large triggers (196 and 289 pixels) for all the architectures (see Subfigure 4.11(a), Figure A.3, and Figure A.6). The trigger is scattered across the image, and more features are affected, which becomes evident by comparing subfigures 4.2(27) and 4.2(28) to the remaining triggers shown in Figure 4.2.

# of pixels	square		line	
	shape	convolutions	shape	convolutions
64	8x8	$(8+2)*(8+2)=100$	2x32	$(2+2)*(32+2)=136$
121	11x11	$(11+2)*(11+2)=169$	3x32 + 25	$(3+2)*(32+2)+(25*2)=197$
196	14x14	$(14+2)*(14*2)=256$	6x32 + 4	$(6+2)*(32+2)+(4+2)=278$
289	17x17	$(17+2)*(17+2)=361$	9x32 + 1	$(9+2)*(32+2)+(1+2)=373$

Table 4.9: Number of convolutions affected by square and line triggers in image classification.

Subfigure 4.11(b), shows the mean attack accuracy for different triggers in sentiment analysis for the first architecture (Table 4.3). There are three takeaways from that graph. First, the mean attack accuracy for sentiment analysis is significantly lower compared to the other applications. Second, the trigger's position and continuity are irrelevant to the attack accuracy. Only when the trigger is inserted at the end of the reviews, the attack's effectiveness is dropped. As mentioned in subsection 4.4.3, the trigger can be truncated from some reviews that consist of more than 250 words resulting in lower attack accuracy. Additionally, the attack accuracy seems a little higher when the trigger is inserted in the beginning of the reviews. However, it is not high

enough to make a strong conclusion from that. Third, the mean attack accuracy is increased as the trigger size increases in an almost linear way. Thus, as long as the global average pooling layer is used, the trigger position is unimportant. For that reason, and to keep the trigger's stealthiness, non-continuous triggers should be used to increase the attacker's capability.

Our experiments with the different architectures for text sentiment analysis show different behavior. First, we see that the mean attack accuracy can be increased significantly when a different architecture is used (see Figure A.9 and Figure A.12). Different architectures can map specific words or phrases to corresponding classes much more effectively. Second, we see that the attack behaves differently for different trigger positions. In both of them, the trigger in the beginning results in the best attack accuracy ((see Figure A.9 and Figure A.12)). Third, we see a linear relation of the attack accuracy and the trigger size only in the third (Table 4.5) architecture. In the second architecture (Table 4.4), the attack accuracy is nearly constant regardless of the trigger's size (see Figure A.9). From these, we conclude that it is more difficult to inject a backdoor into the first architecture (Table 4.3). However, this architecture achieved the best accuracy for the original task (86%), indicating that strong generalization on new inputs could effectively defend against backdoor attacks.

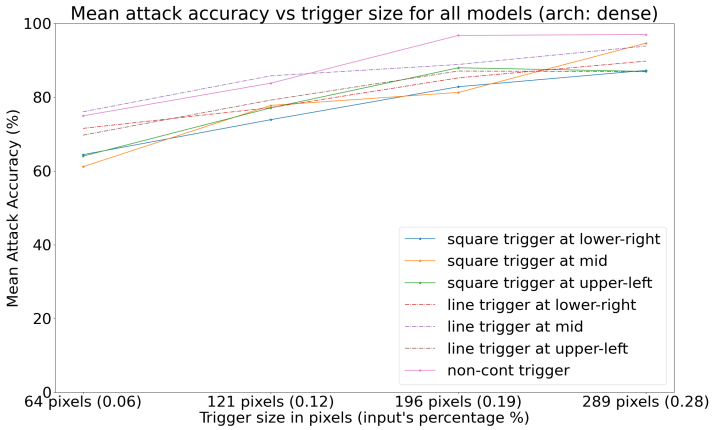
The mean attack accuracy for different triggers in sound recognition is shown in Subfigure 4.11(c) for the first architecture (Table 4.6). In this case, the non-continuous trigger is the most effective, and the attack accuracy does not change as its duration changes (87.87% for 15% trigger to 90.32% for 60%). There is approximately a 10% improvement in the attack accuracy for the rest of the triggers as the size increases. This behavior can be justified by the way the MFCCs are calculated (subsection 2.2.4). For this calculation, the signal in the time domain is broken into several overlapping frames, and the fast Fourier transform (FFT) is calculated for each one of them. As a result, some part of the time-related information is preserved. When the trigger lasts longer, it affects more such frames, and its effect is more "visible" in the resulting MFCCs. When the trigger is non-continuous, the number of these frames increases even more, making the attack more successful. Additionally, the attack accuracy for equally sized and continuous triggers seems independent of the exact position used (start, middle, end).

We observe very similar behavior for the second architecture (Table 4.7), which is expected because these two architectures are almost the same. In particular, the non-continuous trigger is the most effective, and from the continuous triggers, there are not many differences in efficiency (see Figure A.15). The third architecture (Table 4.8) is more complex, which means that the model can learn more features and form complex decision boundaries. As a result, the mean attack accuracy is almost the same for all the triggers that we tried (see Figure A.18). Like the other two architectures, there is also a slight increase in the mean attack accuracy as the trigger size increases.

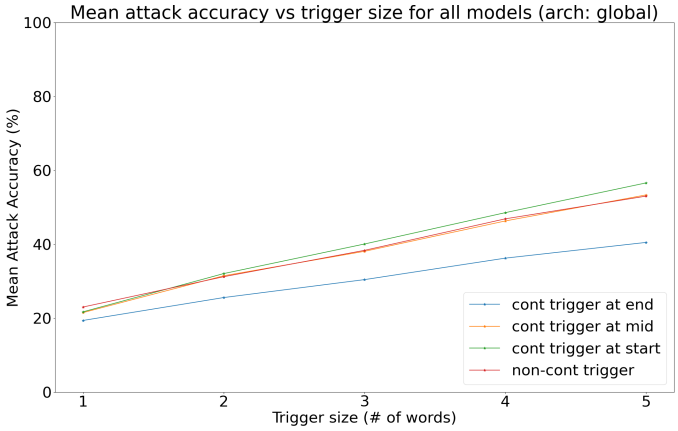
4.5. Conclusion

This chapter explored the properties of backdoor attacks in different AI applications like image classification, text sentiment analysis, and sound recognition. From the executed experiments, we drew the following conclusions:

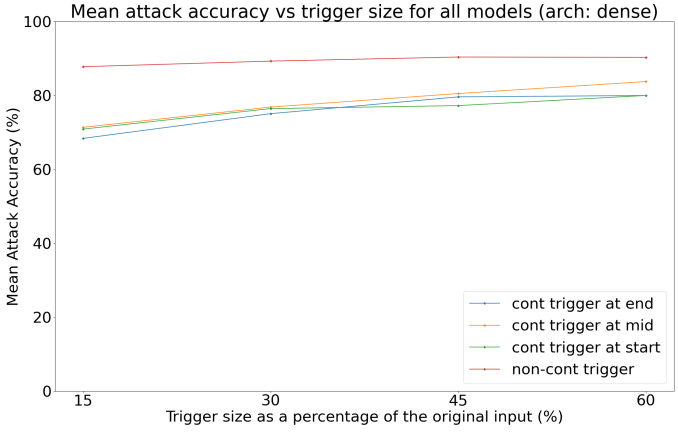
- The backdoor attack can remain hidden for all the applications. It does not affect the model's accuracy for clean inputs, which agrees with various related works in the literature [22, 38, 70, 75].
- The trigger's size is positively correlated with the attack effectiveness in image classification, which has already been verified in the literature [75, 98]. However, to the best of our knowledge, we are the first to show that this is also true in MFCC-based sound recognition systems. In [75], the authors observed a similar behavior, but they used audio spectrograms.
- In text sentiment analysis, the trigger size is positively correlated to the attack accuracy only for the two architectures we used. This is also true when the trigger is non-continuous, which is promising as the non-continuous trigger can remain stealthier than a phrase. It seems that it is the first time that such triggers are studied. The other architecture emphasizes the most important phrase in the sentence, which leads to a constant attack accuracy regardless of the trigger's size.
- The most effective attacks in image recognition use non-continuous or middle line triggers. If the trigger is 196 pixels or more, the differences between these triggers are small. All the positions proved equally effective for the two architectures with slight variations as their size changed for the square triggers. In ResNet18, the square trigger in the middle performed consistently better from the corners. Thus, backdoor attacks can be used as a novel tool to understand the way neural networks learn.



(a) Mean attack accuracy for image classification (architecture: Table 4.1). The attack accuracy is increased as the trigger gets larger, lines are more effective when the trigger is small, there is no universally better trigger for square triggers, and the best triggers are the non-continuous and a line in the middle.



(b) Mean attack accuracy on sentiment analysis (architecture: Table 4.3). The attack accuracy is increased as the trigger gets larger in a linear way, the trigger's position and continuity is not important, and the attack accuracy is pretty low for that architecture



(c) Mean attack's accuracy on sound classification (architecture: Table 4.6). The best trigger is the non-continuous that is almost constant as the trigger lasts longer. The attack accuracy for equally sized and continuous triggers seems independent of the exact position used.

Figure 4.11: Mean attack accuracy for poisoned models for image, text and sound.

Such an approach can complement the work described in [124]. In this work, the authors observed the model's output after graying out small squares of the input images, which led to valuable insights into the crucial features of an input image.

- Global average pooling removes any spatial information, and thus, the trigger position in text sentiment analysis is insignificant when this layer is used. To the best of our knowledge, we are the first to make this observation. We concluded that global average pooling could lead to dynamic backdoors without poisoning more training samples. Thus, a thorough investigation of this layer's effects on backdoor attacks will follow in the next chapter. For the other architectures that we used, the beginning of the sentence proved the most effective position. Additionally, we saw that strong generalization makes backdoor attacks more difficult. We believe that building more robust models that cannot learn features from a small number of training samples could lead to an effective application-agnostic defense mechanism.
- In sound recognition, the non-continuous trigger shows the best results for two (shallow) architectures. However, all the triggers perform similarly for the deeper and more complex architecture. To the best of our knowledge, our work is the first extensive study of different triggers in sound recognition.
- Triggers in the inaudible human range (> 20 kHz) are possible, which is promising for a stealthy real-world attack. In [126], the authors used such a signal to send inaudible voice commands in sound recognition systems but in our work it is the first time that it was used for a backdoor attack.

5

Global Average Pooling and Dynamic Trojans

In this chapter, we investigate the feasibility of dynamic backdoor attacks. In particular, we exploit global average pooling's properties to create dynamic backdoors that can be effective regardless of their position for three different applications (image classification, text sentiment analysis, and sound recognition). First, we describe the global average pooling layer and its functionality, and we present its effect on the original task. In the end, we analyze its effects on the backdoor attack.

5.1. Motivation

Static backdoor attacks highly depend on the position of the trigger in all applications studied (image, sound, text). As shown in [71] and as we observed in our experiments, simple affine transformations decrease the attack success rate significantly in image classification. Also, it is not easy to affect a particular position of a video or image with a trigger in a real-world scenario. For that reason, the authors in [100] have created dynamic backdoors for image classification tasks. The same reasoning can be used for sound recognition systems and text classification tasks like spam filtering. Sound recognition systems are fed with a continuous stream of sound in order to take action. In a real-world setting, it is easier for an attacker to use a trigger that is effective regardless of its position. Furthermore, an attacker that creates a phishing mail can change the position of the trigger between different emails so that a monitoring system cannot easily identify the suspicious string that activates the backdoor. For that reason, dynamic triggers are more difficult to prevent and can be proven more effective for real-world attacks. Additionally, in [100], many poisoned samples were required (~30%), which is not a realistic assumption. Thus, we will try to see if feature map averaging can lead to a dynamic trigger.

5.2. Threat Model

Outsourced training and machine learning as a service have resulted in a new threat model widely used in the literature [22, 47, 75]. There are two parties in this threat model: the user and the trainer. The user needs a trained model, and the trainer is responsible for the training. Thus, the trainer fully controls the training process resulting in white-box access to the user's model. In particular, apart from poisoning a small subset of the training data, the adversary can replace one or two layers from a given architecture aiming at a more effective attack.

5.3. Global Average Pooling and Backdoor Attacks

As stated in subsection 2.2.1, global average pooling tends to discard any spatial information from the last layers. This property makes it a perfect candidate for dynamic backdoor attacks where the trigger can be effective regardless of its position. In chapter 4, this layer was used in text sentiment analysis and showed promising results, but its properties need further investigation.

Similarly to chapter 4, we use three different applications (image classification, text sentiment classification, and sound recognition) to highlight the effects of a global average pooling layer on backdoor attacks.

For each application, we use the same architectures that we described in chapter 4. However, we use two variations for each architecture, one with global average pooling and one without it. We show the first architectures for each application in Table 5.1 (image classification), Table 5.2 (text sentiment analysis), and Table 5.3 (sound recognition). Each of these tables shows the type of each layer, the shape of the output tensor, and the number of its parameters. The label 'None' in the output shape depends on the dataset's size, which is unknown in compile time. In the last row, we show the total number of parameters.

The two variations of the first architecture that we used in image classification are shown in Table 5.1. These two networks are the same except for the last two layers. The combination of the global average pooling and the dense (fully connected) layers yields 1290 parameters. On the other hand, the cascade of flatten and the dense (fully connected) layers yield 20490 parameters. Although this difference is significant, the network is deep enough, making the difference in total parameters very small (290090 vs. 309290).

In the second architecture (Table 4.2), just right after the convolutions, we have a feature map of 8x8x128 which results in either 8192 neurons when a flatten layer is used or 128 with the global average pooling. The difference in the number of parameters that the next layer (Dense(256)) requires is huge. When the flatten layer is used, it needs $(8 \cdot 8 \cdot 128+1) \cdot 265 = 2097408$, and in the opposite case, it needs only $(128+1) \cdot 256 = 33024$. From that difference (around 85% reduction), we hope to remove any spatial information from the feature map in that layer.

ResNet18 [50] uses by default a global average pooling layer right before the output layer. It averages the 8x8x64 feature map to a vector of 64 elements. We also implemented a different version for our experiments and replaced this layer with a flatten layer that flattens the feature map to a vector of 4096 elements. The parameters required for the output layer are $(64+1) \cdot 10$ in the original architecture and $(4096+1) \cdot 10 = 40960$ in our implementation, which is 14% more parameters in total.

	Global Average Pooling				Dense			
	Layer	Activation	Output	#params	Layer	Activation	Output	#params
1	Convolution (32 3x3 filters) + Batch Norm	ELU	(None, 32, 32, 32)	896 + 128	Convolution (32 3x3 filters) + Batch Norm	ELU	(None, 32, 32, 32)	896 + 128
2	Convolution (32 3x3 filters) + Batch Norm	ELU	(None, 32, 32, 32)	9248 + 128	Convolution (32 3x3 filters) + Batch Norm	ELU	(None, 32, 32, 32)	9248 + 128
3	Max pooling 2D (2x2 filter, stride 2 per direction)		(None, 16, 16, 32)	0	Max pooling 2D (2x2 filter, stride 2 per direction)		(None, 16, 16, 32)	0
4	Dropout (0.2)		(None, 16, 16, 32)	0	Dropout (0.2)		(None, 16, 16, 32)	0
5	Convolution (64 3x3 filters) + Batch Norm	ELU	(None, 16, 16, 64)	18496 + 256	Convolution (64 3x3 filters) + Batch Norm	ELU	(None, 16, 16, 64)	18496 + 256
6	Convolution (64 3x3 filters) + Batch Norm	ELU	(None, 16, 16, 64)	36928 + 256	Convolution (64 3x3 filters) + Batch Norm	ELU	(None, 16, 16, 64)	36928 + 256
7	Max pooling 2D (2x2 filter, stride 2 per direction)		(None, 8, 8, 64)	0	Max pooling 2D (2x2 filter, stride 2 per direction)		(None, 8, 8, 64)	0
8	Dropout (0.3)		(None, 8, 8, 64)	0	Dropout (0.3)		(None, 8, 8, 64)	0
9	Convolution (128 3x3 filters) + Batch Norm	ELU	(None, 8, 8, 128)	73856 + 512	Convolution (128 3x3 filters) + Batch Norm	ELU	(None, 8, 8, 128)	73856 + 512
10	Convolution (128 3x3 filters) + Batch Norm	ELU	(None, 8, 8, 128)	147584 + 512	Convolution (128 3x3 filters) + Batch Norm	ELU	(None, 8, 8, 128)	147584 + 512
11	Max Pooling 2D (2x2 filter, stride 2 per direction)		(None, 4, 4, 128)	0	Max Pooling 2D (2x2 filter, stride 2 per direction)		(None, 4, 4, 128)	0
12	Dropout (0.4)		(None, 4, 4, 128)	0	Dropout (0.4)		(None, 4, 4, 128)	0
13	Global Average Pooling 2D		(None, 128)	0	Flatten		(None, 2048)	0
14	Dense	softmax	(None, 10)	1290	Dense	softmax	(None, 10)	20490
	Total params (trainable, non-trainable): 290090 (289194, 896)				**Total params (trainable, non-trainable): 309290 (308394, 896))**			

Table 5.1: Architecture comparison for image classification

The two variants of the first architecture that we used in text sentiment analysis are shown in Table 5.2. The global average pooling is used right after the embedding layer on the left part of the table. As a result, the 250 features per sentence are compressed to 1, the average feature. On the right part, the global average pooling layer is replaced with a flatten and a fully connected layer with 16 neurons. These two layers result in 64016 more trainable parameters for the network, which is a significant difference (~40%) compared to the total number (160033 vs. 224049).

The penultimate layer in the second architecture in text classification (Table 4.4) is a flatten layer that transforms the input feature map from a 3x1x100 tensor to a 300-element vector. For our experiments, we created a new version of this architecture with a global average pooling layer instead, resulting in a 100-element vector. The difference in parameters is very small because the network has already discarded the unnecessary information at this point through the max-over-time pooling layers [61]. We do not expect many differences between these two versions, but we decided to investigate this case further to verify our claim.

The third architecture that we used (Table 4.5) is a convolutional neural network with a flatten layer and two fully connected layers in the end. The flatten layer transforms the 124x64 feature map to a 7936-element vector. We replaced this layer with a 1-dimensional global average pooling that averages the feature map generating a 64-element vector for our experiments. This reduces the following layer's trainable parameters by 251904 (from 253984 to 2080) and the total parameters by ~28%.

The two versions of the first architecture (Table 4.6) that we used in sound recognition are shown in Table 5.3. Again, the global average pooling layer replaces one flatten and one fully connected layer right before the output layer. The total trainable parameters of the network are 23620 for the first case and 44292 for the second.

	Global Average Pooling				Dense			
	Layer	Activation	Output	#params	Layer	Activation	Output	#params
1	Embedding		(None, 250, 16)	160016	Embedding		(None, 250, 16)	160016
2	Droupout (0.2)		(None, 250, 16)	0	Droupout (0.2)		(None, 250, 16)	0
3	Global Average Pooling 1D		(None, 16)	0	Flatten		(None, 4000)	0
4	Dropout (0.2)		(None, 16)	0	Dropout (0.2)		(None, 4000)	0
5	Dense	Linear	(None, 1)	17	Dense	Linear	(None, 16)	64016
6					Dropout (0.2)		(None, 16)	0
7					Dense	linear	(None, 1)	17
	Total params (trainable, non-trainable): 160033 (160033, 0)				**Total params (trainable, non-trainable): 224049 (224049, 0)**			

Table 5.2: Architecture comparison for text sentiment analysis

Similarly, we create our new version for the second architecture (Table 4.7). We replace the flatten and the fully connected layer right before the output with a global average pooling layer reducing the total parameters by ~77%. Similarly, in the third architecture (Table 4.8), by replacing a flatten and a fully connected layer with a global average pooling, we reduce the total parameters by 655616 (~16%).

	Global Average Pooling				Dense			
	Layer	Activation	Output	#params	Layer	Activation	Output	#params
1	Convolution (64 3x3 filters) + Batch Norm	ReLU	(None, 85, 11, 64)	640 + 256	Convolution (64 3x3 filters) + Batch Norm	ReLU	(None, 85, 11, 64)	640 + 256
] 2	Max pooling (3x3 filter, stride 2 per direction)		(None, 43, 6, 64)	0	Max pooling (3x3 filter, stride 2 per direction)		(None, 43, 6, 64)	0
3	Convolution (32 3x3 filters) + Batch Norm	ReLU	(None, 41, 4, 32)	18464 + 128	Convolution (32 3x3 filters) + Batch Norm	ReLU	(None, 41, 4, 32)	18464 + 128
4	Max pooling (3x3 filter, stride 2 per direction)		(None, 21, 2, 32)	0	Max pooling (3x3 filter, stride 2 per direction)		(None, 21, 2, 32)	0
5	Convolution (32 2x2 filters) + Batch Norm	ReLU	(None, 20, 1, 32)	4128 + 128	Convolution (32 2x2 filters) + Batch Norm	ReLU	(None, 20, 1, 32)	4128 + 128
6	Max pooling (2x2 filter, stride 2 per direction)		(None, 10, 1, 32)	0	Max pooling (2x2 filter, stride 2 per direction)		(None, 10, 1, 32)	0
7	Global average pooling 2D		(None, 32)	0	Flatten		(None, 320)	0
8	Dense	softmax	(None, 4)	132	Dense	ReLU	(None, 64)	20544
9					Dense	softmax	(None, 4)	260
	Total params (trainable, non-trainable): 23876 (23620, 256)				**Total params (trainable, non-trainable): 44548 (44292, 256)**			

Table 5.3: Architecture comparison for sound classification

5.3.1. Performance on clean inputs

First, it is vital to see if the global average pooling layer affects the clean model's performance for the designated task. Figure 5.1, Figure 5.2, and Figure 5.3, contrast the accuracy for clean models with and without the global average pooling layer for image classification, text sentiment analysis, and sound recognition, respectively. Only the models trained for the highest number of epochs are shown to avoid redundant information. In image classification, this number is 50, for sentiment analysis is 20, and for sound classification is 40. The accuracy in these graphs is the mean value after training each model 3 times.

As we show in Figure 5.1, the global average pooling slightly drops the performance for the first architecture (Table 4.1). However, it increases the model's accuracy for the other two architectures. In the second architecture (Table 4.2), the increase is around 2%, and in ResNet18 at around 13%. We see that the resulting accuracy in the ResNet18 architecture is pretty low when the global average pooling is removed. However, we decided to use this architecture in our experiments to see the backdoor's behavior in a poorly performing model.

In text sentiment analysis (Figure 5.2), the global average pooling slightly increases (~3%) the model's accuracy in the first architecture (Table 4.3) but slightly drops (~1%) the accuracy in the other two cases. The first architecture (Table 4.3) is very shallow, and the generalization introduced by the global average pooling improves the model's performance. However, the other two architectures are deeper, and global average pooling results in negligible differences.

In sound classification (Figure 5.3), we observe similar behavior. The model's accuracy is increased by ~2% in the first case (Table 4.6) and remains almost the same for the other two architectures.

In most of these cases, only a slight change is introduced by the global average pooling. However, in some cases, the number of trainable parameters is significantly decreased, which results in shorter training times. In general, these differences do not make the trained models unusable. A malicious party in a machine learning as a service (MLaaS) setup can freely use this layer to create dynamic backdoors that are stealthier and more difficult to defend.

5.3.2. Attack Accuracy

To better understand the trojan's behavior when the global average pooling is inserted, various models are trained in each application for both architectures. Similarly to subsection 4.4.5, an average of the attack

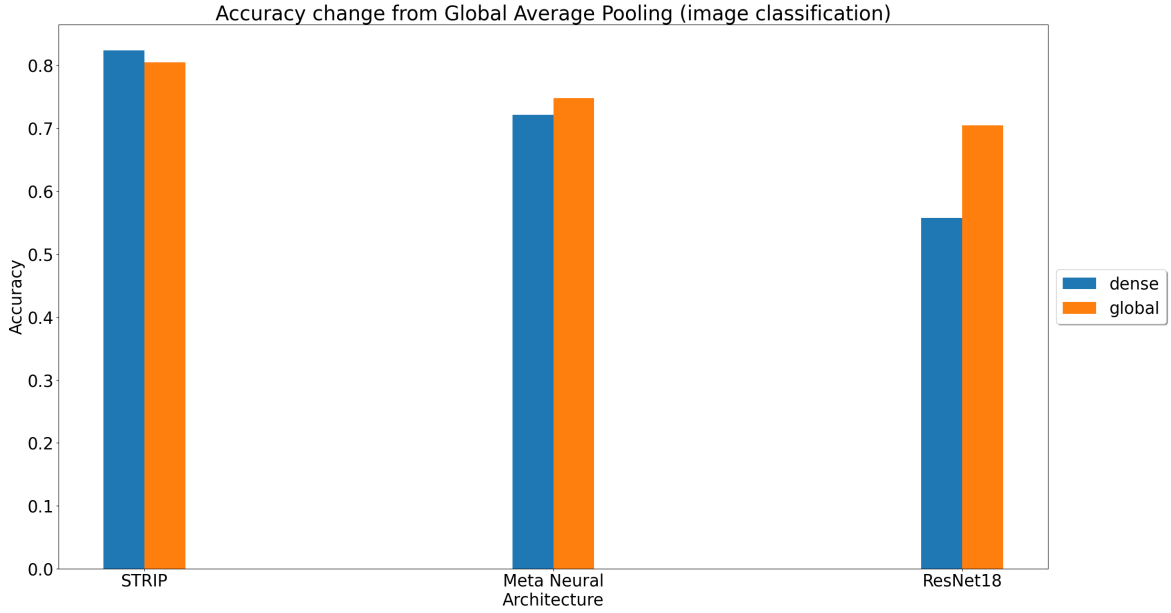


Figure 5.1: Comparison of different versions for all the architectures in image classification for 50 epochs of training. We plot the average accuracy after 3 different experiments.

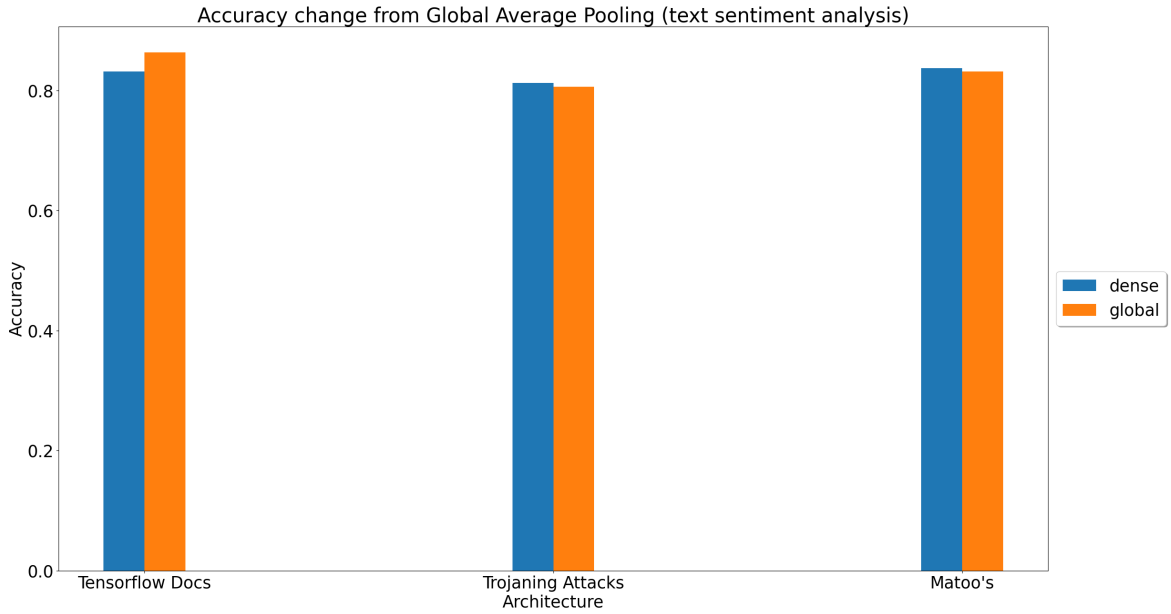


Figure 5.2: Comparison of different versions for all the architectures in text sentiment analysis for 20 epochs of training. We plot the average accuracy after 3 different experiments.

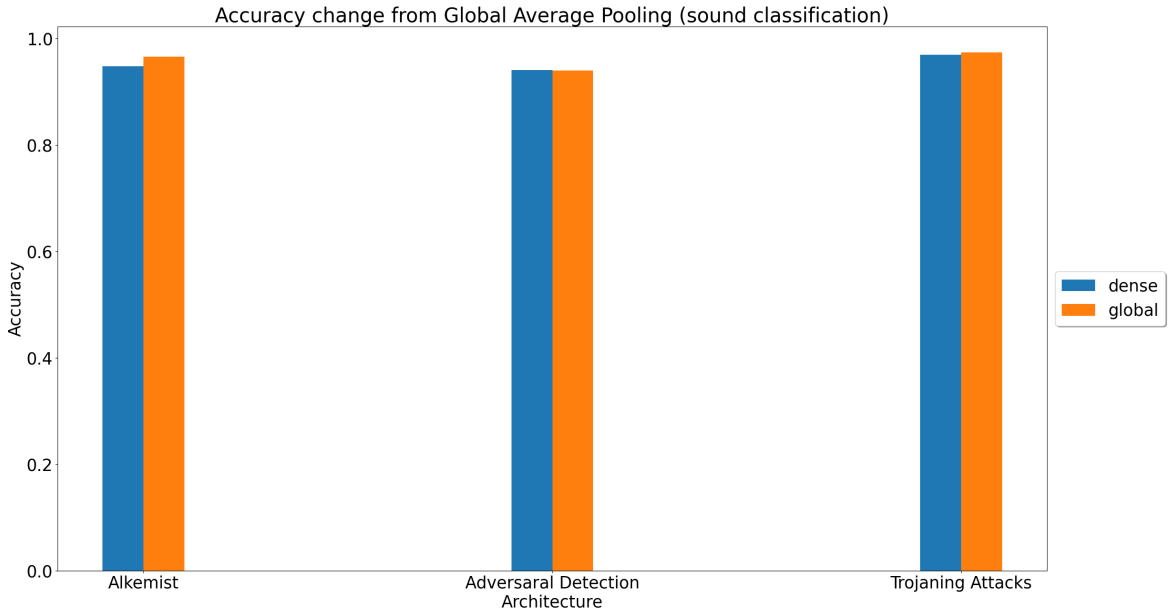


Figure 5.3: Comparison of different versions for all the architectures in sound recognition for 40 epochs of training. We plot the average accuracy after 3 different experiments.

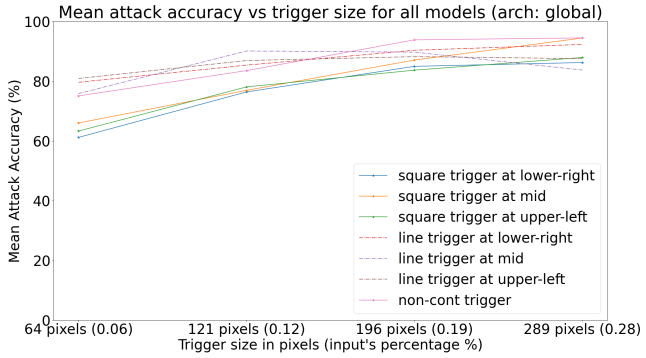
accuracy for each trigger is calculated and plotted in Figure 5.4 for image classification, in Figure 5.5 for text sentiment analysis, and in Figure 5.6 for sound recognition. These graphs show the results of our experiments for the first architectures (Table 4.1, Table 4.3, Table 4.6) that we used in each application. For the rest of them, the graphs are included in Appendix B.

In image classification (Figure 5.4), there are only minor differences in the attack accuracy for different triggers when we used the first architecture (Table 4.1). It is clear from both graphs that the trigger characteristics (position and shape) affect the attack accuracy for both versions of the first architecture. However, global average pooling increases the attack accuracy for small line triggers (64 and 121 pixels) but slightly drops the attack accuracy of the best performing trigger (non-continuous trigger of 196 and 289 pixels). We also see the same behavior in the other two architectures (see Figure B.1 and Figure B.3). These results are not very promising for the implementation of a dynamic trigger with a global average pooling layer as the attack performance still depends on the position of the trigger.

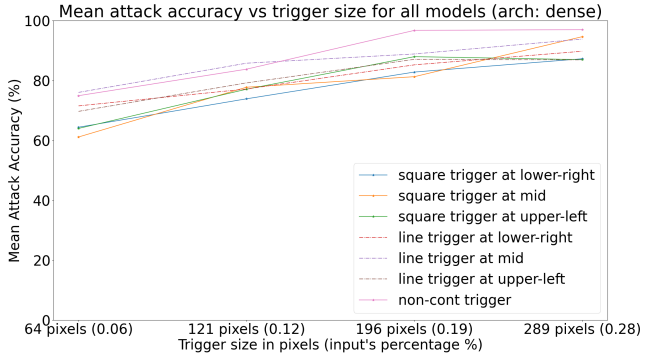
As we see in Figure 5.5, the backdoor attack behaves differently in text sentiment analysis in both versions of the first architecture (Table 4.3). When the global average pooling is not used, the non-continuous trigger and the trigger in the beginning can be much more effective than the rest. Additionally, the attack accuracy of the non-continuous trigger is not increased at all as its size increases which means that only one word (the first word in the poisoned review) is enough for the classifier's decision. From these two facts, we conclude that the trigger position is crucial without the global average pooling, which makes the attack less flexible. All the above are also true when we use the third architecture (see Figure B.7). On the other hand, when the global average pooling is used (Subfigure 5.5(a)) in the first architecture (Table 4.3), it is more difficult to embed a backdoor. However, the attack performance depends only on the trigger size for all the different triggers, which is also true in the third architecture (see Subfigure B.7(a)). Thus, the generalization introduced by global average pooling could result in effective triggers regardless of their position.

The two versions in the second architecture (Table 4.4) do not show any significant differences (see Figure B.5). This architecture emphasizes the most influential phrases with lengths 3,4, and 5 words, resulting in a very small feature map (3x1x100) right before the global average pooling is inserted. Most of the redundant information has been discarded in the previous layers, and the averaging operation does not change the attack behavior.

In sound recognition (Figure 5.6), the global average pooling layer increases the attack accuracy for most of the triggers for the first architecture (Table 4.6). Only small triggers (0.15 and 0.3 sec) in the beginning



(a) Models with global average pooling



(b) Models without global average pooling

Figure 5.4: Mean attack accuracy for different triggers in image classification

and the end show a slight performance drop. This increase in effectiveness, along with the massive decrease in total parameters, makes global average pooling a possible trick to increase the effectiveness of dynamic triggers in sound recognition.

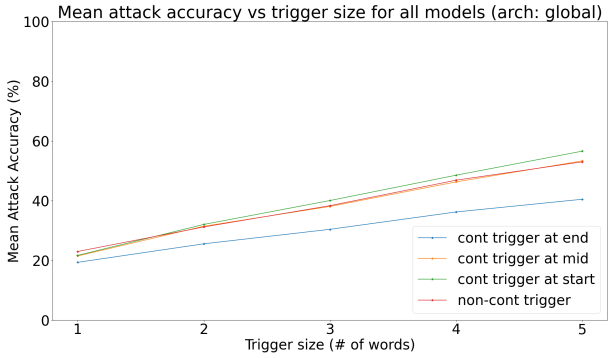
Even though the second architecture (Table 4.7) is very similar, we do not see the same behavior. The attack accuracy is slightly increased only for the non-continuous trigger. On the other hand, for the continuous triggers, the attack's performance is dropped (see Figure B.9). The considerable reduction of the model's parameters in that case (~77%) makes the backdoor attack more difficult for continuous triggers. However, the backdoor's performance is similar regardless of the trigger position, which seems promising for a dynamic backdoor attack.

The backdoor attack behaves similarly for both versions of the third architecture (Table 4.8). We see only a slight increase in its accuracy when the global average pooling is used. In that case, the non-continuous trigger performs slightly better than the rest, but in general, the differences among the different triggers are minimal (see Figure B.11).

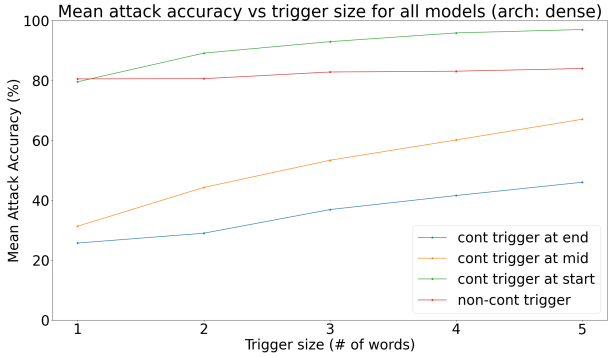
5.4. Dynamic Trojans with Global Average Pooling

In this section, we investigate the feasibility of dynamic triggers through a global average pooling layer. For this set of experiments, we keep constant the training epochs, the number of poisoned samples, and the trigger size to highlight only the effects of global average pooling. During the training, we poison each model with a trigger in a specific position. During the inference, though, we insert the trigger in different positions. In this way, we can verify if the neural network with the global average pooling can identify the trigger regardless of its exact position.

We run each of these experiments three times. The average attack accuracy is shown in Figure 5.7, Figure 5.8, and Figure 5.9 for the first architectures used in image classification, text sentiment analysis, and sound recognition, respectively. For the rest of them, the reader can consult Appendix B. The x-axis of these



(a) Models with global average pooling



(b) Models without global average pooling

Figure 5.5: Mean attack accuracy for different triggers in text classification

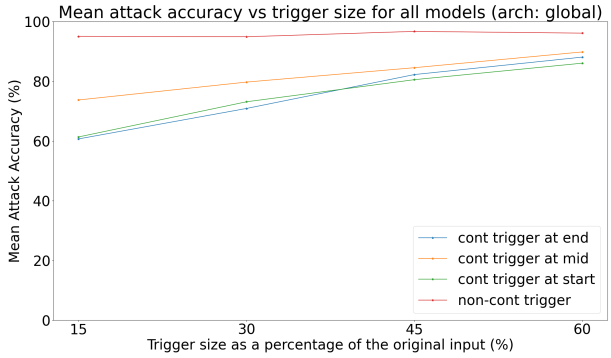
graphs shows the position of the trigger during inference, and the y-axis shows the attack accuracy. We plot a different line for each training trigger. In all the graphs, the dotted lines show the attack accuracy for models without the global average pooling and the normal ones for models with global average pooling.

5.4.1. Image Classification

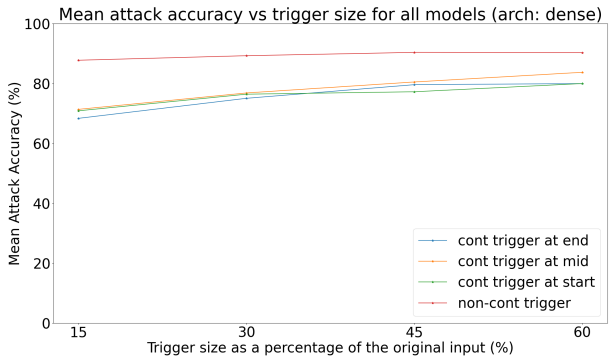
We trained the models for 50 epochs in image classification and poisoned 100 training samples with 64-pixel triggers. Both squares (8x8) and lines (2x32) were used. We plot the results for the first architecture in Figure 5.7. The results for square triggers are shown in Subfigure 5.7(a) and for the line triggers in Subfigure 5.7(b). From these graphs, we can see that in most cases, the backdoor attack is effective only when the exact trigger is used for training and inference regardless of the network version. This is also true for the second (Table 4.2) architecture and ResNet18 [50] (see Figure B.2 and Figure B.4). The networks are deep enough, and the max-pooling layers remove most of the spatial information. In the first architecture (Table 4.1), the 32x32x3 input image results in a 4x4x128 feature map right before the global average pooling. Thus the generalization that it introduces does not have a strong effect. We observe this behavior when the total reduction of trainable parameters is small (first and third architectures) and large (second architecture).

In some experiments, though, we see some interesting results. First, for a non-continuous training trigger and a line on the top as an inference trigger, the attack performs better when the global average pooling layer is used. In particular, the attack accuracy is increased from 12.82% to 36.85%. The same is true for the second architecture (see Figure B.2) but is not observed in ResNet18 (see Figure B.4). This does not necessarily suggest that a dynamic trigger is feasible because, as shown in subfigures 4.2(1) (64-pixel line on top) and 4.2(25) (non-continuous trigger of 64 pixels), there are eight common pixels between the two triggers. The generalization from the global average pooling seems to make this connection stronger.

Additionally, we see that the global average pooling makes the attack more effective when the inference trigger is a square in the middle and the training trigger is a square in one of the two corners. We see similar behavior in all the architectures that we tested. In the first architecture, the attack accuracy is increased



(a) Models with global average pooling



(b) Models without global average pooling

Figure 5.6: Mean attack accuracy for different triggers in sound classification

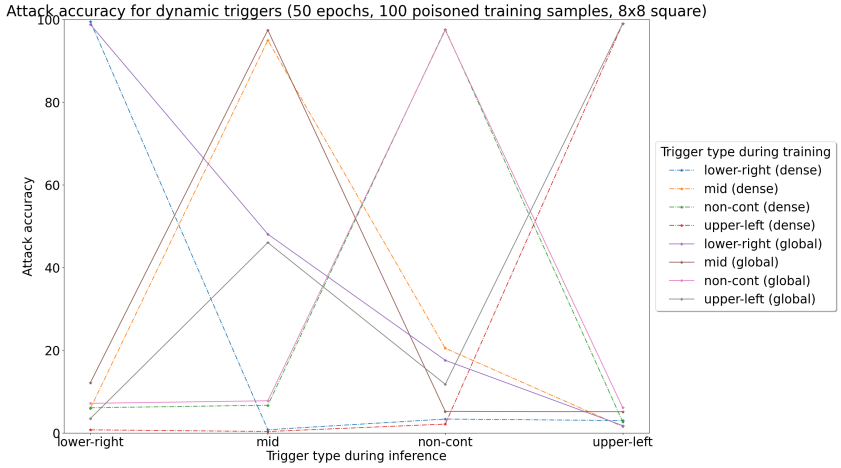
from 0.34% to 46% and from 0.77% to 48% for training triggers in the upper left and lower right corners, respectively. In the second architecture, the attack accuracy is increased from 0.7% to 59% and from 1.21% to 27% (see Figure B.2). In ResNet18, the attack accuracy is increased from 3% to 25% and from 9% to 37% for the same experiments (see Figure B.4).

This is the only case with some potential for implementing dynamic backdoors in image classification but requires further investigation and cannot lead to a general conclusion about implementing dynamic trojans with global average pooling.

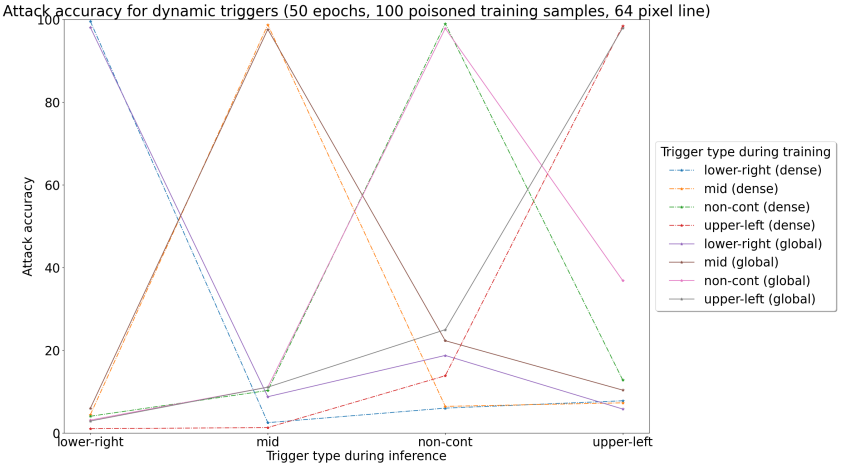
5.4.2. Text Sentiment Analysis

In text sentiment analysis, the models were trained for 20 epochs, and 100 training samples were poisoned with 5-word triggers. The results for the first architecture (Table 4.3) are shown in Figure 5.8. From this graph, it is clear that when a global average pooling layer is used, the attack accuracy is not affected for different triggers between training and inference. The attack's performance is identical for almost all the different training positions. Only when the training trigger is applied in the end, the attack accuracy is slightly dropped. As discussed in chapter 4, this is expected because some of the triggers inserted at the end of the sentence are removed when the sentences are truncated to 250 words. In other words, the backdoor attack has a very similar performance regardless of the trigger position during training and inference. We observe the same behavior in the third architecture (Table 4.5) when we use the global average pooling (see Figure B.8).

On the other hand, when the global average pooling is not used (dotted lines in Figure 5.8) in the first architecture (Table 4.3), the attack's performance varies a lot based on the trigger used for training and inference. When the inference trigger is applied in the end or in the middle of the sentence, the backdoor attack performs poorly (similar performance to the random guessing) for all the different training positions. However, when the inference trigger is inserted at the beginning of the sentence, the attack accuracy becomes more than 97% for all the different training positions. The neural network learns words regardless of their position, but more attention is given to words at the beginning of the sentence during inference. We observe



(a) Square triggers



(b) Line triggers

Figure 5.7: Attack accuracy for triggers in different positions between training and inference in image recognition. Both line and square triggers of 64 pixels were tried for 50 epochs of training and 100 poisoned training samples.

this trend also for the third architecture (see Figure B.8).

The second architecture, though (Table 4.4), does not learn in the same way. This architecture can identify continuous triggers everywhere in a sentence without needing the global average pooling. The max-over-time pooling layers [61] extract essential features from a sentence and discard the remaining information. As a result, if that architecture is used, we can create dynamic triggers that can be effective even if different positions are used for training and inference (see Figure B.6).

We can also make another important conclusion from Subfigure 4.11(b). Even though the generalization introduced from the global average pooling slightly improves the model's performance in the original task (subsection 5.3.1), the attack cannot be very successful. However, global average pooling makes dynamic triggers possible without any additional modification in the training dataset. This is a tradeoff that the attacker needs to consider when designing the attack strategy or a property that a defender could exploit to increase a model's robustness to backdoor attacks.

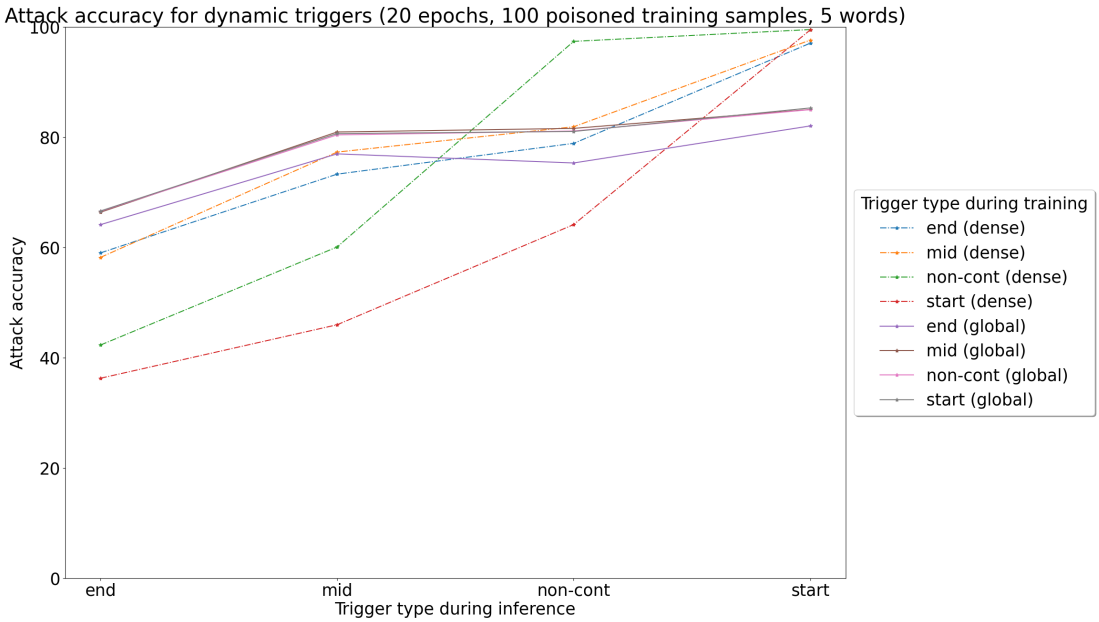


Figure 5.8: Attack accuracy for triggers in different positions between training and inference in text sentiment analysis. The models were trained for 20 epochs, 100 training samples were poisoned and 5 word triggers were used.

5.4.3. Sound Recognition

We train the models for 40 epochs in sound recognition, and we use 100 (1.83%) poisoned training samples. Additionally, we choose a small trigger (0.15 seconds) to avoid overlaps between different positions that larger triggers may introduce. As shown in Figure 5.9, for the first architecture (Table 4.6), only one dashed line shows high attack accuracy in each of the four different points in the x-axis. Each of these cases occurs when the exact trigger is used during training and inference. The attack accuracy for networks without the global average pooling layer remains pretty low when the triggers in training and inference are different. This is also true when we do not use the global average pooling in all three architectures that we tried (see Figure B.10 and Figure B.12). The attack accuracy for different triggers is slightly increased only when a non-continuous trigger is used during inference. As explained in subsection 4.4.5, the non-continuous trigger is the most effective as it affects multiple time windows during the calculation of the MFCCs. Thus, there is an increased chance that it overlaps with the continuous triggers used during training in this set of experiments.

On the other hand, when the global average pooling is used, different triggers between training and inference can be effective in any architecture that we tried (see Figure 5.9, Figure B.10, and Figure B.12). For example, in the first architecture, if during training we use a trigger in the end or in the beginning, the attack accuracy becomes 60.54% and 64.91%, respectively, for an inference trigger in the middle. These numbers become 8.49% and 3.07% without the global average pooling. Additionally, when the inference trigger is non-continuous, the attack accuracy is over 95% for all the different training riggers. Again, without the global average pooling, these numbers are reduced to 32.13%, 51.9%, and 34.82% for inference triggers in the end, middle, and beginning, respectively. We observe similar behavior for inference triggers at the beginning of the signals. This clarifies that for sound recognition, dynamic triggers can be made easily by substituting two layers (flatten + fully connected) with the global average pooling.

5.5. Conclusion

This chapter explored the properties of global average pooling and discussed its effects on the backdoor attacks for three applications (image classification, text sentiment analysis, and sound recognition). We drew the following conclusions:

- We can add the global average pooling in the neural networks used in chapter 4 with minor modifi-

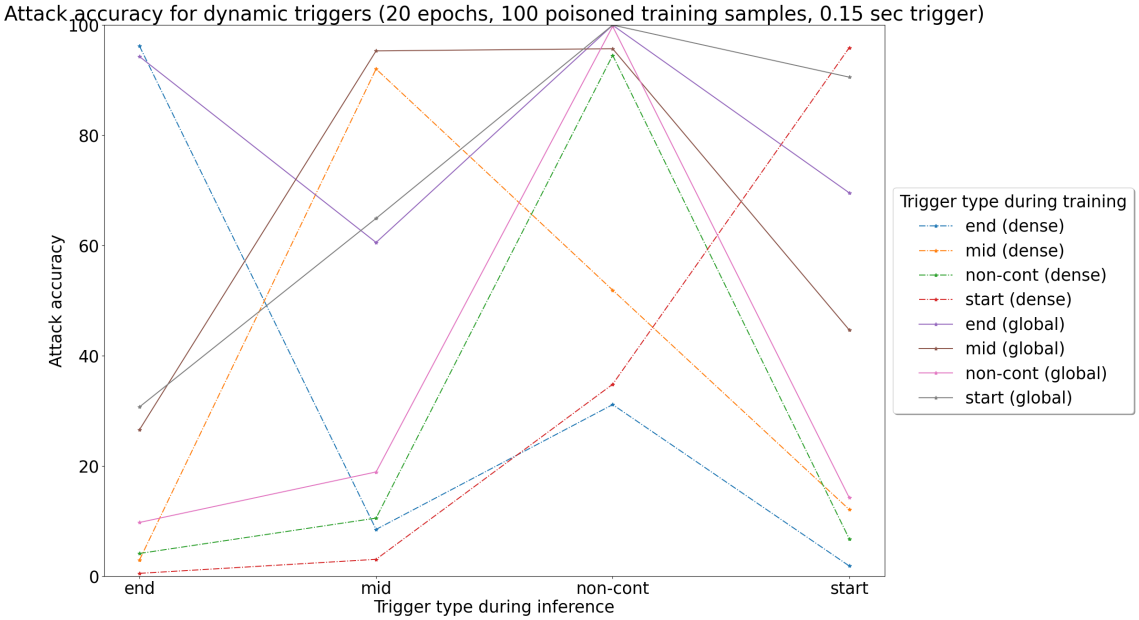


Figure 5.9: Attack accuracy for triggers in different positions between training and inference in sound recognition. The models were trained for 40 epochs, 100 training samples were poisoned and 0.15 second triggers were used.

cations. In particular, it can be added either by substituting a flatten layer (image classification) or by substituting a flatten and a fully connected layer (text sentiment analysis and sound recognition) without any devastating effects on the model's performance.

- Global average pooling alone cannot guarantee dynamic trojans in image classification. The attack accuracy is increased only in a few experiments for different triggers between training and inference, but we did not observe a general trend.
- In two different architectures for text sentiment analysis, the global average pooling makes the triggers effective even if a different position was used in training and inference. However, when this layer is not used, the attack can be more effective when the trigger is added in the beginning during inference. The other architecture uses max-over-time pooling layers, which discard much information from each sentence, and global average pooling does not introduce any new properties to the modeled function. Thus, dynamic backdoors can already be implemented in that case.
- In sound recognition, when the global average pooling is not used, the attack accuracy is low for different triggers between training and inference. In the opposite case, the attack can be quite successful even if we use different triggers in training and inference.
- To the best of our knowledge, we are the first to run experiments with different triggers in training and inference for all three applications showing that in some cases, global average pooling could lead to dynamic backdoor attacks.

6

Conclusions

In this thesis, we explore the behavior of backdoor attacks in neural networks. We have chosen three different applications for our experiments to detect common patterns that may exist. These applications are image classification (CIFAR-10 [65]), text sentiment analysis (IMDB's movie review [78]), and sound recognition (Speech commands [117]). This thesis is divided into two parts. In the first part (chapter 4), we showed how the effectiveness of the backdoor attack is affected by various trigger characteristics such as size, shape, and continuity. Additionally, we experimented with sound triggers in the inaudible human range (> 20 kHz). In the second part (chapter 5), we discuss the feasibility of dynamic backdoor attacks. In particular, we try to exploit the properties of the global average pooling layer in all three applications to create effective triggers in various positions without using more poisoned samples.

In this chapter, we answer the research questions based on the practical work done in chapter 4 and chapter 5, which summarizes the contributions of this project. Then we present its limitations and discuss potential future directions.

6.1. Contributions

What is the effect of trigger characteristics in different applications like image recognition, text sentiment analysis, and sound recognition?

First, we verified that the model's performance for the original task is unchanged for all the tested applications and architectures regardless of the trigger characteristics because only a few training samples need to be poisoned for the backdoor attack which agrees with various related works in the literature [22, 38, 70, 75].

For small triggers (64 pixels) in image classification, we showed that line triggers are more effective than squares. Additionally, we observed that the best performing triggers are lines in the middle and non-continuous triggers. If the trigger is large enough (196 or 289 pixels), though, the differences between them are insignificant. We did not find a clear winner for the position of square triggers because their performance varied for different sizes. In sound recognition, the position of continuous triggers did not play a significant role, and all of them showed very similar performance. For this application, it was proved that the non-continuous triggers are the most effective. The relation of the trigger's position and the attack accuracy in text sentiment analysis depends on the model. In some cases, the trained model included information about the position of the words, while others did not show such behavior.

Furthermore, we verified that the attack accuracy is positively correlated to the trigger size in most cases for all the applications. However, in sound recognition, this relation is not very strong as the features used by the network (MFCCs) calculate the fast Fourier transform, and only a fraction of the time-domain information is retained. Additionally, we saw that a successful classifier in text sentiment analysis could focus on each sentence's most important words or phrases because they can be strongly connected with the sentence's sentiment. Our experiments used such a neural network that emphasized phrases of 3,4 and 5 words. In that case, the trigger size did not play any role as the network could identify all the triggers that we used (1 to 5 words).

Last but not least, we showed that inaudible sounds (> 20 kHz) could be used as triggers in sound recognition which is very promising for stealthy real-word attacks.

Is it possible to exploit global average pooling and create dynamic backdoors in different application

domains like image classification, text sentiment analysis, and sound recognition without increasing the number of poisoned samples?

Partially yes. We are the first to show that in text sentiment analysis and sound recognition, the global average pooling could allow dynamic backdoors without altering the poisoning procedure or increasing the number of poisoned samples. However, in some cases in text sentiment analysis, the dynamic backdoor can be possible without altering the architecture, especially for continuous triggers. For this application, sometimes it is enough to find the more important words or phrases in a sentence. In image classification, though, no such conclusion could be made from our experiments.

We also saw that the global average pooling could be added in existing architectures with minor modifications. Additionally, we showed that the global average pooling layer could increase the accuracy for the original task in some cases. Furthermore, we saw that when a model generalizes well, the backdoor attack needs more effort, which seems to be a promising direction for future provable defenses.

6.2. Limitations and Future Work

Although some interesting conclusions were made during this work, some limitations could still lead to future work and directions.

First, we used only one dataset for each application. Although the chosen datasets are pretty popular in the deep learning community, they are small, and the conclusions that we drew may be irrelevant for larger and more complex datasets. Thus, in the future, it could be helpful to run the same experiments in more complex datasets like ImageNet [34] for image classification, the Enron corpus [63] and the Amazon reviews [82] for text sentiment analysis, and the entire speech commands dataset [117] for sound recognition. This could lead to more robust conclusions about the backdoor's behavior in more realistic scenarios.

The experiments that we tried in chapter 4 are far from exhaustive. We left out of the loop many trigger characteristics. For example, only one target class was tried both in image recognition ("horse"), text sentiment analysis ("negative"), and sound classification ("down"). As mentioned in [98], the best position for a trigger may vary among different target classes. Thus, it would be interesting to find the best position of the trigger for each target class in the dataset. Moreover, we could experiment with the trigger's opacity in image classification and investigate its effect on the attack accuracy.

Additionally, in text classification, we have only used trainable embeddings, which means that our backdoor is also inserted in them through training. In practice, though, pretrained embeddings are more common because they have been already trained in large corpora of text and interpret more accurately possible connections between different words. 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 [85]. Glove is trained from a corpus of 6 billion words and has a vocabulary of 400000 words [94]. We should investigate if we can draw the same conclusions about backdoor attacks in text sentiment analysis when pretrained embeddings are used.

All the trigger characteristics that we investigated in sound recognition are related to the signal's time domain. For example, the trigger's size is its duration, and the trigger's location is a specific time frame inside the 1-second recordings. The spectrum remained constant from these changes as the signal was a sinusoidal pulse with a constant frequency of 21 kHz. The MFCCs, though, use the FFT internally. Thus, it is essential to check what happens if this spectrum is changed for a better view of the backdoor attack and its properties. More specifically, we can design a new trigger containing more frequency components (size in the frequency domain) or a different range of frequencies (position in the spectrum).

During this project, the need for a framework that creates poisoned datasets for different applications was made crystal clear. Even though such a library already exists in python [59], it does not support sound datasets yet. This discouraged us from using it for this project. Thus, a universal tool that can generate backdoored datasets for many different applications based on user-provided options could be very useful for the research community and fast comparison of different attacks and defenses.

As shown in chapter 4, inaudible sounds can be used as triggers. Inspired from the attack described in [126], we could apply this backdoor attack in a real-world setting where some malicious functionality could be embedded in an existing system (biometric or speaker verification) and triggered by the inaudible trigger we produced in this work.

In chapter 4, we saw that a square trigger in the middle was more effective than the two corners when using the ResNet18 architecture. This behavior was not observed in the other two architectures, which indicates that ResNet18 learns differently about the dataset's features. A future direction of this work and backdoors, in general, could be to use them for a deeper understanding of how neural networks learn. Such an approach

can complement the work described in [124], where the authors observed the model's output after graying out small squares of the input images, which led to valuable insights about the crucial features of an input image.

In some experiments, we saw that when the model's generalization is strong, more effort (more training epochs or more poisoned samples) is required for a successful backdoor attack. Additionally, each of our attacks required only a few poisoned training samples, which is in line with the threat model used in the literature [22, 23, 47, 75]. Thus, we concluded that robust generalization techniques and methods that prevent learning from features in very few training samples could result in quite effective defenses.

Recurrent neural networks are used in many NLP applications because they effectively process data [32]. However, in this work, we experimented with backdoor attacks only in convolutional neural networks. While they performed well for the binary classification, maybe they are not suitable for more complex NLP tasks. Thus, in the future, we could run our experiments for such models also.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] A. Adithyan, K. Nagendran, R. Chethana, G. Pandy D., and G. Prashanth K. Reverse engineering and backdooring router firmwares. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 189–193, 2020. doi: 10.1109/ICACCS48705.2020.9074317.

[3] Tarem Ahmed, Boris Oreshkin, and Mark Coates. Machine learning approaches to network anomaly detection. In *Proceedings of the 2nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, SYSML'07, USA, 2007. USENIX Association.

[4] Amazon. Amazon's alexa. https://developer.amazon.com/en-US/alexa, 2014. [Online, Accessed: May 30, 2021].

[5] Apple. Apple's siri. https://www.apple.com/siri/, 2010. [Online, Accessed: May 30, 2021].

[6] AWS. Aws sagemaker. https://docs.aws.amazon.com/sagemaker/index.html, 2017. [Online, Accessed: May 30, 2021].

[7] Ahmadreza Azizi, Ibrahim Asadullah Tahmid, Asim Waheed, Neal Mangaokar, Jiameng Pu, Mobin Javed, Chandan K. Reddy, and Bimal Viswanath. T-miner: A generative approach to defend against trojan attacks on dnn-based text classification. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, August 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/azizi.

[8] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models, 2021.

[9] Chris Bagwell. Sound eXchange (SoX): HomePage, 2015. URL http://sox.sourceforge.net/. [Online, Accessed: May 30, 2021].

[10] M. Barni, K. Kallas, and B. Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105, 2019. doi: 10.1109/ICIP.2019.8802997.

[11] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines, 2013.

[12] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car, 2017.

[13] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. A statistical approach to language translation. In *Proceedings of the 12th Conference on Computational Linguistics - Volume 1*, COLING '88, page 71–76, USA, 1988. Association for Computational Linguistics. ISBN 963 8431 56 3. doi: 10.3115/991635.991651. URL https://doi.org/10.3115/991635.991651.

[14] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June 1993. ISSN 0891-2017.

[15] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017. doi: 10.1109/SP.2017.49.

[16] Nicholas Carlini and David A. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. *CoRR*, abs/1801.01944, 2018. URL http://arxiv.org/abs/1801.01944.

[17] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 513–530, Austin, TX, August 2016. USENIX Association. ISBN 978-1-931971-32-4. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/carlini.

[18] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering, 2018.

[19] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4658–4664. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/647. URL https://doi.org/10.24963/ijcai.2019/647.

[20] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, page 15–26, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450352024. doi: 10.1145/3128572.3140448. URL https://doi.org/10.1145/3128572.3140448.

[21] Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. Badnl: Backdoor attacks against nlp models, 2020.

[22] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning, 2017.

[23] Edward Chou, Florian Tramèr, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attacks against deep learning systems. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 48–54. IEEE, 2020.

[24] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

[25] D. Conway and J.M. White. *Machine Learning for Email: Spam Filtering and Priority Inbox*. O'Reilly Media, 2011. ISBN 9781449320706. URL https://books.google.nl/books?id=_ZqUZVDm_bAC.

[26] Alexis Cook. Global average pooling layers for object localization. https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/, 2017. [Online, Accessed: May 30, 2021].

[27] corbet. An attempt to backdoor the kernel. https://lwn.net/Articles/57135/, 2003.

[28] corbet. Vsftpd backdoor discovered in source code (the h). https://lwn.net/Articles/450181/, 2011.

[29] Robby Costales, Chengzhi Mao, Raphael Norwitz, Bryan Kim, and Junfeng Yang. Live trojan attacks on deep neural networks, 2020.

[30] G. E. Dahl, T. N. Sainath, and G. E. Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8609–8613, 2013. doi: 10.1109/ICASSP.2013.6639346.

[31] George E. Dahl, Jack W. Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3422–3426, 2013. doi: 10.1109/ICASSP.2013.6638293.

[32] Jiazhu Dai and Chuanshuai Chen. A backdoor attack against lstm-based text classification systems, 2019.

[33] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980. doi: 10.1109/TASSP.1980.1163420.

[34] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[35] Murat Dikmen and Catherine M. Burns. 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*, Automotive'UI 16, page 225–228, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450345330. doi: 10.1145/3003715.3005465. URL https://doi.org/10.1145/3003715.3005465.

[36] Bao Gia Doan, Ehsan Abbasnejad, and Damith C. Ranasinghe. Februus: Input purification defense against trojan attacks on deep neural network systems. In *Annual Computer Security Applications Conference*, ACSAC '20, page 897–912, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450388580. doi: 10.1145/3427228.3427264. URL https://doi.org/10.1145/3427228.3427264.

[37] Hironobu Fujiyoshi, Tsubasa Hirakawa, and Takayoshi Yamashita. Deep learning-based image recognition for autonomous driving. *IATSS Research*, 43(4):244–252, 2019. ISSN 0386-1112. doi: https://doi.org/10.1016/j.iatssr.2019.11.008. URL https://www.sciencedirect.com/science/article/pii/S0386111219301566.

[38] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, page 113–125, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450376280. doi: 10.1145/3359789.3359790. URL https://doi.org/10.1145/3359789.3359790.

[39] Yansong Gao, Bao Gia Doan, Zhi Zhang, Siqi Ma, Jiliang Zhang, Anmin Fu, Surya Nepal, and Hyoungshick Kim. Backdoor attacks and countermeasures on deep learning: A comprehensive review, 2020.

[40] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses, 2021.

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[42] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[43] Google. Google ai platform. https://cloud.google.com/ai-platform, 2018. [Online, Accessed: May 30, 2021].

[44] Google. Text classification with movie reviews. https://www.tensorflow.org/tutorials/keras/text_classification, 2020. [Online, Accessed: May 30, 2021].

[45] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013. doi: 10.1109/ICASSP.2013.6638947.

[46] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd edition, 2019. ISBN 1491962291.

[47] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019. doi: 10.1109/ACCESS.2019.2909068.

[48] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition, 2014.

[49] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. doi: 10.1109/ICCV.2015.123.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[51] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[52] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[53] IBM. Data science and machine learning hub. https://www.ibm.com/analytics/machine-learning, 2017. [Online, Accessed: May 30, 2021].

[54] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/ioffe15.html.

[55] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35, 2018. doi: 10.1109/SP.2018.00057.

[56] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35, 2018. doi: 10.1109/SP.2018.00057.

[57] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (3rd Edition - Draft)*. Prentice-Hall, Inc., 2020. ISBN 0131873210.

[58] Uday Kamath, John Liu, and James Whitaker. *Deep Learning for NLP and Speech Recognition*. Springer, 01 2019. ISBN 978-3-030-14595-8. doi: 10.1007/978-3-030-14596-5.

[59] Kiran Karra, Chace Ashcraft, and Neil Fendley. The trojai software framework: An opensource tool for embedding trojans into deep learning models, 2020.

[60] Dmitriy Kharutskiy. Sound spectrum analysis, 2018. URL https://apps.apple.com/us/app/sound-spectrum-analysis/id1434975523. [Online, Accessed: May 30, 2021].

[61] Yoon Kim. Convolutional neural networks for sentence classification, 2014.

[62] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[63] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226. Springer, 2004.

[64] Eric Knorr. How paypal beats the bad guys with machine learning. https://www.infoworld.com/article/2907877/how-paypal-reduces-fraud-with-machine-learning.html, 2015. [Online, Accessed: May 30, 2021].

[65] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[67] Ram Shankar Siva Kumar, David R. O'Brien, Kendra Albert, Salomé Viljöen, and Jeffrey Snover. Failure modes in machine learning systems. *CoRR*, abs/1911.11034, 2019. URL http://arxiv.org/abs/1911.11034.

[68] Quan Le, Oisín Boydell, Brian Mac Namee, and Mark Scanlon. Deep learning at the shallow end: Malware classification for non-domain experts. *Digital Investigation*, 26:S118–S126, 2018. ISSN 1742-2876. doi: https://doi.org/10.1016/j.diin.2018.04.024. URL https://www.sciencedirect.com/science/article/pii/S1742287618302032.

[69] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *Proceedings 2019 Network and Distributed System Security Symposium*, 2019. doi: 10.14722/ndss.2019.23138. URL http://dx.doi.org/10.14722/ndss.2019.23138.

[70] Shaofeng Li, Minhui Xue, Benjamin Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[71] Yiming Li, Tongqing Zhai, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shutao Xia. Rethinking the trigger of backdoor attack, 2020.

[72] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey, 2021.

[73] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network, 2014.

[74] Y. Liu, A. Mondal, A. Chakraborty, M. Zuzak, N. Jacobsen, D. Xing, and A. Srivastava. A survey on neural trojans. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*, pages 33–39, 2020. doi: 10.1109/ISQED48828.2020.9137011.

[75] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018.

[76] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1265–1282, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. doi: 10.1145/3319535.3363216. URL https://doi.org/10.1145/3319535.3363216.

[77] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans, 2017.

[78] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

[79] Alexey Malanov. The multilayered security model in kaspersky lab products. https://www.kaspersky.com/blog/multilayered-approach/6601/, 2017. [Online, Accessed: May 30, 2021].

[80] Sandya Mannarswamy. Everything you need to know about neural networks. https://www.opensourceforu.com/2017/03/neural-networks-in-detail/, 2017. [Online, Accessed: May 30, 2021].

[81] Akshay Mattoo. Imdb sentiment analysis. https://github.com/matakshay/IMDB_Sentiment_Analysis, 2020. [Online, Accessed: May 30, 2021].

[82] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 43–52, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336215. doi: 10.1145/2766462.2767755. URL https://doi.org/10.1145/2766462.2767755.

[83] David Mickisch, Felix Assion, Florens Greßner, Wiebke Günther, and Mariele Motta. Understanding the decision boundary of deep neural networks: An empirical study, 2020.

[84] Microsoft. Azure machine learning. https://azure.microsoft.com/en-us/services/machine-learning/, 2019. [Online, Accessed: May 30, 2021].

[85] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[86] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997. ISBN 978-0-07-042807-2.

[87] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020.

[88] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

[89] NIST. Trojans in artificial intelligence. https://www.nist.gov/itl/ssd/trojai, 2019. [Online, Accessed: May 30, 2021].

[90] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. *Signals & Systems (2nd Ed.)*. Prentice-Hall, Inc., USA, 1996. ISBN 0138147574.

[91] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597, 2016. doi: 10.1109/SP.2016.41.

[92] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016. URL http://arxiv.org/abs/1605.07277.

[93] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Vgg face descriptor. https://www.robots.ox.ac.uk/~vgg/software/vgg_face/, 2015. [Online, Accessed: May 30, 2021].

[94] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[95] Arnab Poddar, Md. Sahidullah, and G. Saha. Speaker verification with short utterances: a review of challenges, trends and opportunities. *IET Biom.*, 7:91–101, 2018.

[96] Vinay Uday Prabhu and Abeba Birhane. Large image datasets: A pyrrhic win for computer vision? *CoRR*, abs/2006.16923, 2020. URL https://arxiv.org/abs/2006.16923.

[97] E. Quiring and K. Rieck. Backdooring and poisoning neural networks with image-scaling attacks. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 41–47, 2020. doi: 10.1109/SPW50608.2020.00024.

[98] A. S. Rakin, Z. He, and D. Fan. Tbt: Targeted neural network attack with bit trojan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13195–13204, 2020. doi: 10.1109/CVPR42600.2020.01321.

[99] Jordan Robertson and Michael Riley. The big hack: How china used a tiny chip to infiltrate u.s. companies. https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies, 2018. [Online, Accessed: May 30, 2021].

[100] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic backdoor attacks against machine learning models, 2020.

[101] Saeid Samizade, Zheng-Hua Tan, Chao Shen, and Xiaohong Guan. Adversarial example detection by classification for deep speech recognition. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3102–3106, 2020. doi: 10.1109/ICASSP40776.2020.9054750.

[102] Bruce Schneier. *Security Is a Weakest-Link Problem*, pages 103–117. Springer New York, New York, NY, 2003. ISBN 978-0-387-21712-3. doi: 10.1007/0-387-21712-6_8. URL https://doi.org/10.1007/0-387-21712-6_8.

[103] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1528–1540, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978392. URL https://doi.org/10.1145/2976749.2978392.

[104] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016. ISSN 1476-4687. doi: 10.1038/nature16961. URL https://doi.org/10.1038/nature16961.

[105] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[106] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15 (56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[107] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199.

[108] Te Juin Lester Tan and Reza Shokri. Bypassing backdoor detection algorithms in deep learning, 2020.

[109] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection, 2020.

[110] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[111] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks, 2018.

[112] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks, 2019.

[113] Valerio Velardo. Deep learning audio application. https://github.com/musikalkemist/Deep-Learning-Audio-Application-From-Design-to-Deployment, 2020. [Online, Accessed: May 30, 2021].

[114] VideoLan. Vlc media player. https://www.videolan.org/vlc/index.html, 2006.

[115] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[116] Ruxin Wang, Congying Han, Yanping Wu, and Tiande Guo. Fingerprint classification based on depth neural network. *CoRR*, abs/1409.5188, 2014. URL http://arxiv.org/abs/1409.5188.

[117] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209, 2018. URL http://arxiv.org/abs/1804.03209.

[118] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*, pages 529–534, 2011. doi: 10.1109/CVPR.2011.5995566.

[119] M.J. Wolf, K.W. Miller, and F.S. Grodzinsky. Why we should have seen that coming: Comments on microsoft's tay "experiment," and wider implications. *The ORBIT Journal*, 1(2):1–12, 2017. ISSN 2515-8562. doi: https://doi.org/10.29297/orbit.v1i2.49. URL https://www.sciencedirect.com/science/article/pii/S2515856220300493.

[120] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[121] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor. Hardware trojans: Lessons learned after one decade of research. *ACM Trans. Des. Autom. Electron. Syst.*, 22(1), May 2016. ISSN 1084-4309. doi: 10.1145/2906147. URL https://doi.org/10.1145/2906147.

[122] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. *arXiv preprint arXiv:1910.03137*, 2019.

[123] Jonas Zaddach, Anil Kurmus, Davide Balzarotti, Erik-Oliver Blass, Aurélien Francillon, Travis Goodspeed, Moitrayee Gupta, and Ioannis Koltsidas. Implementation and implications of a stealth hard-drive backdoor. In *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC '13, page 279–288, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320153. doi: 10.1145/2523649.2523661. URL https://doi.org/10.1145/2523649.2523661.

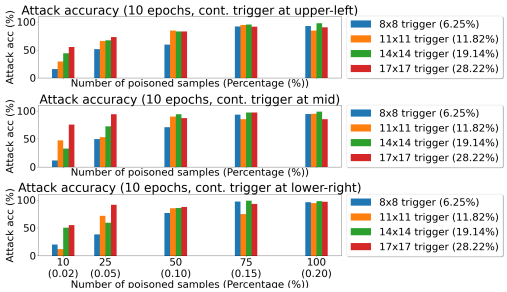
[124] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013.

[125] Tongqing Zhai, Yiming Li, Ziqi Zhang, Baoyuan Wu, Yong Jiang, and Shu-Tao Xia. Backdoor attack against speaker verification, 2021.

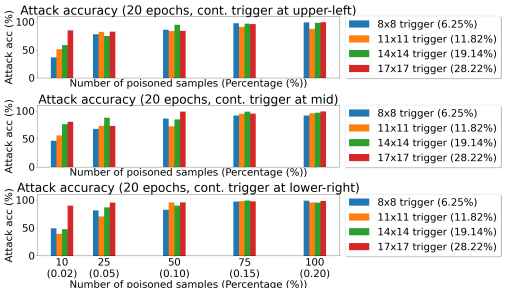
[126] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 103–117, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3134052. URL https://doi.org/10.1145/3133956.3134052.

A

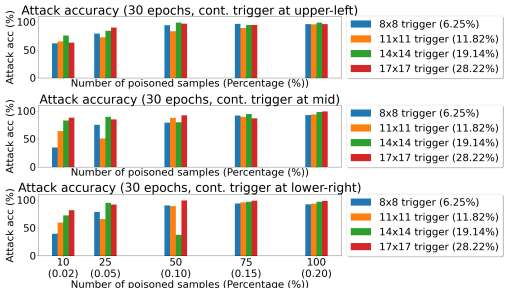
Additional Experiments for Chapter 4



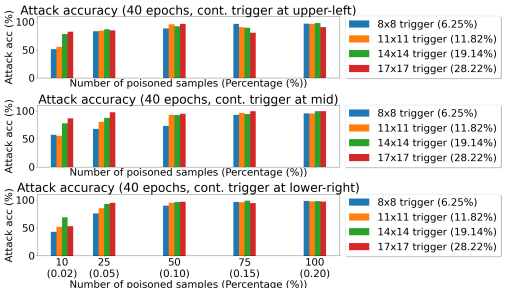
(a) Attack's accuracy on image classification for a square trigger and 10 epochs of training



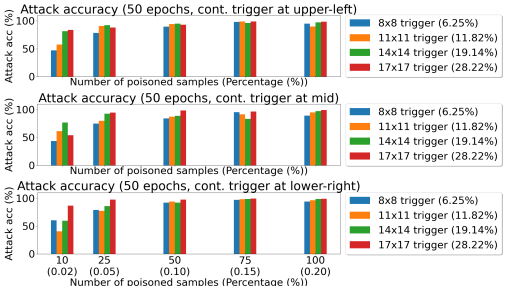
(b) Attack's accuracy on image classification for a square trigger and 20 epochs of training



(c) Attack's accuracy on image classification for a square trigger and 30 epochs of training



(d) Attack's accuracy on image classification for a square trigger and 40 epochs of training



(e) Attack's accuracy on image classification for a square trigger and 50 epochs of training

Figure A.1: Attack accuracy in image classification for square triggers of size from 8x8 to 17x17 pixels for the second architecture (Table 4.2). For more than 50 poisoned samples, the attack accuracy is high for all the triggers. In any other case, large triggers seem to be more effective. Additionally, we see similar behavior for all the different positions tried.

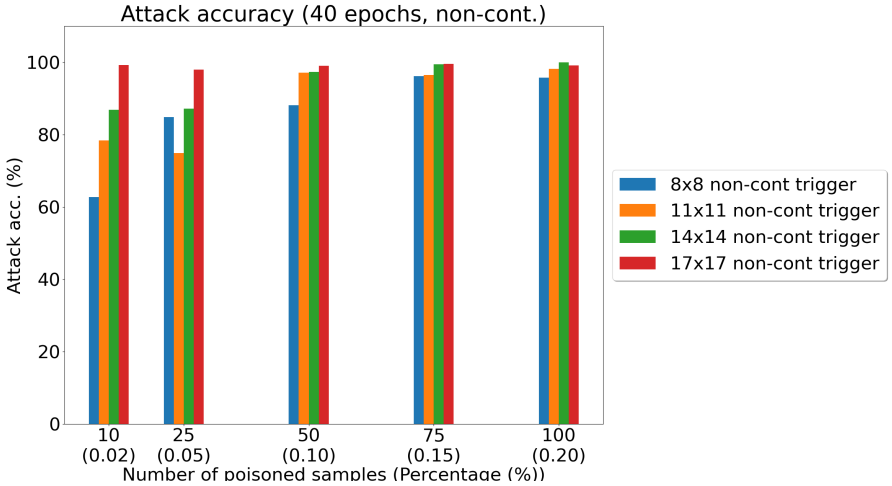


Figure A.2: Attack accuracy for non-continuous triggers in image classification for 40 epochs of training and the second architecture (Table 4.2). The attack accuracy increases as the trigger gets larger. However, when many poisoned samples are used (50 or more), the attack accuracy is high regardless of the size.

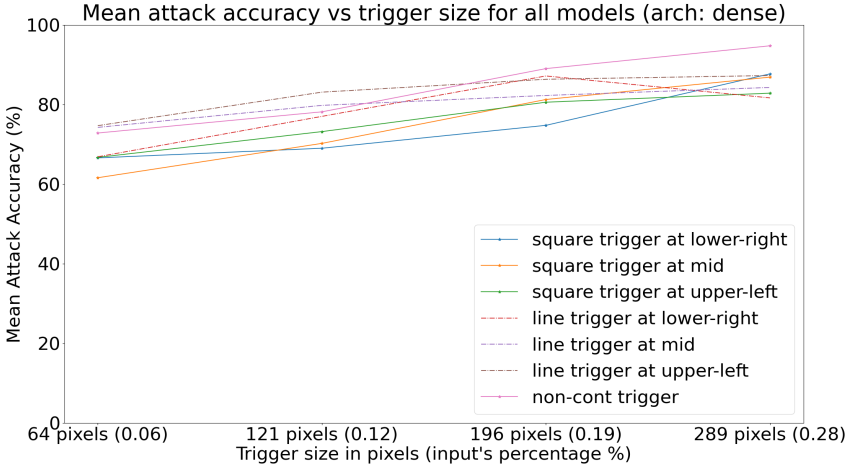
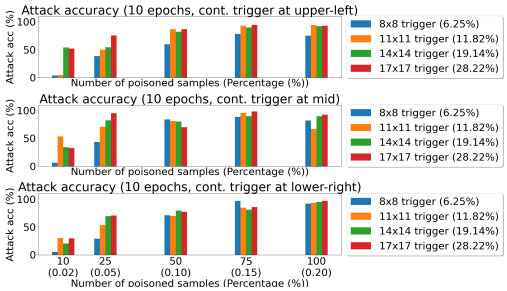
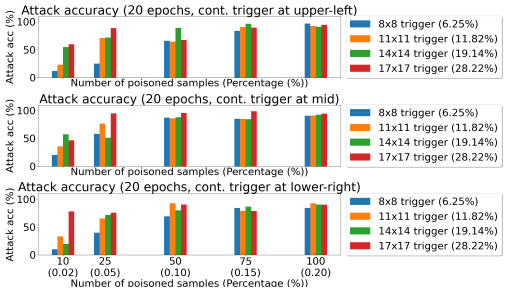


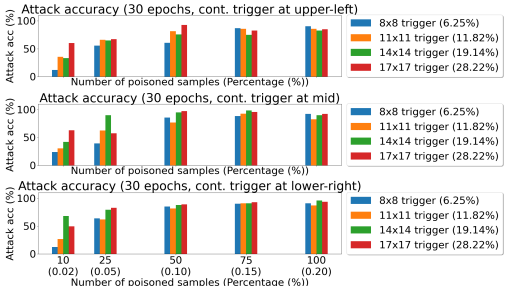
Figure A.3: Mean attack accuracy for image classification and the second architecture (Table 4.2). Large non-continuous triggers (196 or 289 pixels) show the best performance. Also, lines are more effective than squares, especially when the trigger is small (196 pixels or less).



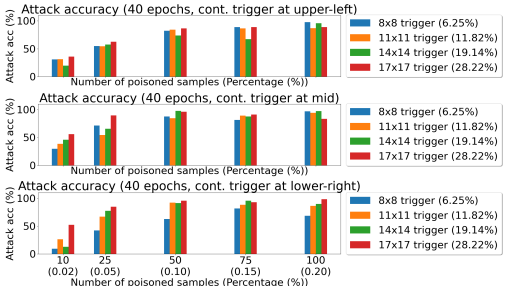
(a) Attack's accuracy on image classification for a square trigger and 10 epochs of training



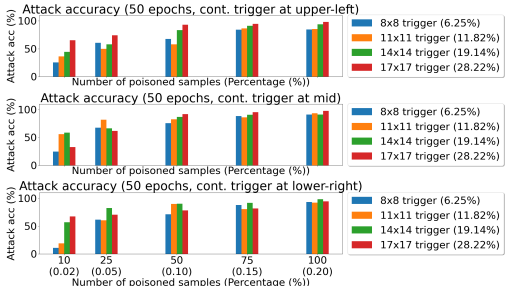
(b) Attack's accuracy on image classification for a square trigger and 20 epochs of training



(c) Attack's accuracy on image classification for a square trigger and 30 epochs of training



(d) Attack's accuracy on image classification for a square trigger and 40 epochs of training



(e) Attack's accuracy on image classification for a square trigger and 50 epochs of training

Figure A.4: Attack accuracy in image classification for square triggers of size from 8x8 to 17x17 pixels when ResNet18 is used [50]. For more than 50 poisoned samples, the attack accuracy is high for all the triggers. In any other case, large triggers seem to be more effective. The square trigger seems to perform better in the middle, especially for smaller triggers

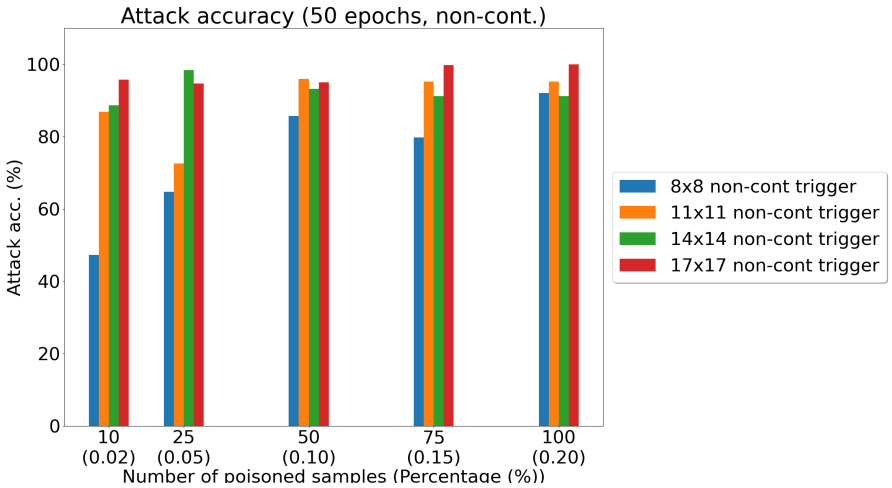


Figure A.5: Attack accuracy for non-continuous triggers in image classification for 40 epochs of training with the ResNet18 architecture [50]. The attack accuracy increases as the trigger gets larger. However, when many poisoned samples are used (50 or more), the attack accuracy is high regardless of the size.

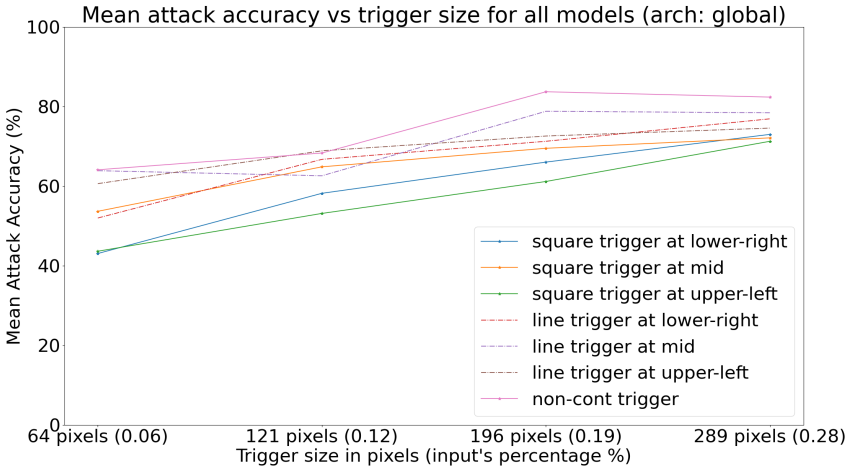
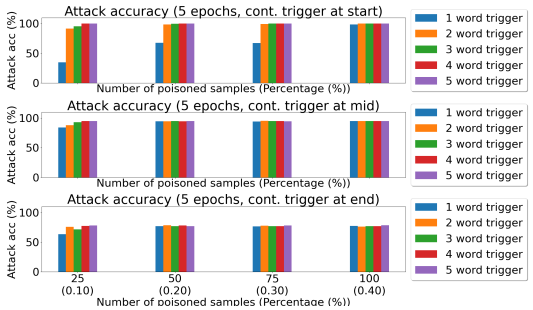
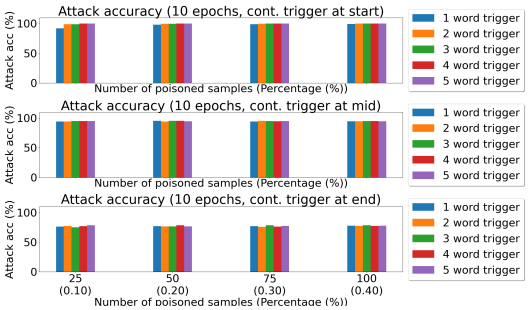


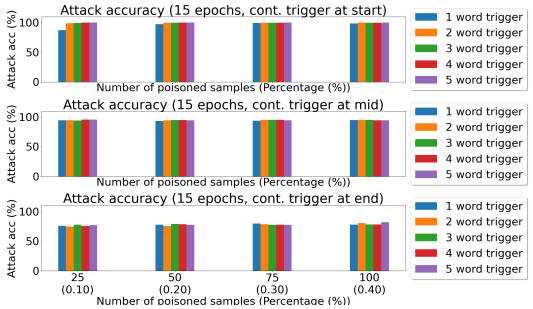
Figure A.6: Mean attack accuracy for image classification and the third architecture (Table 4.2). Large non-continuous triggers (196 or 289 pixels) show the best performance. Also, lines are more effective than squares when the trigger is 196 pixels or less.



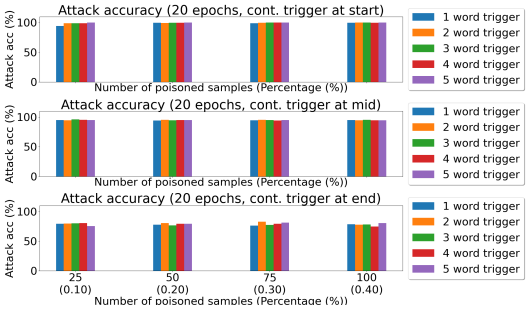
(a) Attack's accuracy on sentiment analysis for 5 epochs of training



(b) Attack's accuracy on sentiment analysis for 10 epochs of training



(c) Attack's accuracy on sentiment analysis for 15 epochs of training



(d) Attack's accuracy on sentiment analysis for 20 epochs of training

Figure A.7: Attack accuracy in sentiment analysis for continuous triggers (1 to 5 words) and the second architecture (Table 4.4). The attack accuracy is almost identical for all the trigger sizes due to the max-over-time pooling layers. Triggers in the beginning and the middle are slightly better than in the end.

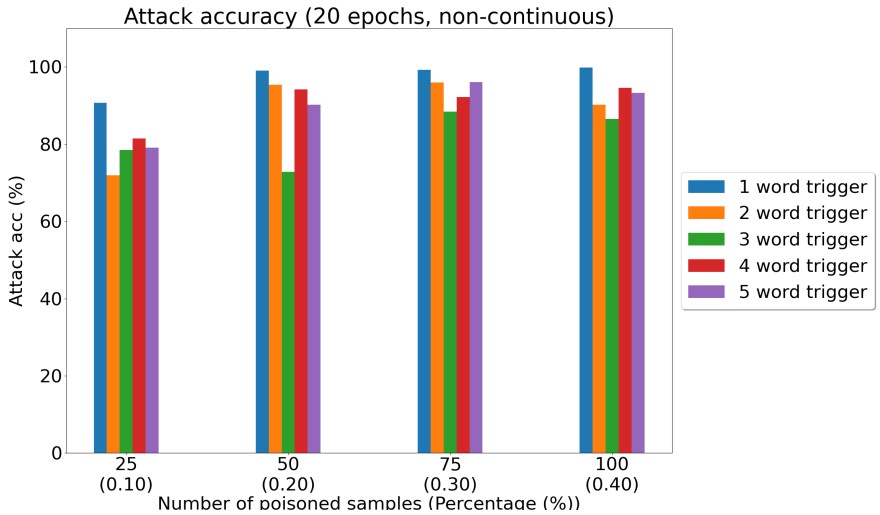


Figure A.8: Attack accuracy for non-continuous triggers in sentiment analysis, for 20 training epochs and the second architecture (Table 4.4). This architecture does not recognize efficiently non-continuous triggers because it focuses only on 3, 4, or 5-word phrases.

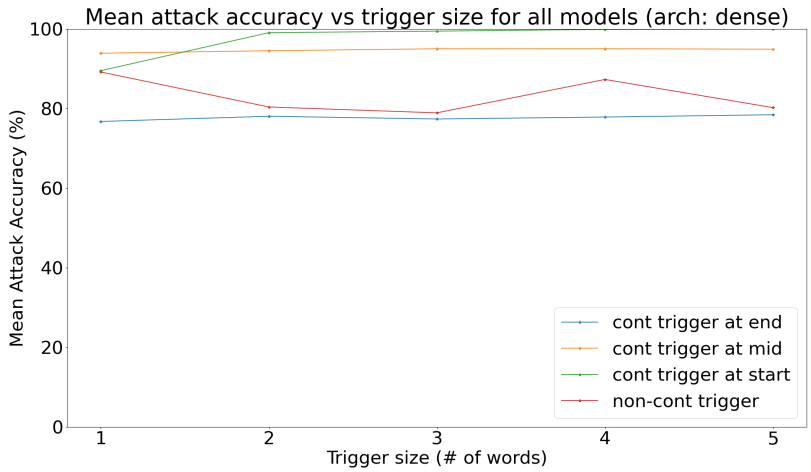
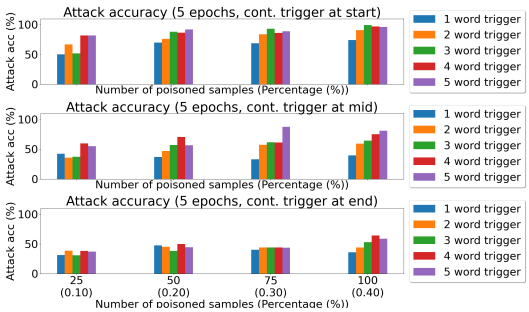
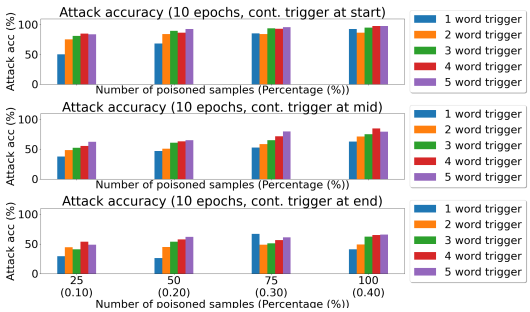


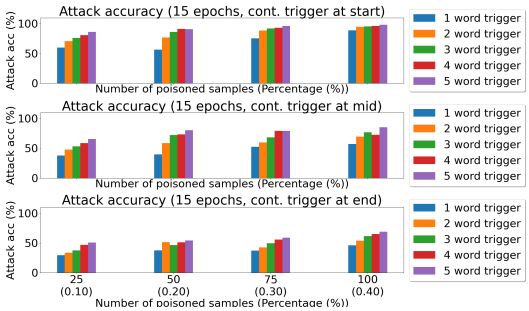
Figure A.9: Mean attack accuracy for text sentiment analysis for the second architecture (Table 4.4). The trigger in the beginning shows the best performance. Additionally, the trigger size does not affect the attack accuracy due to the model's architecture.



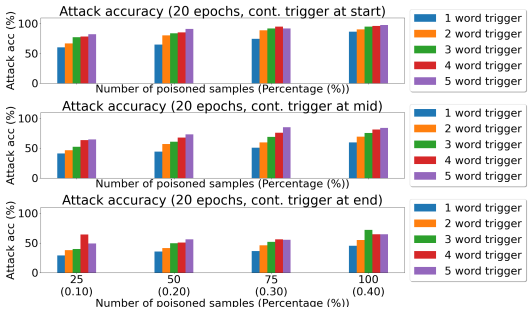
(a) Attack's accuracy on sentiment analysis for 5 epochs of training



(b) Attack's accuracy on sentiment analysis for 10 epochs of training



(c) Attack's accuracy on sentiment analysis for 15 epochs of training



(d) Attack's accuracy on sentiment analysis for 20 epochs of training

Figure A.10: Attack accuracy in sentiment analysis for continuous triggers (1 to 5 words) and the third architecture (Table 4.5). There is a clear correlation between the trigger size and the attack accuracy. Additionally, the trigger in the beginning shows the best performance.

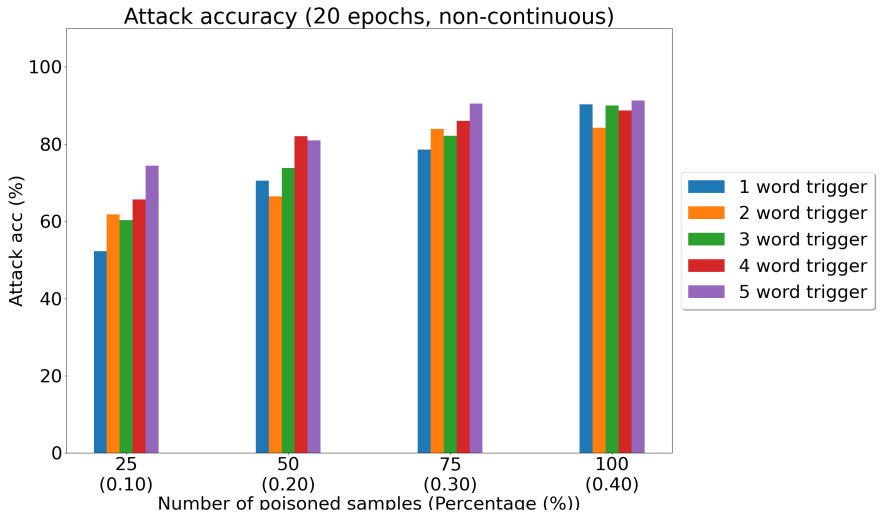


Figure A.11: Attack accuracy for non-continuous triggers in sentiment analysis, for 20 training epochs and the second architecture (Table 4.5). The attack accuracy is increased as the trigger size increases, but not always in a linear way.

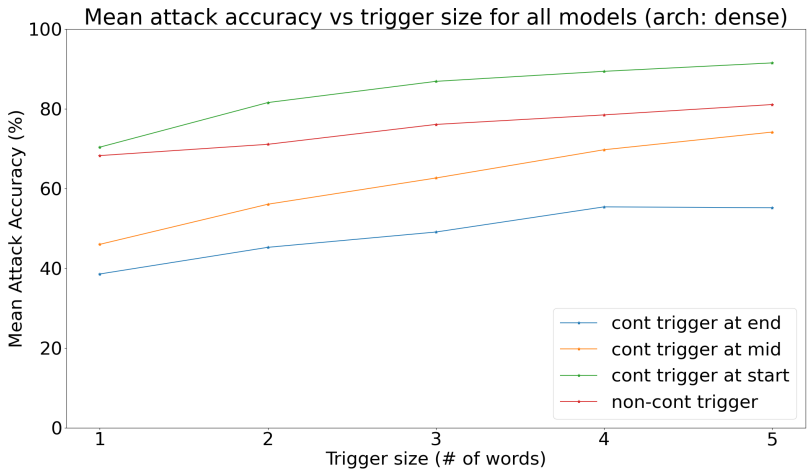
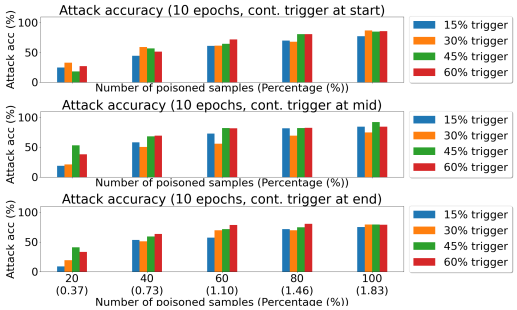
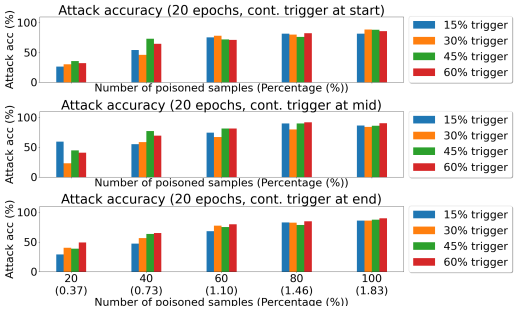


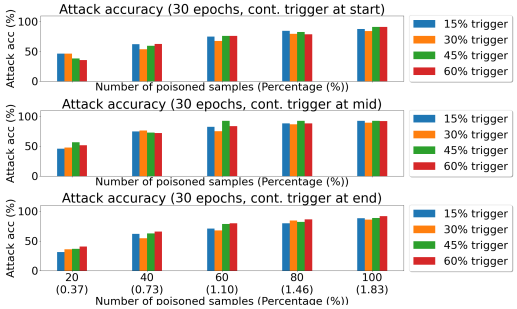
Figure A.12: Mean attack accuracy for text sentiment analysis for the third architecture (Table 4.5). We see a linear relation between the trigger size and the attack accuracy, and the best position is in the beginning.



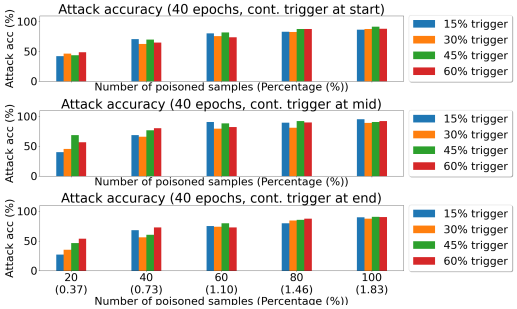
(a) Attack's accuracy on sentiment analysis for 10 epochs of training



(b) Attack's accuracy on sentiment analysis for 20 epochs of training



(c) Attack's accuracy on sentiment analysis for 30 epochs of training



(d) Attack's accuracy on sentiment analysis for 40 epochs of training

Figure A.13: Attack accuracy in speech recognition for continuous triggers for the second architecture (Table 4.7). No clear correlation between trigger duration and attack accuracy is shown.

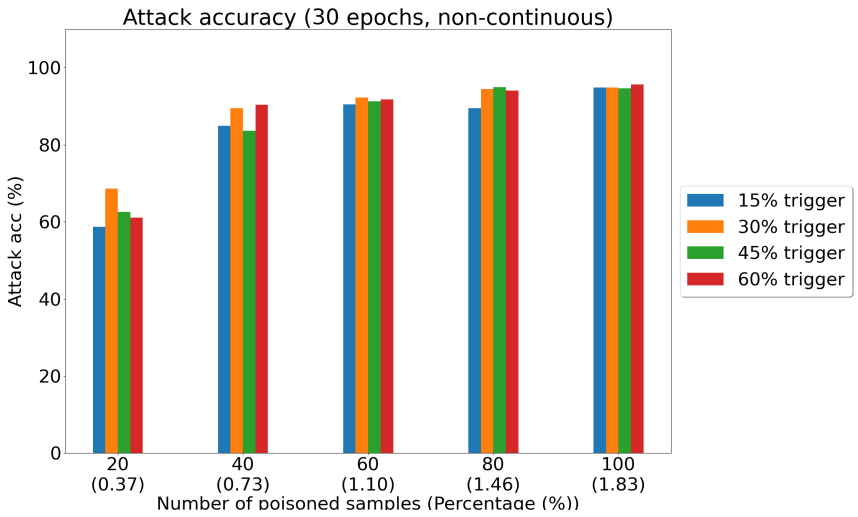


Figure A.14: Attack accuracy for non-continuous triggers in sentiment analysis, 20 training epochs, and the second architecture(Table 4.7). No connection between trigger size and attack accuracy is shown.

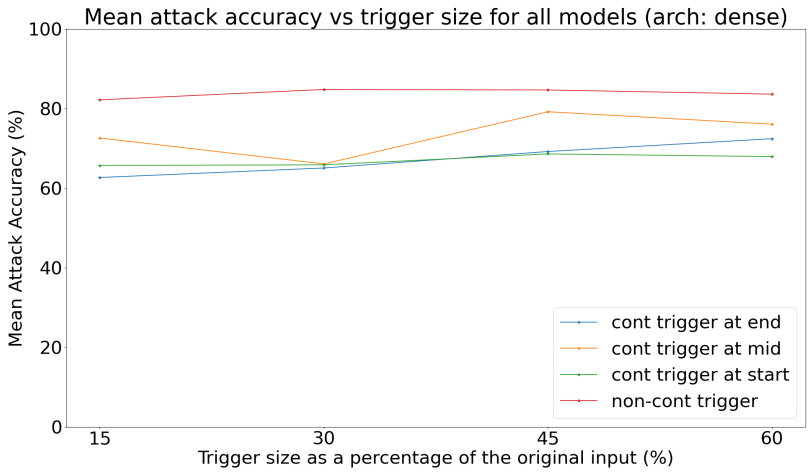
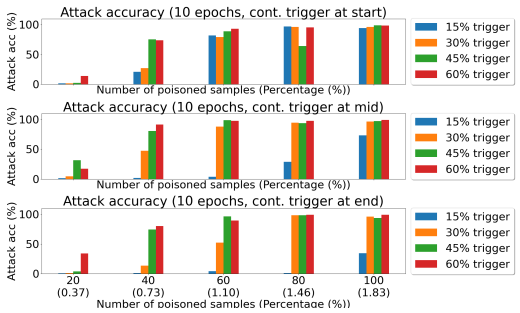
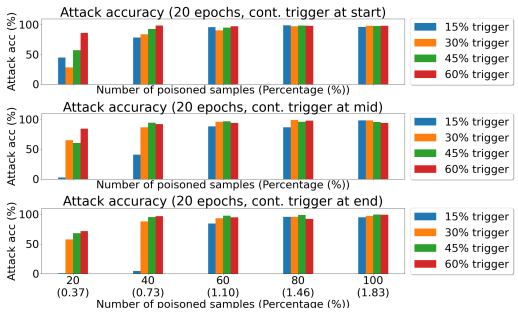


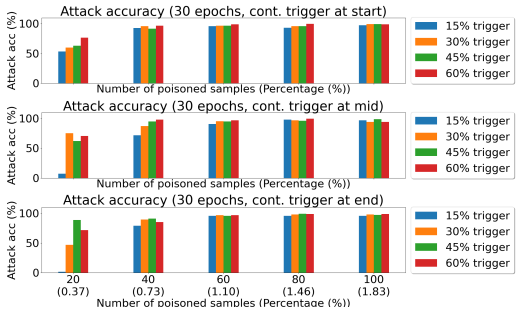
Figure A.15: Mean attack accuracy for sound recognition and the second architecture (Table 4.7). The non-continuous trigger is the best.



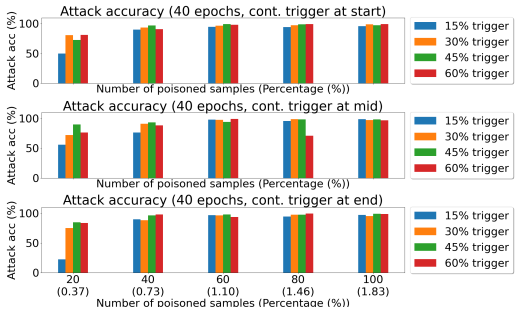
(a) Attack's accuracy on sentiment analysis for 10 epochs of training



(b) Attack's accuracy on sentiment analysis for 20 epochs of training



(c) Attack's accuracy on sentiment analysis for 30 epochs of training



(d) Attack's accuracy on sentiment analysis for 40 epochs of training

Figure A.16: Attack accuracy in speech recognition for continuous triggers for the third architecture (Table 4.8). In some cases, the attack accuracy is increased when the trigger size is increased from 0.15 to 0.3 seconds

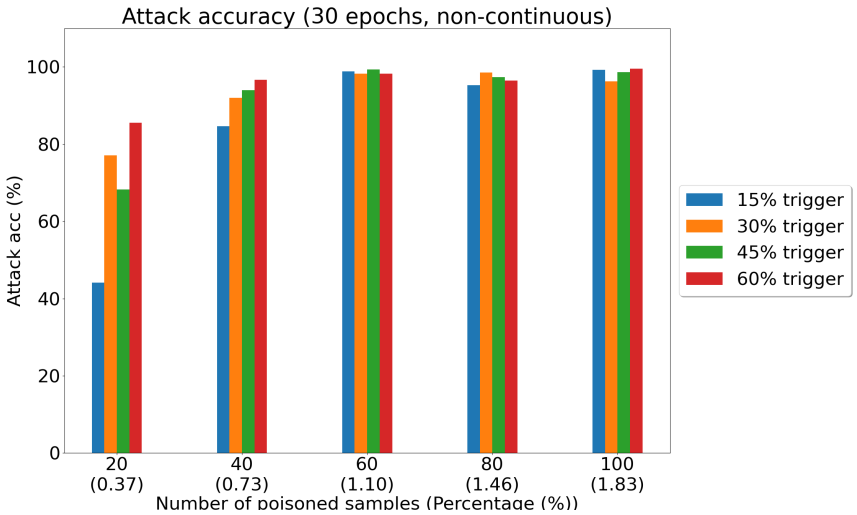


Figure A.17: Attack accuracy for non-continuous triggers in sound recognition, 20 training epochs and the third architecture (Table 4.8). There is a weak connection between the trigger size and the attack accuracy.

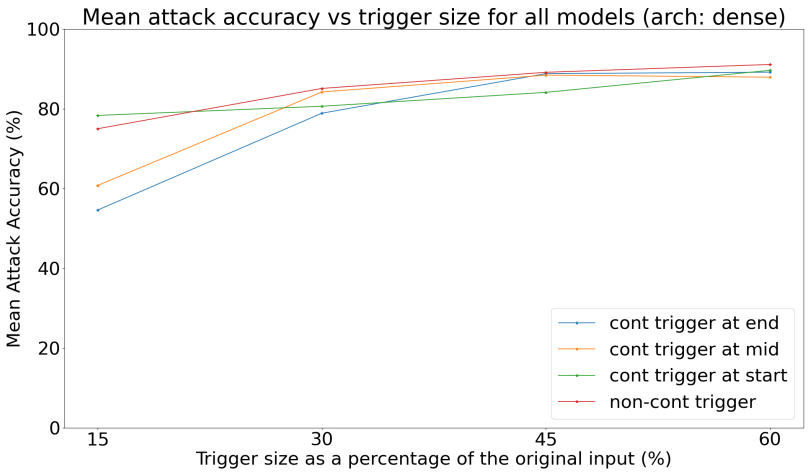
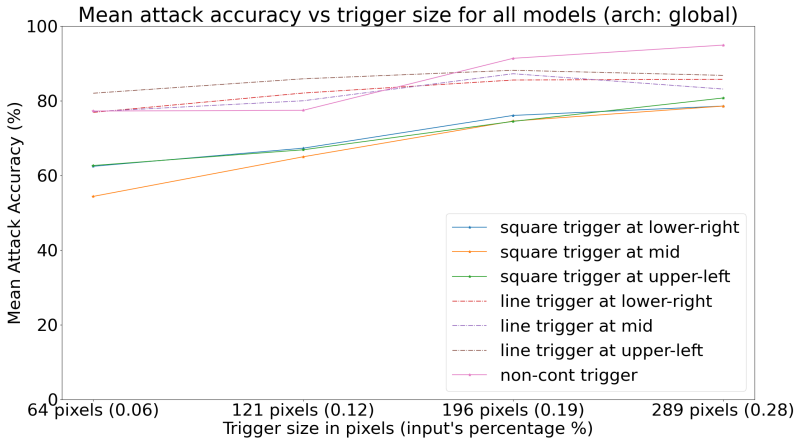


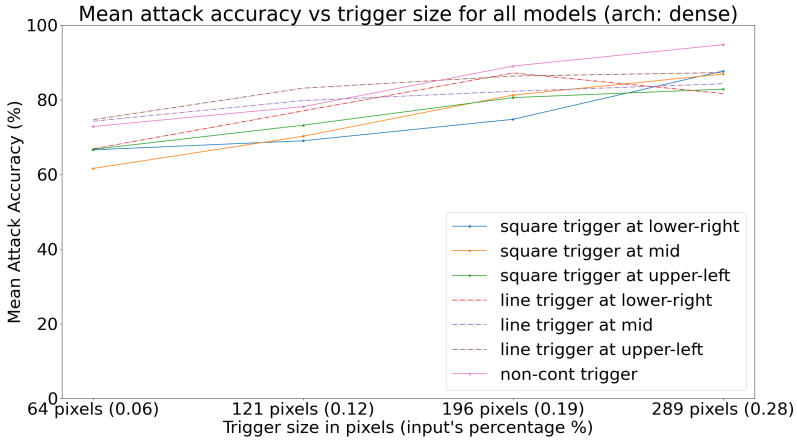
Figure A.18: Mean attack accuracy for sound recognition for the third architecture (Table 4.8). This is a deep architecture, and not many differences are observed, especially for 0.3-second triggers or more.

B

Additional Experiments for Chapter 5

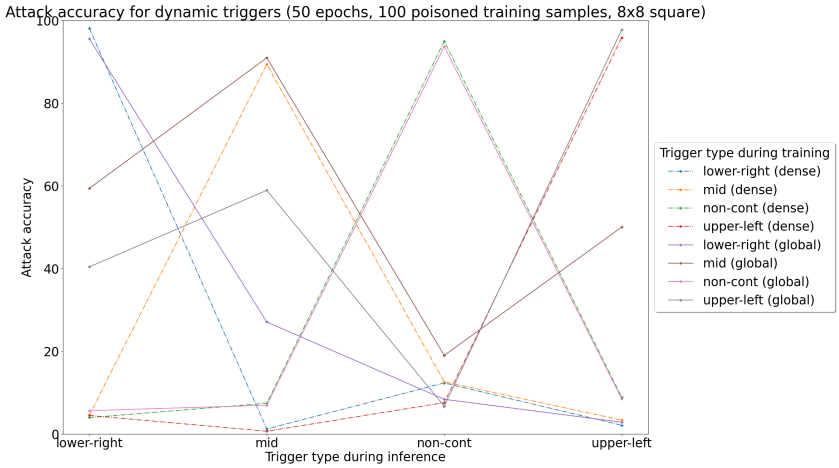


(a) Models with global average pooling

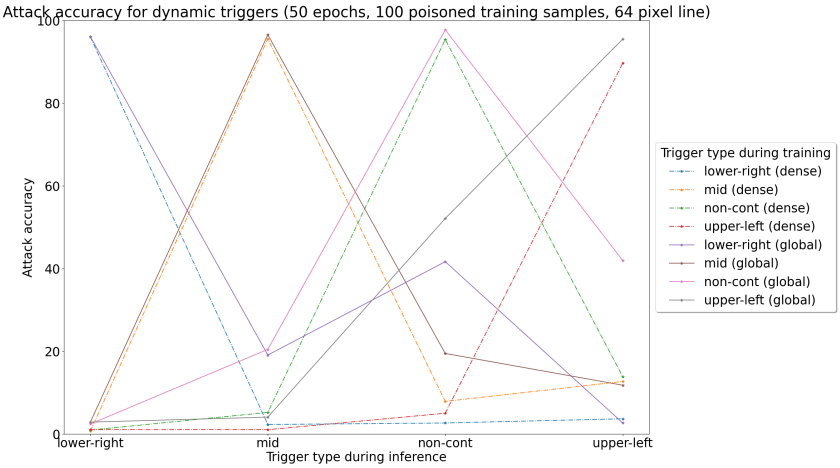


(b) Models without global average pooling

Figure B.1: Mean attack accuracy for different triggers in image classification for the first architecture (Table 4.2). Global average pooling increases the attack accuracy for small line triggers (64 and 121 pixels) and slightly reduces the maximum attack accuracy for the best performing trigger.

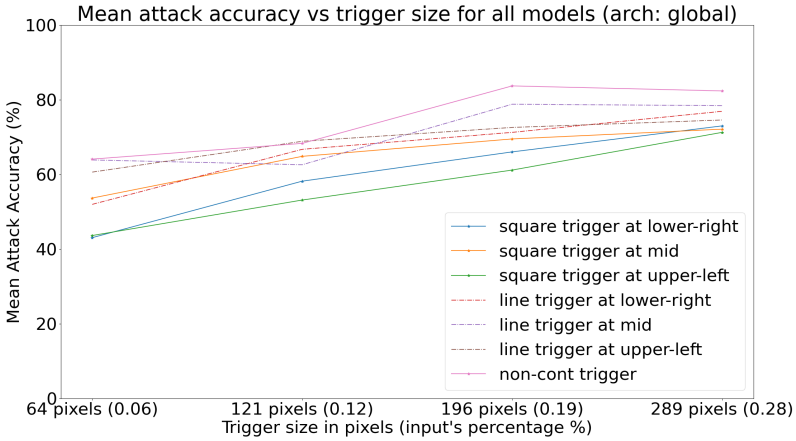


(a) Square triggers

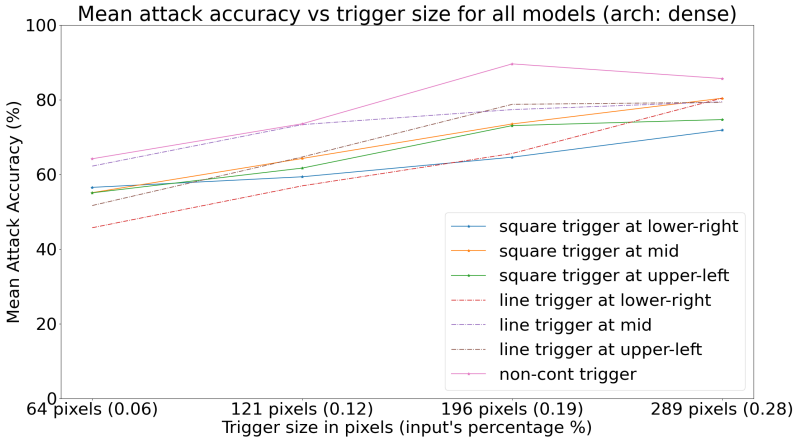


(b) Line triggers

Figure B.2: Attack accuracy for triggers in different positions between training and inference in image recognition for the second architecture (Table 4.2). Both line and square triggers of 64 pixels were tried for 50 epochs of training and 100 poisoned training samples.

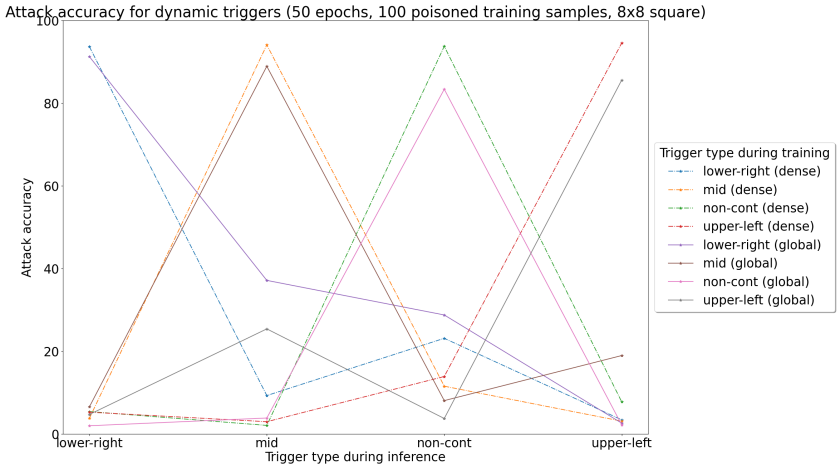


(a) Models with global average pooling

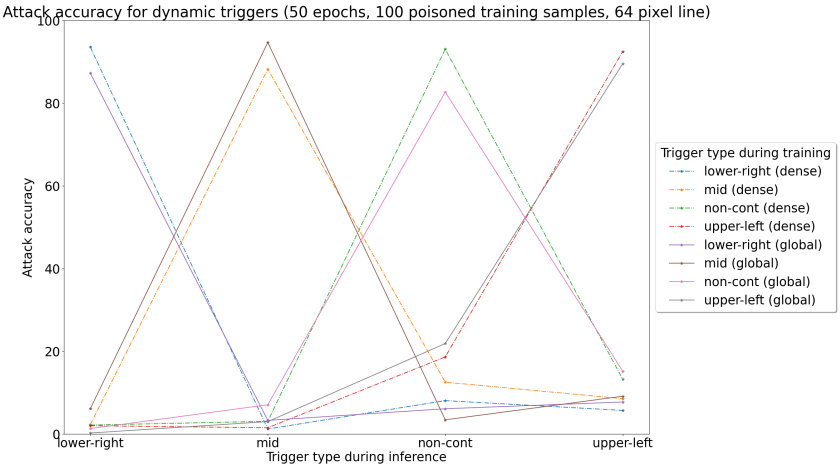


(b) Models without global average pooling

Figure B.3: Mean attack accuracy for different triggers in image classification for ResNet18. Global average pooling increases the attack accuracy for small line triggers (64 and 121 pixels) and slightly reduces the maximum attack accuracy for the best performing trigger

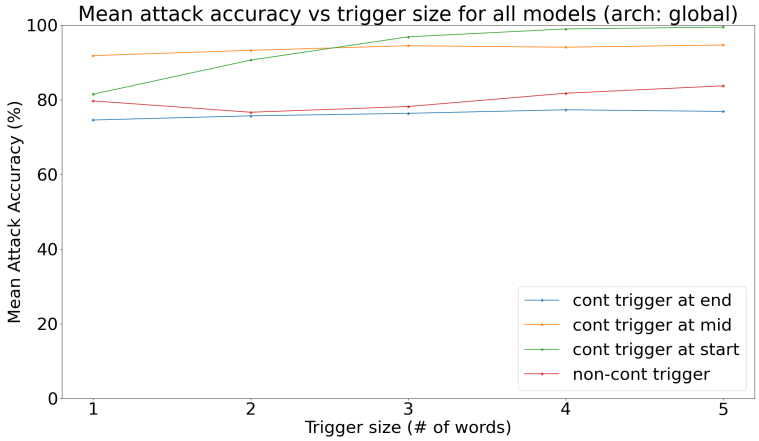


(a) Square triggers

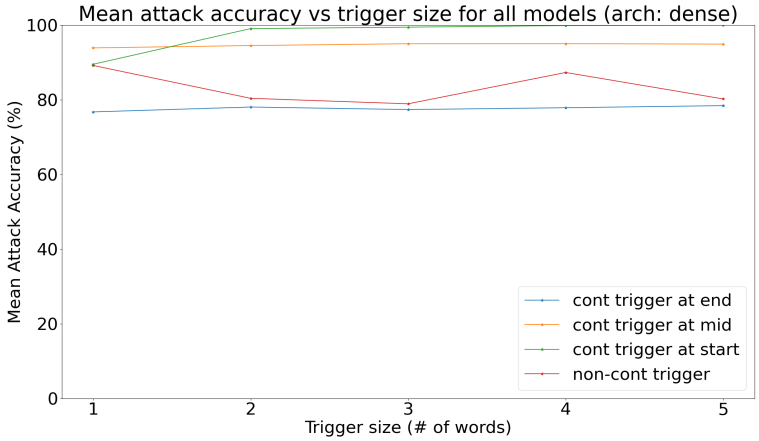


(b) Line triggers

Figure B.4: Attack accuracy for triggers in different positions between training and inference in image recognition for the ResNet18 [50]. Both line and square triggers of 64 pixels were tried for 50 epochs of training and 100 poisoned training samples.



(a) Models with global average pooling



(b) Models without global average pooling

Figure B.5: Mean attack accuracy for different triggers in text sentiment analysis for the second architecture (Table 4.4). There are no significant changes after global average pooling is inserted.

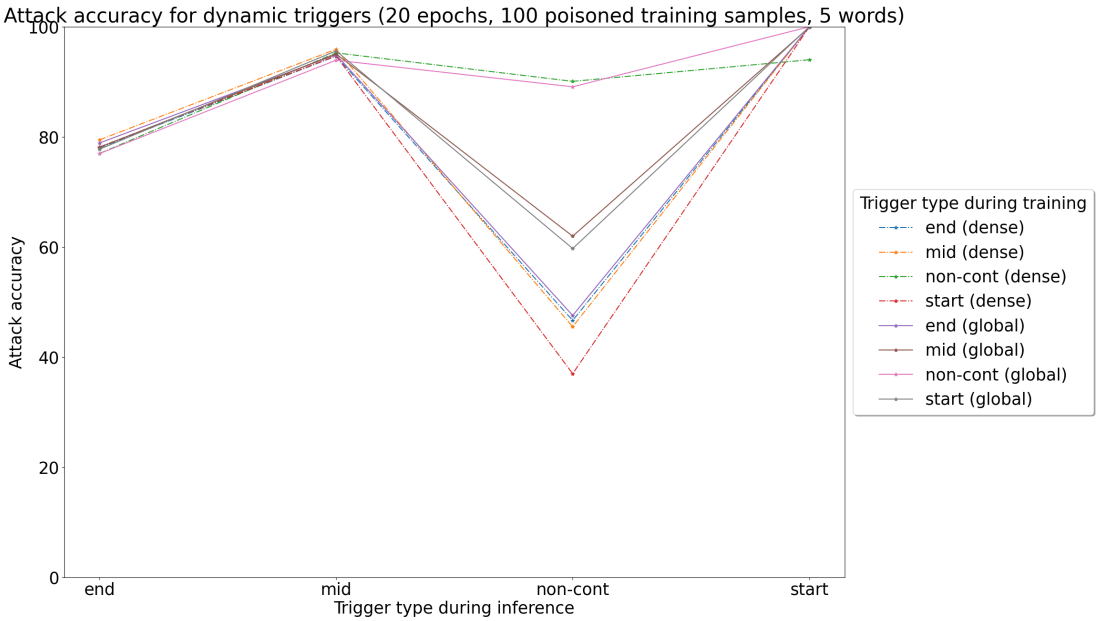
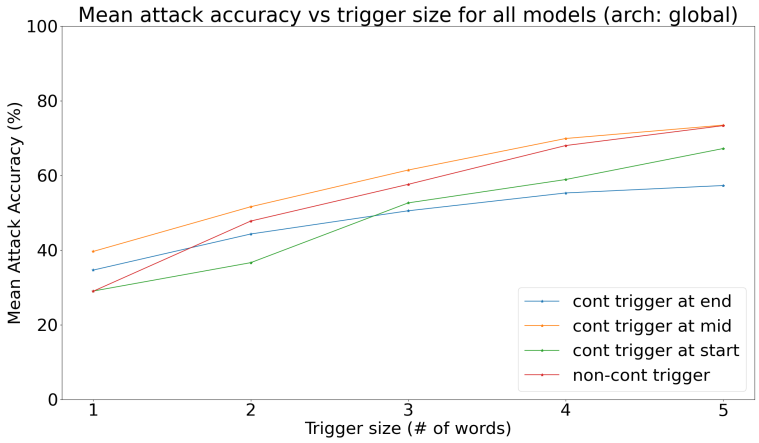
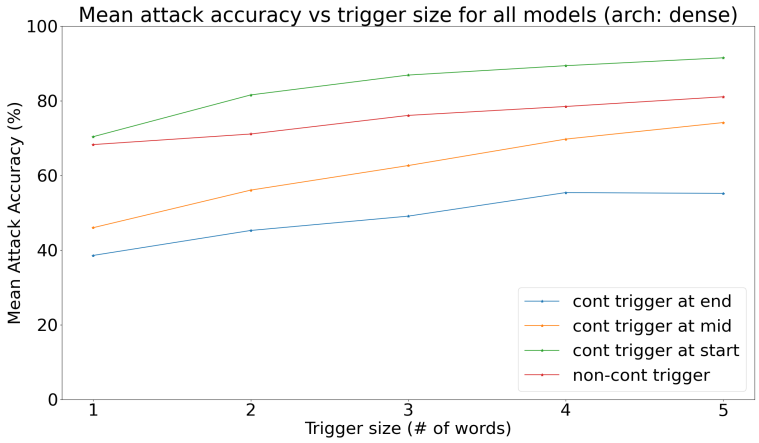


Figure B.6: Attack accuracy for triggers in different positions between training and inference in text sentiment analysis for the second architecture (Table 4.8). The models were trained for 20 epochs, 100 training samples were poisoned and 5 word triggers were used.



(a) Models with global average pooling



(b) Models without global average pooling

Figure B.7: Mean attack accuracy for different triggers in text sentiment analysis for the third architecture (Table 4.5). After global average pooling is inserted, the attack is more flexible because we see similar performance for all the positions. However, its accuracy is worse due to the generalization introduced.

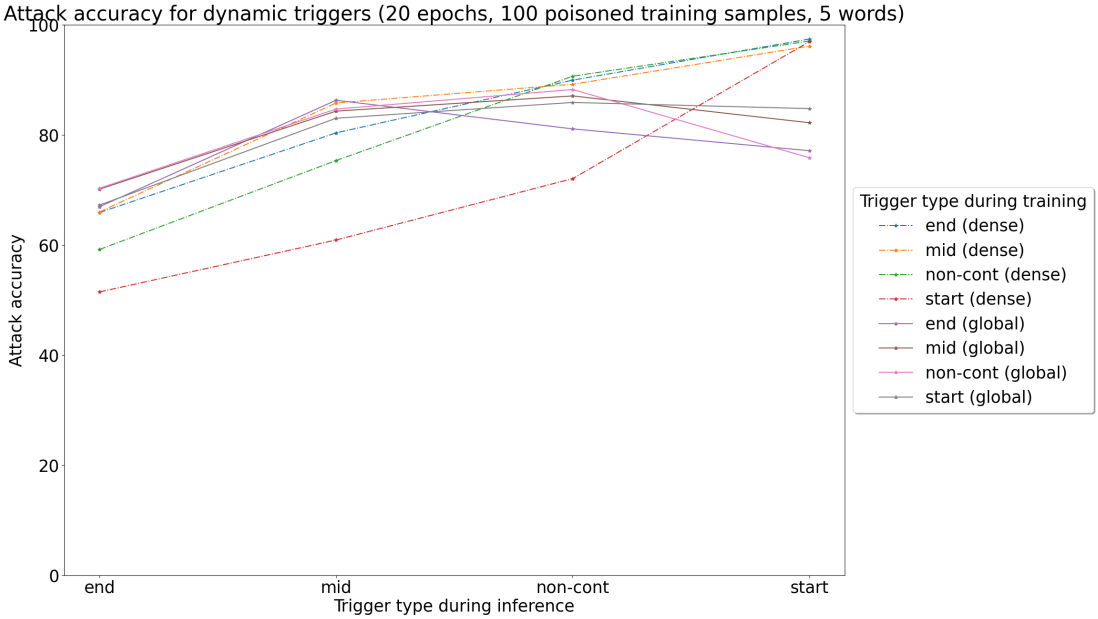
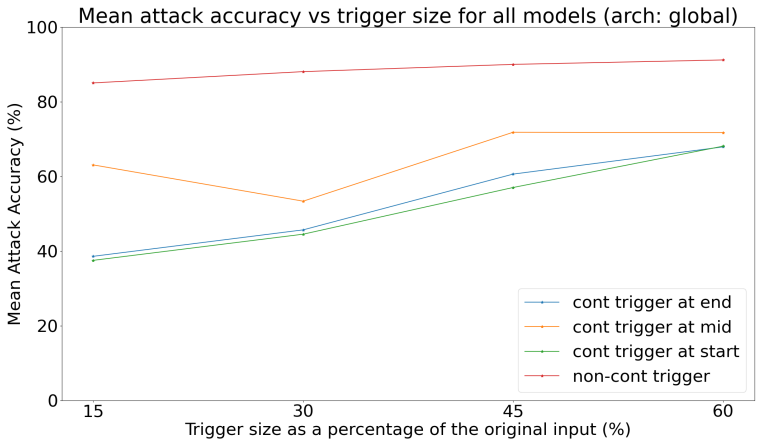
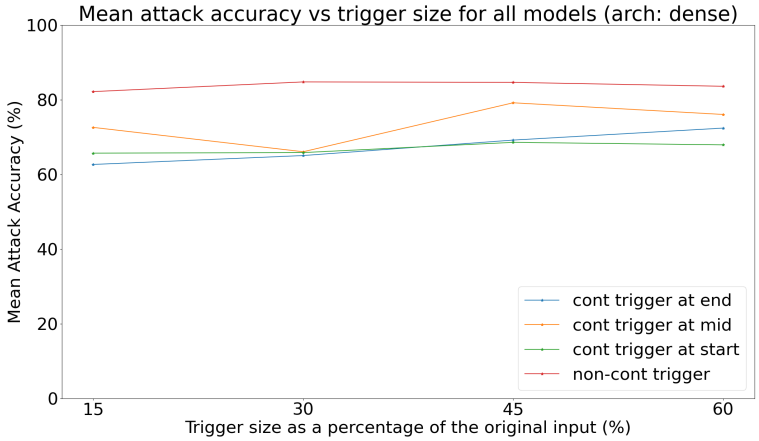


Figure B.8: Attack accuracy for triggers in different positions between training and inference in text sentiment analysis for the third architecture (Table 4.5). The models were trained for 20 epochs, 100 training samples were poisoned and 5 word triggers were used.



(a) Models with global average pooling



(b) Models without global average pooling

Figure B.9: Mean attack accuracy for different triggers in sound classification for the second architecture (Table 4.7).

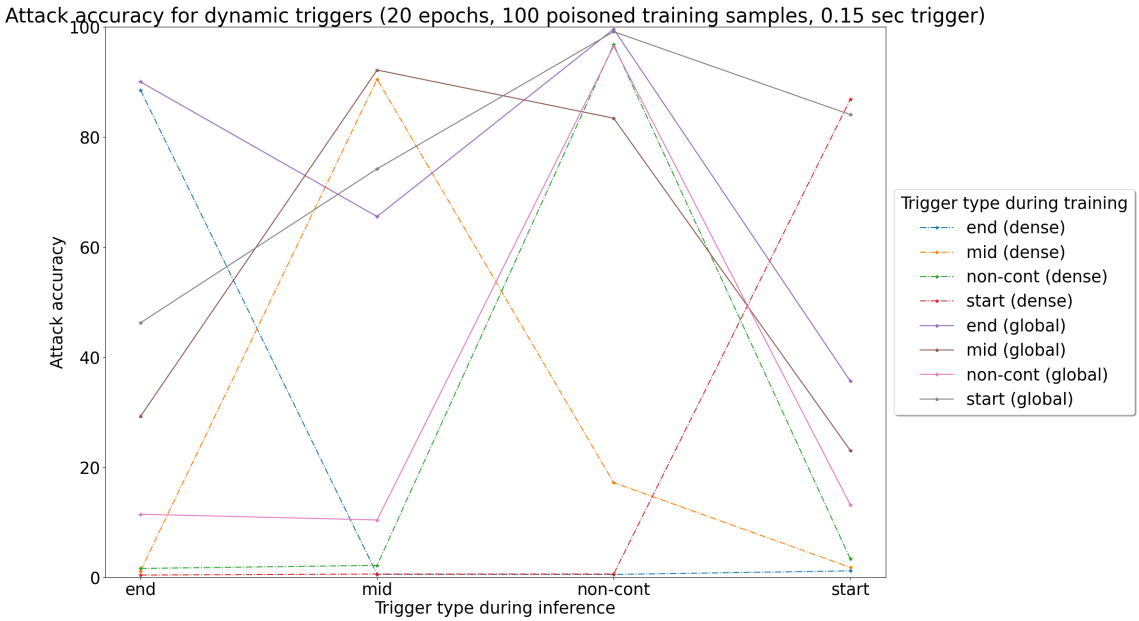
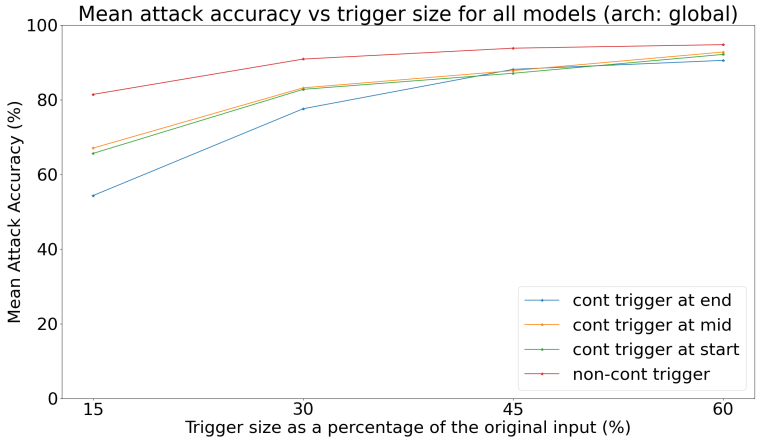
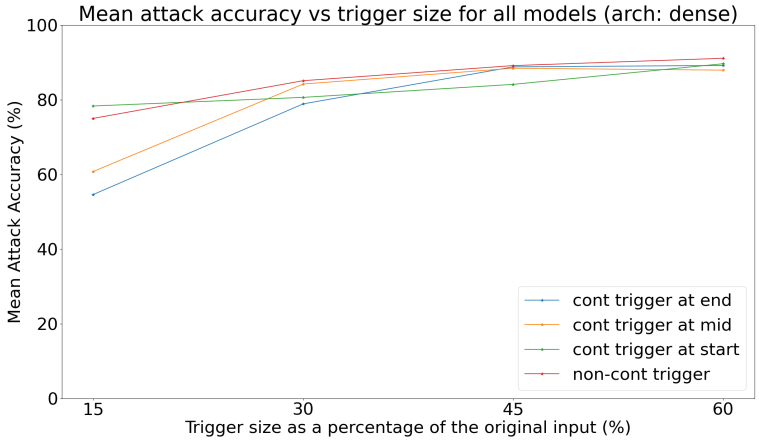


Figure B.10: Attack accuracy for triggers in different positions between training and inference in sound recognition for the second architecture (Table 4.7). The models were trained for 40 epochs, 100 training samples were poisoned and 0.15 second triggers were used.



(a) Models with global average pooling



(b) Models without global average pooling

Figure B.11: Mean attack accuracy for different triggers in sound classification for the third architecture (Table 4.8). Only minor differences are introduced when global average pooling is inserted.

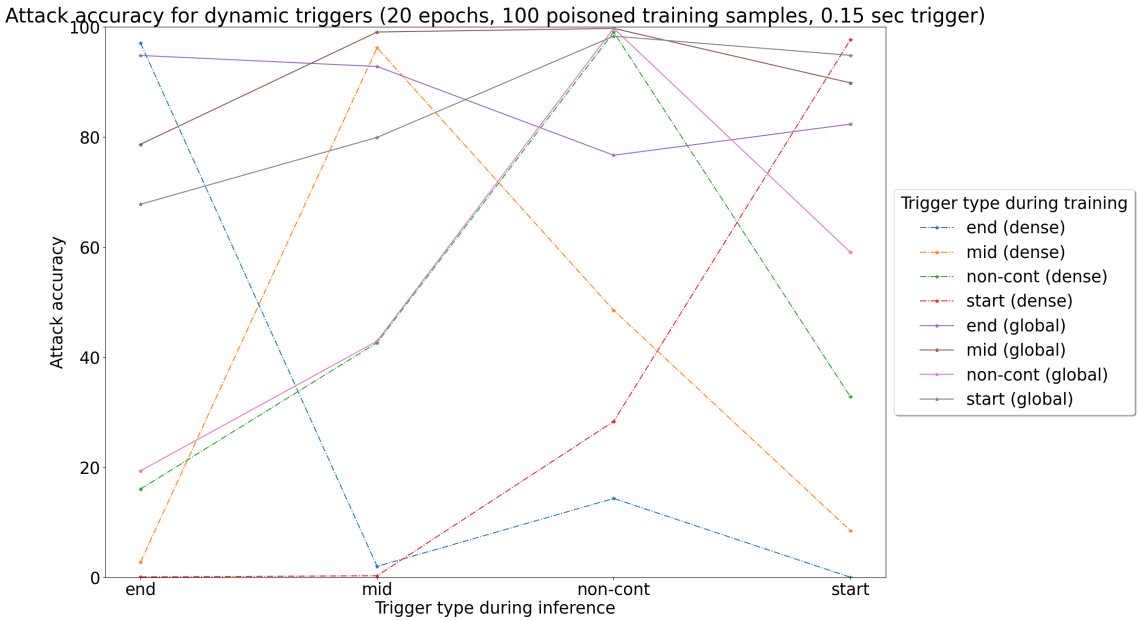


Figure B.12: Attack accuracy for triggers in different positions between training and inference in sound recognition for the third architecture (Table 4.8). The models were trained for 40 epochs, 100 training samples were poisoned and 0.15 second triggers were used.