# Cloud-Based Machine Learning Models as Covert Communication Channels

Torsten Krauß
University of Würzburg
Würzburg, Germany
torsten.krauss@uni-wuerzburg.de

Jasper Stang
University of Würzburg
Würzburg, Germany
jasper.stang@uni-wuerzburg.de

Alexandra Dmitrienko
University of Würzburg
Würzburg, Germany
alexandra.dmitrienko@uni-wuerzburg.de

## ABSTRACT

While Machine Learning (ML) is one of the most promising technologies in our era, it is prone to a variety of attacks. One of them is covert channels, that enable two parties to stealthily transmit information through carriers intended for different purposes. Existing works only explore covert channels for federated ML. Thereby, communication is established among multiple entities that collaborate to train a model, while relying on access to model internals.

This paper presents covert channels within ML models trained and publicly deployed in cloud-based (black-box) environments. The approach relies on targeted poisoning, or backdoor, attacks to encode messages into the model. It incorporates multiple well-chosen backdoors only through dataset poisoning and without requiring access to model internals or the training process. After model deployment, messages can be extracted via inference.

We propose three covert channel versions with varying levels of message robustness and capacity while emphasizing minimal extraction effort, minimal pre-shared knowledge, or maximum message stealthiness. We investigate influencing factors affecting embedded backdoors and propose novel techniques to incorporate numerous backdoors simultaneously for message encoding. Experiments across various datasets and model architectures demonstrate message transmission of 20 to 66 bits with minimal error rates.

## CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → *Machine learning*;

## KEYWORDS

covert channel; machine learning; poisoning attacks; backdoors

**ACM Reference Format:**
Torsten Krauß, Jasper Stang, and Alexandra Dmitrienko. 2024. Cloud-Based Machine Learning Models as Covert Communication Channels. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '24), July 1–5, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3634737.3657026


*ASIA CCS '24, July 1–5, 2024, Singapore, Singapore*

ACM ISBN 979-8-4007-0482-6/24/07…$15.00
https://doi.org/10.1145/3634737.3657026

## 1 INTRODUCTION

A recently emerged technology is Machine Learning (ML), a fast-growing field that has significantly impacted various areas, e.g, speech recognition [8], object detection [17, 26, 30], and predictive analysis [2]. Such models trained by an ML process designed to streamline and automate various tasks are increasingly deployed in critical applications such as healthcare [33] and national security [40, 43]. However, it has also become a target for adversaries performing a variety of attacks. In one of the malicious attacks on ML models, the adversary can misuse the model to embed a hidden message that is concealed within the model. Those messages can then be extracted by a third party, essentially forming a covert communication channel misusing the ML model as a message carrier.

A covert channel [38] is a communication pathway designed to operate stealthily, often with the intent of circumventing conventional security measures and remaining undetected. These channels facilitate the concealed exchange of information or data, posing challenges for security mechanisms in detecting their existence or monitoring the transmitted data. The first historical covert channel dates back to Histiaeus, a tyrannical leader who needed to secretly convey a strategy to his nephew. He shaved the head of a messenger, inscribed the message on the man's scalp, and waited for the hair to regrow, concealing the secret. Upon the messenger's arrival, the nephew shaved the messenger's head, thus revealing the hidden plans [15]. A modern example is the TV series "The Americans", which is based on real-life spy stories, in which spies embed encrypted data within images stored on public websites, making the data imperceptible to a human observer [9]. Covert channels in the information technology domain take on diverse forms, such as covert data transmission within seemingly innocuous network protocols [38], obscure encodings integrated into legitimate files [11], or the exploitation of hidden communication within a computer system's architecture [42]. Thereby, the channels focus on contemporary and innovative techniques and serve various purposes centered around hidden communication between malicious actors.

**Problem Statement.** Covert channels have evolved over time, adapting to the prevailing media and communication methods of their respective eras. Detecting these covert channels is crucial to be aware of the potential of malicious activities facilitated by emerging technologies. Further, it is the first step towards mitigation of covert channels, which is pivotal for maintaining security and reliability of computer systems and networks. Security professionals and researchers employ various techniques and strategies to unveil and prevent the illicit use of covert channels, ensuring the protection of sensitive information and the identification and prevention of unauthorized activities. As one emerging technology these days

is ML, a pivotal question that arises is the potential misuse of ML models as covert channels for hidden message transmission between two remotely situated, collaborating entities unable to communicate directly. However, covert channels leveraging ML models as message carriers have not been thoroughly explored.

**Existing Works.** Prior research that delves into covert channels using ML models [4, 7, 18, 23] exclusively centers around Federated Learning (FL) [32], a framework in which multiple clients collaborate in successive rounds to collectively train a model using their local datasets, managed by a central server. In this context, the covert message exchange takes place between two or more distinct clients. The majority of these approaches require access to the model's weights, which can be directly manipulated to conceal and extract the message [4, 18, 23]. Besides, Costa *et al.* [7] use a specifically poisoned dataset to tweak the model's predictions for some specific pre-shared samples, which can be used to extract one bit for each sample only with inference access to the model.

**Approach.** This paper is the first work that investigates the feasibility of establishing an efficient covert channel between two entities leveraging a centralized cloud-based ML environment, namely a Machine Learning as a Service (MLaaS) provider. We seek to utilize the trained ML model as a carrier for covert communication. For message embedding, contrary to the research conducted in FL, we restrict ourselves to manipulations of the training data. The model internals and the training process remain unaltered as they are executed by the MLaaS provider. By submitting a manipulated training dataset that includes an encoded message to the MLaaS platform, the sender of the covert message incurs incorporation of the message along with the regular task into the model. To realize the message embedding within the training data, we leverage targeted poisoning attacks [13, 27], so-called backdoors. These backdoors introduce concealed, attacker-defined behaviors, typically manifesting as erroneous predictions, that are activated through the inclusion of a pre-defined trigger in the model's input. After the public deployment of the trained model, a receiver with the knowledge of the pre-defined backdoor triggers should be able to extract the hidden message through inference on the deployed model. This scenario allows the realization of a covert communication channel in the domain of ML akin to the concept of an image containing a hidden message stored on a public website, as exemplified earlier.

**Contributions.** In particular, this paper contributes the following:

- We are the first to explore the problem of covert channels in MLaaS settings, which disallow access to model internals or the training process and solely rely on dataset manipulations and inference for message embedding and extraction.
- We develop three covert channels with distinct message encoding techniques centered around the embedding of backdoors. The techniques use different encoding rules based on varying backdoor trigger locations and appearances and offer a high degree of flexibility to both senders and receivers of covert messages, enabling them to choose the specific instantiation that aligns with the application's requirements, encompassing the size of pre-shared knowledge, inference complexity for extraction, and the channel's stealthiness within the model's regular task.
- We propose novel and robust dataset poisoning strategies for backdoor injection that allow the simultaneous embedding of multiple backdoors into a single model. The strategy carefully selects suitable samples and systemically poisons them to facilitate the model's ability to discern the subtle differences in the trigger's appearance.
- We introduce a covert channel, that uses ineffective backdoors, a novel concept, that implants information into the model by embedding backdoors with only poisoning a negligible dataset portion. As a result, the small changes in the model caused by the poisoned samples do not change the benign predictions but can be used to transmit information.
- Throughout this work, we are the first to explore various influence factors that affect the effectiveness of injected backdoors when systematically embedding multiple backdoors within a single model in a structured manner. This includes the trigger type, appearance attributes (such as shape and color), and the location within the sample. The results of our study are of independent interest and could be leveraged in other areas of research, such as backdoor attacks and model watermarking.
- We conduct a systematic large-scale study leveraging different datasets (CIFAR-10 [25], STL-10[6]) and model architectures (ResNet-18 [17], SqueezeNet [20]) validating the effectiveness of the three proposed covert channels. This study demonstrates the channels' capacity with remarkably low error rates, enabling robust transmission of long genuine messages ranging from 20 to 66 bits.

## 2 BACKGROUND

**Backdoor Attacks.** Poisoning attacks empower adversaries to manipulate model predictions. These attacks are either untargeted [31, 36] or targeted [13, 27]. Untargeted attacks aim to reduce model performance, while targeted attacks, known as backdoors, introduce hidden behavior while preserving main utility of the model.

A backdoor comprises two components: A trigger and an attacker-chosen target label, called target. The former is injected into a training sample provoking the malicious prediction, while the latter specifies the desired misprediction for triggered samples. Triggers can take various types, including mask triggers which are subtle noise overlays [5], visible pixel patterns [13], or the like.

Adversaries can embed backdoors through data poisoning, which involves manipulating samples from the training dataset by introducing triggers and changing the assigned labels towards the backdoors target label. The poison data rate (PDR) determines the proportion of data samples within the training dataset that get contaminated with backdoors during dataset poisoning. Alternatively, the adversary can directly modify the training process by adjusting hyperparameters or excluding certain parameters from training, a technique referred to as model poisoning.

Clean-label backdoors [39] are a unique subset of targeted poisoning attacks, where the labels associated with triggered samples remain unaltered instead of mislabeling the sample to the attacker-chosen target label. The model establishes a connection between the trigger and the sample's original (clean) label, which is the backdoor target class. During inference, a sample from a different

class, armed with the trigger, is misclassified into the backdoor's target class. Clean-label attacks exhibit the advantage of having limited adverse effects on the model's performance because the poisoned samples are still labeled with the ground-truth class.

The backdoor's impact on the model's main task can be evaluated by comparing the model's prediction performance on the training or test set, the so-called accuracy, with that of an unpoisoned model, which also gives insight into the backdoor's stealthiness. The attack success rate (ASR) quantifies the extent to which the backdoor targets are successfully predicted for a set of triggered samples.

In this paper, we leverage clean-label backdoors to establish a covert communication channel within a model.

**Covert Channels.** A covert channel [38], in the context of information security, serves as a concealed communication path, designed to operate discreetly with the aim of evading security measures and remaining undetected. These channels enable parties (authorized or unauthorized) to exchange information or data covertly, making it challenging for security mechanisms or external observers to monitor or at least identify the presence of transmitted data. Covert channels can take on various forms, e.g., transmitting hidden data through seemingly benign network protocols [38] or exploiting hidden channels within a computer system's architecture [42].

Covert channels are utilized for multiple objectives all centered around secret communication among malicious entities. Consequently, detecting and mitigating covert channels is of paramount importance for safeguarding the security of and trust in the reliability of computer systems and networks.

In assessing channel performance, one metric is the bit error rate (BER) [37], which quantifies the ratio of incorrect bits to received bits. However, when the length of the received message differs from the original message, calculating the BER becomes impractical. Then, the Levenshtein distance (LD) [46] can measure the dissimilarity between two strings of varying lengths by quantifying the minimum number of edit operations (insertions, deletions, or replacements) necessary to transform one string into the other. For two identical bit sequences, LD is zero; conversely, differing sequences entail operations to align them. By dividing the LD by the maximum sequences length, the Levenshtein error rate (LER), can be computed, offering a meaningful metric for covert channel performance. A LER of 0.0 indicates errorless transmission, a LER of 1.0 that all bits were incorrect.

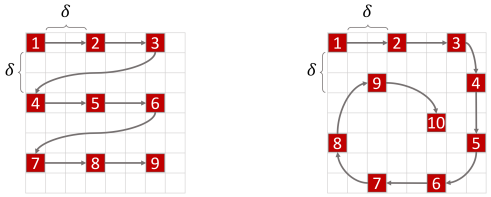
In this paper, we establish a covert channel within an ML model and evaluate the performance using BER and LER.

## 3 COVERT CHANNEL CONCEPT

In this section, we describe the core concept of our approach. We begin with presenting our system model and objectives in Sect. 3.1, followed by the description of general approach in Sect. 3.2.

### 3.1 System Model

Our system model involves two main actors: a sender and a receiver of the covert message. They aim to (mis)use a machine learning (ML) model that has been trained and hosted in a cloud-based environment. The collaborating entities share a finite amount of pre-shared knowledge. Subsequently, they are physically separated, with no possibility for direct communication, as any traceable or visible interaction is to be avoided.[1] The sender has the goal of sending a hidden message to the receiver. Therefore, the sender utilizes an ML model trained and deployed in a cloud-based environment, such as a Machine Learning as a Service (MLaaS) provider. Thereby, the cloud provider is considered a trustworthy black box service, meaning that direct access to the model's weights is not permitted and the requested ML training is reliably executed.[2] In this scenario, the sender provides a dataset on which the model is trained and the receiver performs inference on the trained model. Consequently, the concealed message must be exclusively integrated within the training dataset provided by the sender, and it should be retrievable through model inference by the receiver. As the dataset is provided by the sender, it can be manipulated arbitrarily to embed the message before upload to the MLaaS provider.




(a) Traverse (TP) (b) Spiral (SP)

**Figure 1: Location patterns with $\delta = 2$ for a $8 \times 8$ image.**

**Objectives.** The selection of message encoding and dataset manipulation techniques is pivotal, as they must align with the requirements of the covert channel to ensure the success of this new communication method allowing message extraction by only leveraging the pre-shared knowledge. Critical objectives are high transmission capacity, as well as high fidelity, meaning that the model's regular task is not significantly impacted. Further, the message extraction should be reliable achieving low message error rates. Moreover, the approach should be generalizable allowing for covert channel instantiation in different application scenarios.

### 3.2 General Approach

In a nutshell, our method utilizes targeted poisoning attacks, so-called backdoors (cf. Sect. 2), which are discreetly embedded into the DNNs, for establishing the covert channel. Based on pre-shared knowledge that contains the structural characteristics of these embedded backdoors, we embed a series of carefully designed backdoors into the DNN. These backdoors are crafted to encapsulate the intended message we aim to convey.[3] By leveraging the pre-shared knowledge, the receiver can analyze the presence or absence of these backdoors, essentially extracting the encoded message. A major challenge is how to embed a maximum amount of effective

[1] E.g., the receiver can be located in another country. The communication should be hidden from the infrastructure-owning party, e.g., an internet service provider.

[2] In any covert channel, detection is possible if the transmission medium owner, e.g., the MLaaS provider, is aware of the transmission and has an incentive to intervene, e.g., conducting message erasure, alteration, or insertion. However, integrity measures like HMAC [24], implemented by the sender, can mitigate these risks. Nevertheless, such a malicious entity is out-of-scope of this work.

[3] The backdoor's stealthiness influences the channel's stealthiness. An MLaaS provider that is aware of the covert channel could use dataset-cleaning methods to detect and manipulate the message. However, such an entity is not considered in this work.

and distinguishable backdoors within the model to fulfill the transmission capacity objective of the covert channel. This is achieved by varying backdoor trigger locations, and a specifically crafted dataset poisoning methods.

Our general approach encompasses two distinct phases: (i) Establishing a pre-shared knowledge, and (ii) covert communication. In the first phase, the sender and receiver exchange pre-shared knowledge functioning as keys for the covert channel. In a second phase, the channel is established with the help of the pre-shared keys.

**Pre-Shared Knowledge.** As our covert channel design hinges on backdoors, it is imperative to pre-define the backdoor strategy and its specifications before initiating communication. The pre-shared knowledge is composed of two parts. First, the backdoor specification comprising of the trigger type, appearance, and location, and, second, a mapping between backdoor targets and bits.

*Backdoor Trigger Type.* Our covert channel is agnostic to the trigger method, which we refer to as "type". As an illustrative example, we predominantly focus on monochromatic pixel triggers [13]. The sample sketches containing a bird in App. Fig. 10 visualize examples of such triggers that inject a red square into an image. In addition, we discuss mask triggers [5, 41] in App. C.

*Backdoor Trigger Appearance.* Pixel triggers allow a variety of possible shapes, such as squares, triangles, and crosses. We also categorize their color as a defining aspect of their appearance.

*Backdoor Trigger Location.* If multiple backdoors are embedded into a model, we expect the precise location of the trigger's pixels within the sample to be highly important, as otherwise overlapping triggers might not be distinguishable. Hence, it is essential to establish a spatial sequence of distinct trigger locations in advance. We define the "traverse pattern" (TP) and the "spiral pattern" (SP), which we depict in Fig. 1. TP traverses from the top-left corner to the bottom-right corner, while maintaining a spacing of $\delta$ between adjacent triggers. SP follows a spiral movement starting from the top left corner continuing clockwise towards the center of the image. Defining such a pattern reduces the amount of pre-shared knowledge compared to sharing the complete set of points directly.

*Backdoor Target Mapping.* In our approach, we place particular emphasis on clean-label backdoor attacks [39], wherein the labels associated with the tainted data remain unchanged, and the samples are not randomly generated. Consequently, the DNN establishes a connection between the trigger and the corresponding clean label, which is the original label of the (triggered) training sample. Subsequently, during inference, a sample from a different label class that is equipped with the trigger is misclassified into the backdoor's target, which is the clean label class. One noteworthy advantage of employing clean-label attacks lies in their limited adverse impact on the model's performance as the training data are not mislabeled.

**Covert Communication** Once the sender and receiver establish pre-shared knowledge, they can establish covert communication channel. We depict the process in Fig. 2, which comprises encoding and decoding phases and includes the following steps:

(1) **Encoding.** The sender converts the message into an encoded representation that adheres to the covert channel's protocol. This encompasses the message encoding process where the message is transformed into a binary sequence.
(2) **Dataset Poisoning.** Next, the encoded message is integrated into the training dataset, such that both, the dataset's regular task and the hidden message are comprised. In particular, the bit sequence is mapped to various backdoors, aligning with the backdoor strategy outlined in the pre-shared knowledge. This mapping results in a set of triggers and corresponding targets forming trigger-target pairs that encode the message. We depict them as backdoor definitions in Fig. 2. During dataset poisoning, as many backdoors as possible, adhering to the sender-defined PDR are embedded.
(3) **Dataset Submission.** The poisoned dataset, along with hyperparameters, is sent to the MLaaS provider for training.
(4) **Model Training.** After submission to an MLaaS provider, the modified dataset is used to train an ML model, preserving the encoded message within.
(5) **Sample Generation.** Next, the message extraction phase begins. Here, the receiver initially creates a corresponding inference sample using one part of the pre-shared knowledge. In particular, an evaluation sample is sequentially equipped with the different triggers as specified by the backdoor strategy. Once a trigger is present in the input sample, the corresponding backdoor can be activated.
(6) **Inference Loop.** Upon using this sample for inference, the prediction result containing the hidden encoded message from the covert channel can be obtained. In particular, the triggered sample utilized for inference will generate the respective sender-chosen prediction. This prediction contains the target associated with the embedded backdoor. Depending on the target labels in the backdoor strategy and the prediction result, it can be inferred whether the message is complete or not. If incomplete, a new subsequent triggered sample is generated, and the inference process of step 6 is repeated with a fresh trigger.
(7) **Decoding & Error Correction.** Afterward, the observed backdoor targets are translated into the transmitted binary bit sequence, adhering to the backdoor strategy defined in the pre-shared knowledge. Once the message is complete, error correction techniques can be applied to enhance transmission reliability. Using the second part of the pre-shared knowledge, the receiver decodes the message embedded in the prediction result to retrieve the original message.

After clarifying the fundamentals above, we present three specific instantiations in the next section, each employing distinct encoding and decoding strategies leveraging different pre-shared knowledge.

## 4 COVERT CHANNEL INSTANTIATIONS

Below, we present three covert channel variations, namely single-trigger backdoors (STB), multi-trigger backdoors (MTB), and light-trigger backdoors (LTB), that instantiate our general approach and are characterized by a unique way to define type, appearance, location patterns, and target labels that map bits to backdoors. Each method focuses on specific covert channel properties. While STB is the most straightforward approach with high robustness, the size of the pre-shared knowledge increases for classification tasks with more classes. Therefore, we design an alternative approach, MTB,

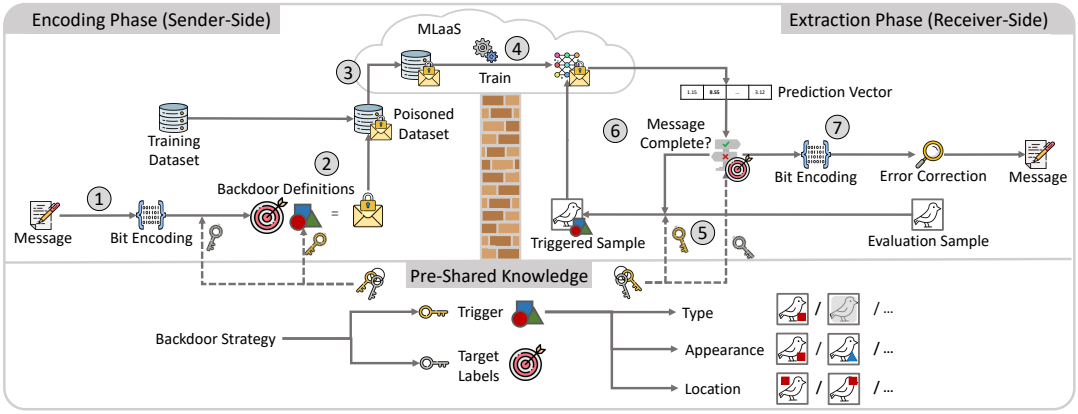



**Figure 2: Visualization of the encoding and decoding steps of our covert communication channel based on backdoors.**

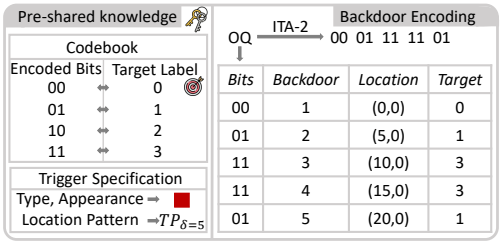



**Figure 3: Example of STB encoding "OQ" into five backdoors. The model's main classification task has four classes (0-3).**

which rely on a fixed amount of pre-shared knowledge. A drawback from both approaches is the negative impact on the model's performance on the main task, which is addressed by LTB. Hence, each approach has strengths and weaknesses and the sender can choose the concrete instantiation that fits best the specific use-case.

## 4.1 Single-Trigger

First, we consider an intuitive approach based on a bit-encoding codebook. The approach relies on one specific trigger appearance, hence we name it single-trigger backdoors (STB).

*Encoding.* For illustrative purposes, we opt for a monochromatic (red) square pixel as trigger appearance. The trigger specification includes the designated location pattern and an associated codebook, responsible for mapping backdoor targets to specific bits. They constitute integral components of the pre-shared knowledge. Note, that the size of the codebook is dependent on the number of classes within the classification task, where more classes could lead to an increased codebook size. This, however, has the advantage that a single target label can encode a longer bit sequence.

Fig. 3 depicts the concept of STB for an example with a 4-class classification task (target labels 0, 1, 2, and 3) and a text message consisting of the two characters "OQ". This message is encoded into a binary string using the ITA-2 encoding [21], as applied in our experimental setup and discussed in App. A. As the codebook accommodates bit-tuples, a single backdoor represents two bits. The backdoor location is systematically adjusted based on the TP location pattern. For instance, for the spacing $\delta = 5$ the coordinates for for the first and the second backdoor would be (0,0) and (5,0). The backdoor targets align with the codebook-defined bits.[4]

*Dataset Poisoning.* The backdoors are then embedded onto separate samples from within the train dataset via clean-label backdoor attacks [39] using data poisoning (cf. Sect. 2). For instance, all five backdoors in Fig. 3 are injected into different samples. Even if backdoor 2 and 5 with different trigger locations (5,0) and (20,0) have the same target label 1, the backdoors are not injected onto the same samples. The overall amount of poisoned samples results from the poison data rate (PDR) chosen by the message sender.

*Extraction & Decoding.* For decoding, the trigger defined in the pre-shared knowledge is injected into an arbitrary sample, which subsequently undergoes inference. By systematically adjusting the trigger's location in accordance with the pre-shared location pattern, the receiver can derive a sequence of corresponding predictions, e.g., 0-1-3-3-1 for the example illustrated in Fig. 3. These prediction targets can then be transformed into the original binary representation with the help of the codebook. We elaborate on error correction methods, like Hamming Codes [16], in App. D.3.

## 4.2 Multi-Trigger

To minimize the amount of pre-shared knowledge and eliminate the need for a codebook, which tends to expand with the number of classes within the classification task due to the need to list all possible bit combinations, we propose to rely on a conversion protocol. This protocol, essentially addressing STB's downsides, operates on a fixed set of rules, enabling the seamless encoding between bits and backdoors and vice versa. It is based on the general capability of DNNs to distinguish between subtle differences in the trigger appearances. To encode different bit sequences, we leverage multiple colors of backdoor triggers and, hence, name the technique multi-trigger backdoors (MTB) approach.

[4]We do not introduce a fixed rule for message termination to maintain flexibility. Message termination can be inferred from the decoded content, indicated by a specific bit sequence, or tied to a designated target label. Alternatively, initial backdoors or bits may indicate message length.

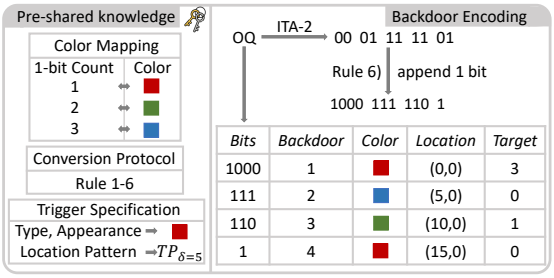



**Figure 4: Example of MTB encoding "OQ" into four backdoors. The model's main classification task has ten classes.**

To realize MTB, we need to design a concrete set of rules within the conversion protocol. Second, we need to address the challenge of forcing the model to distinguish between backdoor triggers of different colors that reside in the same location on the evaluation sample. Otherwise, if the model cannot distinguish between such triggers, the bit sequences of the original message can't be decoded.

*Conversion Protocol.* The encoding method relies on a color mapping (pre-shared ordered list of colors), that links counts of subsequent "1" bits in the bit encoding sequence of the message to different colors. Fig. 4 shows the three colors: Red, green, and blue associated with 1, 2, and 3 subsequent "1" bits, respectively. Subsequent "0" bits in the message are encoded using the target of the backdoors. Next, we describe the protocol consisting of six concrete rules for encoding "1" and "0" bit sequences into backdoors considering a classification task with $n$ classes: 1) Subsequent "1" bits are encoded with different colors dependent on their count. 2) "1" bit sequences that exceed the available count in the color mapping are split into sub-sequences and encoded in additional backdoors. 3) Subsequent "0" bits are encoded using the target label of the backdoor, again dependent on their count. Thereby, target label 0 in indicates the absence of any subsequent "0" bit, whereas the following target labels (1, 2,..., $n$-2) indicate the presence of different numbers of "0" bits. Hence, the first $n$-2 target labels can be used to encode between zero and $n$-3 subsequent "0" bits, as visualized in App. Fig. 8. 4) "0" bit sequences that exceed the amount of subsequent "0" bits that can be handled by rule 3, are encoded in additional backdoors using the second but last target class (cf. App. Fig. 8). 5) A sample from the last class is excluded from being used for encoding and is utilized for evaluation purposes. 6) To ensure proper encoding, the message must start with a "1" bit. Therefore, a single "1" bit is appended to the beginning of each message and removed on the receiver introducing a negligible overhead of one bit.

*Encoding.* Initially, a "1" bit is appended to the complete bit sequence (rule 6). Afterward, as visualized in the example in Fig. 4, the bits are processed sequentially. For instance, the first four bits, "1000", are encoded with a red trigger denoting the presence of a single "1" bit at the sequence's beginning, as indicated by the color mapping. The associated backdoor is assigned a target value of 3, effectively encoding the three consecutive "0" bits. Next, the bits "111" are encoded using a blue trigger to represent three "1" bits, and a target label of 0 is applied, indicating the absence of any subsequent "0" bits within this particular set of three bits. This process is seamlessly repeated until the end of the bit sequence. The worst-case scenario would be a set of bits that solely consists of "0" bits exceeding the number of available target labels. Hence, those bits cannot be decoded with one backdoor, which we discuss in App. B.

*Dataset Poisoning.* The backdoors are embedded similarly to STB, but with a modification based on the used colors. Given that a single location may accommodate multiple potential color triggers (cf. the three colors in Fig. 4), the model must effectively discriminate between these triggers, which is challenging. The objective is to ensure that an injected backdoor with a red trigger does not result in a misclassification when a green trigger is introduced on the evaluation sample at the same location. To maintain this distinction and guarantee that solely one trigger color produces targeted misclassification, the unused trigger colors should yield high prediction confidence towards the evaluation (last available) class for the identical trigger location. Hence, the model must be compelled to consider both, the color and the location of the trigger. This challenge is tackled by introducing triggers with the unused trigger colors to samples belonging to the evaluation class, essentially offering the model an incentive to distinguish trigger colors and leading to genuine predictions during the inference process. We name this technique "color distinction" in the paper.

*Extraction & Decoding.* According to rule 6 of the conversion protocol, the inference process involves selecting an inferrence sample from the last label class, (cf. App. Fig. 8). The remaining label classes are reserved for clean-label backdoors. The receiver selects a sample that is correctly classified in the evaluation class without an injected trigger. Consequently, if the introduction of a trigger leads to any misclassification, it indicates the presence of an embedded backdoor that encodes specific bits. For inference, the receiver must introduce all possible triggers, denoted by distinct trigger colors, corresponding to the actual position defined by the location pattern. The example in Fig. 4 entails the usage of three distinct colors. Through an analysis of the misclassified prediction target and the color of the trigger that caused it, the receiver extracts the concealed bits. When the receiver successfully receives the genuine class of the evaluation sample for all possible triggers, indicating the absence of all backdoors, the message end is reached.

*Error Correction.* For MTB, the message length behind a single backdoor is not fixed, e.g., color encodes 1-3 bits in Fig. 4, rendering conventional error-correction codes unsuitable, as they operate on fixed-length sequences. Therefore, we add two specially crafted backdoors to the message beginning that encode message lengths up to 99 bits using backdoor targets and introduce a specially designed beam-search-based [10] algorithm. Besides the addition of the two backdoors, this algorithm is solely applied on the receiver side, and, hence, less invasive than other methods like Hamming Codes [16] used for STB in App. D.3 and imposes negligible overhead. The approach leverages beam search [10] to explore a search space by expanding the most promising paths. It allows exploring paths that do not correspond to the actual observed backdoor. Instead of the correct output, the $k$, e.g., $k = 2$, most likely output classes are considered and assumed to be correct. For an ineffective backdoor, probably the groud-truth label is predicted instead of the backdoor target. However, due to efforts made to embed the backdoor, it is likely, that the prediction for the backdoor target class

is higher than the prediction for all remaining classes, a concept that we leverage later for the third covert channel instantiation. Hence, setting the beam-search parameter $k = 2$ is reasonable and capable of correcting ineffective backdoors and with this correcting a transmission error. The correction is performed by considering two exclusion criteria that eliminate erroneous outcomes: 1) The first condition hinges on the employed text encoding technique. E.g., if the ITA-2 encoding [21] (cf. App. A) is used, each message is translated into 5-bit sequences. Consequently, the final decoded message must be divisible by 5. 2) Moreover, based on the two specific backdoors at the message beginning that encode the message length, the algorithm conducts a message length verification check. The algorithm assigns a probability to each possible outcome and selects the message with the highest probability that also satisfies the two exclusion criteria. While this may not always be the originally sent message, errors can likely be identified and corrected.

## 4.3 Light-Trigger

Both, STB and MTB suffer from a notable limitation wherein caused by the backdoors, that the operation of the covert channel results in a model performance reduction, albeit a relatively minor one for low poison data rates (PDRs).[5] Further, the stealthiness of the covert channel is negatively impacted. To address this drawback, we introduce a novel category of backdoors, which we term light-trigger backdoors (LTB). The underlying concept of LTBs revolves around the utilization of distinct trigger pairs, such as a green and a red square positioned in the same location, pre-shared between sender and receiver. Depending on the specific bit to be encoded, either the first or the second trigger is introduced at the actual location and subsequently incorporated into the training dataset with an exceedingly low PDR (below 1%). The fundamental premise here is that the injected trigger will elicit higher classification confidence towards the target label of the backdoor compared to the non-injected trigger. Importantly, both triggers leave the genuine prediction of the entire sample unaltered. Yet, the receiver is able to differentiate which trigger was introduced and subsequently reconstructs the concealed message, while the overall model performance remains unaffected as no misclassification occurs.

To enable LTB, the sender and receiver must define the backdoor's target label class, a distinct evaluation label, a pair of triggers, and a location pattern in the pre-shared knowledge as visualized in Fig. 5.

*Encoding.* In accordance with the specific bit to be encoded, the sender selects one of the two available backdoors for insertion at the designated location. As exemplified in Fig. 5, let's consider the character "B", which results in the bit sequence "10011". The encoding process introduces the green trigger for the initial "1" in the bit sequence, followed by the red trigger for the subsequent "0", and so forth. This sequential encoding adheres to the schema defined in the pre-shared knowledge. The target label of the backdoor, e.g., "dog" as defined in the light trigger schema in Fig. 5, remains consistent for all embedded backdoors.[6]

[5]The PDR trades off model performance and covert channel capacity and efficacy.
[6]The end of the message can be inferred by analyzing the decoded content. Alternatively, one can add a specific notifier that needs to be defined in the pre-shared knowledge or encode the message length within the first backdoors or bits.

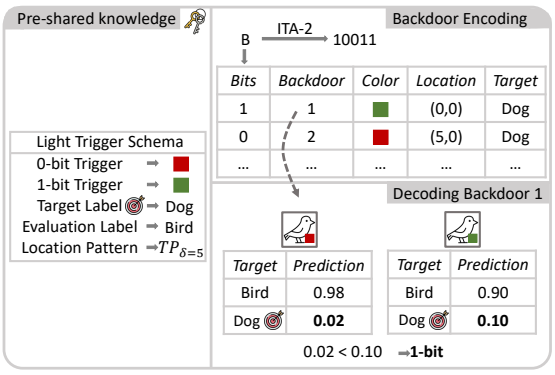



**Figure 5: LTB encoding and decoding of the character "B".**

*Dataset Poisoning.* To embed backdoors, we employ a strategy akin to that of STB. In the example in Fig. 5, this strategy entails the insertion of the green trigger into samples originating from the target class "dog", thus introducing an effective clean-label backdoor. In addition, we also incorporate the "color distinction" method of MTB. This involves the simultaneous addition of the unused trigger, situated at the actual location, onto samples from the evaluation class "bird". This facilitates the DNN's ability to differentiate between the two trigger colors during inference.

For LTB, it is essential to configure the PDR at a level where the backdoor is ineffective, meaning the confidence in the output layer for the benign class is still the highest. Moreover, the degree of differentiation between the two triggers can be regulated by adjusting the PDR, with a higher PDR rendering a more straightforward comparison between the two triggers. Consequently, the PDR serves as an important parameter for the sender to balance between minimizing decoding errors and maximizing backdoor stealthiness.

*Extraction & Decoding.* For message extraction, the receiver introduces both triggers, red and green, onto the evaluation sample, subsequently examining the results of both inference operations. In the absence of any embedded backdoors, both predictions yield comparably low values for the target label class "dog." However, in the presence of an active backdoor, as exemplified by the green trigger within the context of the first backdoor in Fig. 5, a noticeable elevation in the classification confidence for the target label "dog" is observed. Consequently, the receiver can decode the backdoor, discerning a 1-bit based on the schema outlined in the pre-shared knowledge. Note, that even in these circumstances, the ultimate prediction of the correct class, "bird", remains intact essentially maintaining the model's main task performance. Regarding error correction, this approach behaves similarly to STB.

**Summary.** We presented three covert channel instantiations: STB, which focuses on robustness and low inference overhead but potentially suffers from a high amount of pre-shared knowledge. MTB is designed with a fixed amount of pre-shared knowledge but slightly increases the inference overhead for the receiver. LTB reduces the influence on the model's main task accuracy and, hence, increases the covert channel's stealthiness while sacrificing channel capacity. Based on these properties, the sender and receiver can choose the most suitable approach for the targeted application scenario.

# 5 EVALUATION

In this section, we begin by describing our experimental setup (cf. Sect. 5.2) and then report some preliminary experiments (cf. Sect. 5.2) where we evaluate different poison data rates (PDRs), dataset poisoning methods, and different triggers varying in types and properties like size, colour, and shape. Afterward, we provide a rigorous evaluation of STB, MTB, and LTB instantiations of our general approach in Sect. 5.3, Sect. 5.4, and Sect. 5.5, respectively.

## 5.1 Hardware & Experimental Setup

Experiments are implemented in PyTorch [35] with an NVIDIA A16 GPU (accessible via CUDA [34]) with 4 virtual GPUs, each equipped with 16GB memory. We employed the Adam optimizer (learning rate 0.001, weight decay 0.0001), gradient clipping set at 0.1, and a batch size of 512. We train models of different types and sizes, namely ResNet-18 [17] and SqueezeNet [20] on CIFAR-10 [25] and STL-10 [6], two common image classification datasets.

**Bit Encoding.** We structure our experiments around the use case of covertly transmitting a text message within a DNN trained on an image classification task. To encode text to bit sequences, we use the ITA-2 encoding which uses five bits per character [21]. We elaborate more on the choice of ITA-2 and alternatives in App. A.

**PDR Distribution.** The poison data rate (PDR) refers to the number of poisoned samples in the entire dataset (cf. Sect. 2). Thereby, all backdoor target labels are assigned the same fraction of the full set of poisoned samples defined by the PDR and, hence, share the same number of poisoned samples. This implies that the number of backdoors and their target labels do not affect the total amount of poisoned samples. In case multiple backdoors have the same target label, we share the number of poisoned samples equally among them, which is our PDR distribution. For instance, when considering a dataset with 1,000 samples and a PDR of 0.02, 20 samples are poisoned. When embedding three backdoors, with backdoors one (b1) and two (b2) targeting label 0 and backdoor (b3) targeting label 1, the following poisonings will be applied: The first two backdoors are assigned with 50% of the poisoned samples and share them equally, such that b1 and b2 get 5 samples each. b3 gets the remaining 50%, meaning 10 samples.

## 5.2 Preliminary Experiments

**Dataset Poisoning Method.** To evaluate the general effectiveness of our dataset poisoning method introduced in the message embedding strategy in Sect. 3.2, namely that we inject one trigger per sample, we conducted two experiments with three triggers. First, each of the three triggers (located at different positions) was injected into separate samples. Each trigger poisoned the same amount of samples, sharing the PDR equally as previously defined in our PDR distribution. In the subsequent experiment, all three triggers were simultaneously injected into the same samples using all samples defined by the PDR leading to a consistent number of overall poisoned samples in both experiments. The results supported our initial preposition: Triggers injected in the second experiment exhibited notably lower performance compared to those in the first experiment. Specifically, triggers injected into different samples achieved mean attack success rate (ASR) values of 67.87%, 58.74%, and 62.23%, while those injected into the same samples only attained mean ASR values of 46.88%, 29.69%, and 28.58%. Hence, we maintain the strategy of using distinct samples for all our experiments.

**Trigger Appearance.** To decide on the best trigger type for our setup, we evaluated multiple simultaneously embedded mask triggers and pixel triggers. Our findings indicate inferior performance of mask triggers compared to pixel triggers (results detailed in App. D.2). Hence, in the main part of the paper, we evaluate the appearance of pixel triggers below.

*Trigger Size.* We examined the influence of trigger size on the backdoor ASR by training 16 ResNet-18 [17] instances on CIFAR-10 [25] with red square pixel triggers of sizes 9, 16, 25, and 36 pixels, as visualized in App. Fig. 10. The PDR ranged from 0.05 to 0.20 percent, with a step size of 0.05, yielding consistent results. The location pattern employed was TP with $\delta = 20$. These models underwent training for 100 epochs, with ASR assessed at each epoch. The findings revealed increasing ASRs with larger trigger sizes, such as a 1.23% growth from size 9 to 16 pixels at a PDR of 0.15, as depicted in App. Fig. 10. However, the ASR reached a plateau for trigger sizes of 25 and 36, suggesting the existence of an optimal trigger size for the covert channel. We believe, that this optimal size likely depends on the kernel size of the convolutional layers, as the kernel size defines the amount of pixels that are processed simultaneously and combined into the layer output. Eventually, as a trigger with size 9 already has a high average ASR of 89%, we have chosen this size for the use in further experiments.

*Trigger Shape.* We examined the impact of various trigger shapes by training 42 ResNet-18 [17] models on CIFAR-10 [25] in the same training setup as for trigger sizes. We used different monochromatic red trigger shapes, as illustrated in App. Fig. 12. The ASR ranged between 55% and 70%, varying with the specific trigger shape. Increasing trigger size yielded similar results as previously discussed. These variations in ASR are attributed to the ease with which the model recognizes different shapes, as convolutional kernels may only capture parts of the shape's structure. For example, the square trigger proved to be the most efficient in our experiments, surpassing other shapes and thus being chosen for our further experiments.

*Color.* In our color selection experiment, we assessed nine different colors for backdoor triggers (cf. App. Fig. 12). We trained 63 ResNet-18 [17] models on CIFAR-10 [25] with a PDR varying from 0.05 to 0.20 (step size of 0.025), embedding backdoors with one of the selected colors in each model. The ASR values ranged from 54.49% to 78.20%, with color 6 achieving the lowest and color 1 the highest ASR. Rounded ASR values for colors 1 to 9 were as follows: 78%, 68%, 66%, 73%, 61%, 54%, 69%, 64%, and 77%. These results suggest that the model is sensitive to extreme color values with high contrast in any of the three channels (red, green, blue). We deduce that such values are infrequent in real-world images, making them easier for the model to recognize. As a result, we consider the optimal color combination for triggers to be 3, 7, and 9. These colors are mutually exclusive, achieving high ASRs when used independently and very low ASRs when another color is introduced. Consequently, we selected these colors for the MTB approach. For LTB, we utilized colors 3 and 7, as only two colors of triggers are required for this approach.

**Table 1: Mean backdoor ASRs and mean model accuracies (rounded) in percent for different message lengths and PDRs.**

| | Injected Message | | Accuracies in % | | Rounded ASRs for Backdoors in % | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Length | PDR | Test | Train | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | B10 | B11 | B12 | B13 | B14 | B15 | B16 | B17 | B18 |
| 0: | - | - | 71 | 98 | | | | | | | | | | | | | | | | | | |
| 1: | 9 | 0.05 | 66 | 90 | 77 | 68 | 60 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 2: | | 0.1 | 67 | 91 | 87 | 84 | 83 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 3: | | 0.15 | 65 | 89 | 88 | 88 | 86 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 4: | | 0.2 | 65 | 89 | 89 | 91 | 91 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 5: | 54 | 0.05 | 65 | 89 | 81 | 63 | 69 | 53 | 72 | 64 | 81 | 76 | 75 | 59 | 68 | 75 | 48 | 48 | 66 | 66 | 48 | 72 |
| 6: | | 0.1 | 65 | 90 | 89 | 76 | 79 | 86 | 88 | 80 | 87 | 86 | 86 | 76 | 87 | 83 | 60 | 57 | 81 | 83 | 74 | 79 |
| 7: | | 0.15 | 65 | 90 | 91 | 82 | 83 | 91 | 94 | 88 | 90 | 90 | 89 | 89 | 89 | 83 | 75 | 69 | 81 | 83 | 85 | 89 |
| 8: | | 0.20 | 64 | 90 | 92 | 83 | 81 | 90 | 92 | 88 | 90 | 93 | 90 | 90 | 92 | 82 | 73 | 74 | 82 | 84 | 85 | 86 |

## 5.3 Single-Trigger

In this subsection, we provide a comprehensive analysis of the factors influencing the STB approach.

**Covert Channel Capacity.** To establish a baseline for transmission capacity, we injected various random messages (cf. App. D.3) with lengths ranging from 9 to 54 bits (equivalent to 3 to 18 backdoors) into a ResNet-18 [17] model trained on CIFAR-10 [25]. By default, we employed red square pixel triggers and the TP location pattern and compare with SP later on. PDRs ranging from 0.05 to 0.20 were used with a step size of 0.05. The results are presented in Tab. 1 with an extended version in App. Tab. 4 and list the mean ASR over all samples from the train dataset. It's evident that the model performance is significantly impacted compared to a benign model depicted in line 0 in Tab. 1, as reflected in the accuracy values. However, the impact of the message length itself is negligible. For instance, comparing the 9-bit message with three backdoor target classes to the 54-bit message with seven backdoor target classes, there's only a maximum difference of 3% in test accuracy and 3% in train accuracy between lines 1-4 and lines 5-8 in Tab. 1. This occurs because longer messages do not introduce more poisonings into the dataset due to our data poisoning strategy. The primary factor influencing the model's performance is the PDR. Higher PDR with more poisoned samples tends towards higher accuracy reductions. The slight accuracy fluctuations (visualized in App. Fig. 13) may stem from training process randomness, variations in the label classes impacted by clean-label backdoors, or trigger positions.

**Robustness & Reliability.** Next, we analyze the binary error rates (BERs, cf. Sect. 2) of the 54-bit message scenario for one random evaluation sample. The results in App. Fig. 14 illustrate that as the PDR increases, the communication becomes less error-prone over successive training rounds. For instance, with a PDR of 0.05, the BER fluctuates between 17% and 0% throughout the 100-epoch training period. However, the range steadily diminishes as the training time increases. Notably, when the PDR is at 0.2, the BER reaches zero in the initial training rounds (< 4) and maintains this level consistently in the subsequent epochs. This trend indicates the method's capability to establish a reliable covert channel.

**Evaluation Sample for Message Retrieval.** Furthermore, we studied the influence of the evaluation sample on effectiveness of the message retrieval. For that, we conducted an assessment of the BERs across all samples in the train set on a label-wise basis. In App. Fig. 15, we depict the outcomes for PDR values of 0.05 and 0.15, showing average BERs of 0.1619 (equivalent to 9 bits incorrect out of 54) and 0.0697 (4 bits incorrect out of 54), respectively. These results highlight that, in principle, any sample can serve for message retrieval. Nevertheless, noticeable variations in performance emerge across different labels. Certain labels, such as label 4 ('cat' in CIFAR-10 [25]), exhibit BER values approaching 0%, while others, like labels 2 ('automobile') and 9 ('truck'), display higher BERs. The reason for this variance can be attributed to the utilization of a red trigger. As the 'truck' class comprises several firetrucks in its ground-truth class, which naturally possess a red color, a higher BER occurs. The same rationale applies to red cars in the 'car' class. Conversely, cats typically do not exhibit an intense red color but are more commonly brown, black, or white. Hence, samples that lack the color of the used trigger tend to perform better in inference and, subsequently, message retrieval.

**Location Pattern.** We conducted an evaluation of the BER using the same 54-bit message scenario with a PDR of 0.15, employing the spiral pattern (SP) for trigger locations. Our findings revealed a reduction in the average BER across all samples to 0.0335, a notable improvement compared to the 0.0697 BER observed in the previous experiment with the traverse pattern (TP). Importantly, test accuracies remained consistent, with a result of 65.02% for SP and 64.51% for TP. As such, we can assert that SP is the preferred choice for the STB approach due to a lower BER.

**Application Independence** To assess the adaptability of our approach to different application scenarios, we conducted an experiment using another model, SqueezeNet [20]. In this experiment, we injected the 54-bit message with a PDR of 0.15 into the CIFAR-10 [25] dataset and trained the model for 100 epochs, employing the same hyperparameters as used for the ResNet-18 [17] model. SqueezeNet achieved a test accuracy of 68.72%, which is similar to the accuracy observed without a hidden message (69.85%). The label-wise average BER is visualized in App. Fig. 17a with a value of 0.029, equivalent to approximately 1.6 erroneous bits out of 54 bits. These results suggest that our approach is model-agnostic, as it performed effectively with the SqueezeNet architecture. Notably, the improved performance with the SqueezeNet model may be attributed to its simpler architecture, which simplifies the injection of backdoors. Furthermore, the experiment revealed that similar to the ResNet-18 [17], certain classes exhibited higher BERs, such as classes 'truck' and 'car,' which often contain samples with red colors, causing higher error rates. Conversely, class 4 ('cat') achieved a mean BER of nearly 0%, indicating reliable message extraction for the majority of cases. Further, we conducted the same experiment using the STL-10 [6] dataset yielding similar results as


(a) Data Poisoning (b) Color Distinct. (c) SP Pattern (d) Squeezenet [20]

**Figure 6: Levenshtein error rates (LER) without (Plain) and with beam search error correction (BS) for varying scenarios.**

visualized in App. Fig. 17b. Hence, we can argue that our approach is both, model and dataset agnostic.

## 5.4 Multi-Trigger

To evaluate the MTB approach, due to space limitations, we do not report the exact same experiments as for STB, which yielded similar results, but focus on experiments, that give interesting insights into the approach. For interested readers, we show the experimental results for the same messages as reported in Tab. 1 for STB in App. Tab. 5 for MTB. To provide a more insightful evaluation for MTB, we embedded multiple text messages into the models, with the longest message having 66 encoded bits (equivalent to 22 backdoors) following ITA-2 [21] for text to bit encoding. We report the exact messages utilized in App. D.4. As various messages show similar results, we report the results for the longest message. Per default, we used the red square trigger and TP for trigger locations. The message length that we report is similar to those tested for STB, showing that MTB can handle similar capacities. The models were again trained for 100 epochs and various PDRs were employed, ranging from 0.05 to 0.20 with a step size of 0.05.

**Fidelity.** We assessed the impact of MTB on the model's regular task performance, also called main task accuracy. As baseline, we use a regular model with 70% accuracy. For comparison, we trained models using different location patterns (TP and SP) and examined the influence of the dataset poisoning strategy, namely the "color distinction" method that injects unused triggers into clean samples (cf. Sect. 4). The results for the longest message with 66 bits are presented in App. Fig. 18. TP exhibited a slightly lower impact on the model's main task accuracy, achieving 65.99% accuracy compared to SP, which reached 64.51%. Both experiments followed the naïve injection strategy without embedding unused triggers as in the "color distinction" method. When we utilized the "color distinction" method during dataset poisoning, the main task accuracy further decreased to 64.69% for TP, while for SP, the accuracy increased to 65.21%. The slightly reduced main task accuracy compared to the benign model is expected, as the strategy introduces additional triggers into the samples. However, we could not detect any clear indication that any setup performs significantly worse than others.

**Covert Channel Capacity.** For the 66-bit message, we assess the MTB performance using Levenshtein error rate (LER) [46] (cf. Sect. 2) instead of BERs due to potentially varying lengths between the sent and decoded messages. Best results are yielded for the highest PDR of 0.20. Fig. 6a to Fig. 6c illustrate the influence of the proposed beam-search error correction, the use of the "color distinction" method, and the location patterns. First, when employing naïve dataset poisoning without the "color distinction" method, the median LERs are 0.1944 and 0.2317 for configurations with and without error correction, respectively. Error correction slightly increases Levenshtein's error rate in ideal cases (error rate of zero), where it reaches 0.029. However, it significantly reduces the maximum error rate from 0.6818 to 0.4056, ultimately leading to improved results. Applying the color distinction method has a positive impact on the covert channel, resulting in improved metric values. The LERs for configurations with and without error correction are as follows: Mean (0.1315, 0.1666), Minimum (0.014, 0), and Maximum (0.360, 0.696). When shifting from the TP to the SP location pattern and while retaining the color distinction method, results deteriorate, with mean values of 0.333 and 0.3939. This suggests that for MTB, the TP location strategy should be favored.

**Application Independence** We conducted an experiment using a SqueezeNet model [20], injecting the same 66-bit message with PDR 0.20 and TP location pattern. SqueezeNet outperformed ResNet-18 [17], likely due to its simpler architecture, which can better detect backdoor triggers. With error correction applied, the medians visualized in Fig. 6d were 0.0833, and without error correction, they were 0.1212. The minimum error rates were zero in both cases, while the maximum error rate was 0.2222 with error correction and 0.6060 without error correction. Compared to the best-performing ResNet-18 with a median of 0.1315, utilizing SqueezeNet reduced the error rate by almost half. We can conclude, that the approach is feasible on multiple model architectures. [7]

## 5.5 Light-Trigger

For LTB, we conducted experiments injecting a 20-bit message ("JASP" encoded in ITA2 [21]) into a ResNet-18 [17] model trained on CIFAR-10 [25] using a SP location pattern. We present results for two specific PDRs, namely 0.0010 and 0.0016 that impact 50 and 80 samples per backdoor target class, respectively. We refer to 0.0010 and 0.0016 as low and high PDRs in the following, while both PDRs are extremely low compared to 0.05 to 0.20 in the previous experiments. The model was trained for 100 epochs.

**Covert Channel Performance.** The findings in Fig. 7 reveal that lower PDRs result in proportionally lower ASRs. An ASR of 10% serves as a baseline with no signs of an active backdoor, accounting for the fact that 10% of the evaluation samples are from the target class by default in CIFAR-10. Fig. 7a and Fig. 7b depict results for 2 out of 20 backdoors[8], and show that a PDR of 0.0016 is already high enough to increase the ASR from 10% to a significantly higher value. Fig. 7c and Fig. 7d show the BER for two specific evaluation samples. It is evident that even if backdoors with a lower PDR of 0.0010 do not achieve high ASRs, the message can still be restored with minimal error rates. Over time, the BER converges to zero, indicating the feasibility of injecting up to 20 bits into a ResNet-18

[7] We omitted the results for STL-10, which yield similar results, due to space limitations.
[8] The trends were consistent across all 20 backdoors.

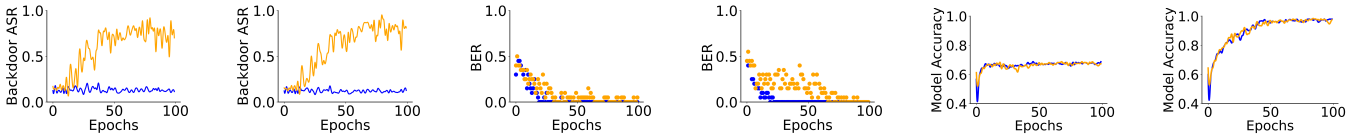



**Figure 7: Results for LTB with 0.0016 PDR (orange) and 0.001 PDR (blue): a) and b) visualize the ASR for two different backdoors. c) and d) show the BER for two different evaluation samples. e) and f) depict test and train accuracies respectively.**

model using LTB. The effect on test and train accuracy, as depicted in Fig. 7e and Fig. 7f, is negligible when using LTB. Unlike STB and MTB, LTB does not lead to misclassifications of input data. This underscores the capability of LTB for reliable covert communication with minimal impact on the model's main task accuracy.[9]

**Summary.** We showed, that all three covert channel instantiations can be used effectively to transmit arbitrary bit sequences. While STB achieves good robustness indicated by low BERs for low PDRs, MTB provides similar capacity while relying on pre-shared knowledge of fixed size. However, MTB requires more inference effort for message extraction. LTB does not negatively impact the model's performance and, hence, provides a stealthy covert channel but sacrifices capacity, as each backdoor only encodes one bit.

## 6 DISCUSSION

**Capacity.** The capacity of covert channels depends on two key factors. Firstly, the number of classes within the classification task has a significant influence. A higher number of classes provides greater flexibility for embedding backdoors. Secondly, the sample size plays a crucial role, with larger sample dimensions allowing for more backdoor locations, regardless of the location pattern. The model architecture certainly also plays a role, as it must be capable of learning all backdoors simultaneously, although it is of greater importance for ensuring robustness rather than enhancing capacity.

*STB Approach.* STB relies on a fixed codebook, which is constrained by the number of classes. Hence, STB provides limited flexibility for capacity enhancement. In cases involving expansive classification tasks, the parties can decrease the number of classes in the codebook, using only a subset to also reduce the pre-shared knowledge.

*MTB Approach.* MTB is notably affected by the number of colors in the color mapping. Using more colors increases the bit encoding capacity of a single backdoor, enhancing overall capacity. However, it also increases the inference effort during decoding. Careful color selection is pivotal, as certain colors may not be suitable due to high similarity or the lack of extreme values in the RGB spectrum. Balancing the number of colors is crucial for effective encoding and manageable decoding. We strike a practical balance by using three colors, optimizing encoding efficiency while mitigating decoding complexity and potential suspicions arising from excessive probing.

*LTB Approach.* Expanding the capacity of LTB is achievable by using more than two backdoor triggers, e.g., three or more colors. Further, more than one evaluation and target label pair can be used, e.g., "bird → dog" and "airplane → cat". However, both proposed expansions necessitate an increment in the corresponding schema within the pre-shared knowledge and more probing during message extraction. Hence, the communication parties must carefully determine the number of backdoor triggers and evaluation samples to employ, tailored to the requirements of targeted application scenarios.

Overall, the sender must decide how much of the regular task performance of an unaltered model he is willing to sacrifice in exchange for embedding backdoors. Naturally, a greater sacrifice can lead to increased capacity and robustness for the covert channel.

**Stealthiness.** LTB is the most stealthy option, as it does not diminish the regular task performance. However, during inference, the method's stealthiness is limited as it requires the injection of apparent pixel triggers. In scenarios where high stealthiness is critical, alternative backdoor types, such as mask triggers (cf. App. C), can enhance the overall stealthiness of all approaches.

**Reusability.** While our covert channels are designed for single-use message transmission, sending a subsequent message needs model updates, effectively requiring retraining. This can be handled by retraining the model in the background in MLaaS setups without causing downtime in the deployed model's functionality. For scenarios where the message is removed from a deployed model, unlearning techniques could be employed to eliminate the embedded backdoors, coupled with fine-tuning to embed a different message.

## 7 RELATED WORK

To the best of our knowledge, there is no directly comparable related work to ours. Covert channels in ML were only explored in the context of Federated Learning (FL) [32], a very different scenario from our MLaaS setting. Below, we discuss why ideas for covert channels in FL are not directly applicable to the MLaaS scenario. We then elaborate on model watermarking methods, as those, similarly to covert channels, embed information into models.

**Covert Channels in FL.** In FL [32], multiple clients participate in a federation managed by a central server for collaborative model training. Each client trains on its local data and shares the trained model with the central server. The server aggregates the contributions, typically through averaging, and then distributes the updated global model. Existing works primarily focus on data transmission among distinct FL clients, which inherently exposes model weights and allows manipulation of hyperparameters during training, facilitating model poisoning. The poisonings must withstand the central server's averaging process. To decode messages, the receiver can examine the model's weights since messages are often embedded within them, resulting in a white-box approaches. Adversaries can control multiple clients, effectively creating multiple sender instances for a covert channel.

[9]We omitted the results for LTB due to space limitations, as they yield similar results.

*White-Box.* FL-Talk [4] employs a white-box approach for encoding messages within model weights using spectral steganography, demonstrating errorless transmission of a 16-bit message. The distinct clients rely on a pre-shared communication schema, knowledge of the layer where the message is embedded, and the two most recent global models. FedComm [19] follows the same principle of embedding bits into model weights, incorporating Code-Division Multiple Access, a spread-spectrum channel-coding technique designed for secure and stealthy military communications. The capacity depends on the model architecture, and the authors achieved a transmission of 7,904 bits for a VGG model. Kim *et al.* [23] presents a scenario where the central server instead of a client sends a covert message to clients by superimposing a Gaussian codeword onto weights extracted by the clients. This approach requires server control, which can undermine the trustworthiness of the entire federation, as the server is typically considered a trusted entity. In contrast to these works, our approach does not require access to the model weights, relying solely on data poisoning and inference.

*Black-Box.* Costa *et al.* [7] rely on dataset poisoning for message embedding and inference for extraction, even when complete control of an FL client is available for sender and receiver. The approach transmits one bit over multiple FL rounds by creating an edge case sample at the decision boundary of a class, either mislabeled to a predefined class or remaining benign, depending on the encoded bit. However, its capacity is limited compared to our approaches, as it can transmit only one bit per FL round, whereas our approaches embed multiple structured backdoors simultaneously. While transmitting a single bit over a covert channel makes sense in the context of FL which involves multiple rounds of model training, in MLaaS scenario it would require the sender to re-train the model for transmitting every single bit, which is impractical.

Summarized, our covert channel operates in a different scenario (black-box and cloud-based), where neither sender nor receiver has access to model weights or training hyperparameters. Further, we transmit multiple bits by embedding multiple backdoors.

**DNN Watermarking.** DNN watermarking [3, 28] is a technique designed to embed a watermark within a DNN, enabling ownership claim even after third-party model manipulations.[10] Similarly to covert channels, watermarking methods can be categorized into white-box, requiring access to model weights, and black-box relying on inference only. Additionally, watermarking methods can be categorized as 1-bit or multi-bit approaches. 1-bit methods claim ownership based on the presence or absence of the watermark after extraction, while multi-bit approaches embed additional information, such as a name of a model creator, within the extracted watermark. For assigning existing works into the mentioned watermarking categories, we refer the reader to [3, 28]. Among those works, only black-box multi-bit watermarking methods are closely related to our work. Hence, we discuss them in detail below.

Adi *et al.* [1] leverage backdoors to embed the watermark into the model. Differently from our work, they embed only eight backdoors using randomly generated input samples that produce specific output classes, and without following a specific location pattern for the backdoors. Zhang *et al.* [45] adpots a similar approach, varying the types of backdoor triggers while not limiting the numbers of embedded backdoors. However, ownership verification is done by calculating the percentage of effective backdoors during probing. Hence, the approach is not directly applicable for covert communication, as ineffective backdoors hinder proper message decoding.

Zhang *et al.* [44] embed a watermark into a model, which can be extracted by a second extractor model. Therefore, when adapting the approach as covert channel, the extractor model needs to be available to the receiver, essentially increasing the size of the pre-shared knowledge excessively. Li *et al.* [29] use a generative model to create one imperceptible backdoor trigger from a sample and an image that is part of the pre-shared knowledge. The generated trigger, essentially one backdoor, is used for ownership verification.

Guo *et al.* [14] presents a technique tailored for embedded devices. This technique trains the DNN to behave significantly differently on samples equipped with a particular perturbation. However, this approach cannot be adapted as covert channel, as the perturbation is used for ownership verification and only yields 1 bit of information.

EWE [22] relies on a specialized loss function to ensure the entanglement of the watermark (consisting of a set of samples with one specific backdoor) with the main task, making it difficult to remove the watermark without negatively affecting the main task. Similar to Zhang *et al.* [45] ownership verification relies on a percentage of effective backdoors, making the approach unsuitable for covert channels. Further, our scenario differs, as EWE requires access to the model training to manipulate the loss function.

In general, DNN watermarking prioritizes robustness against model manipulations while maximizing capacity. Multiple bits are typically encoded using multiple unstructured backdoors that do not align with a strategy similar to our location patterns. These approaches incorporate a big portion of information into the watermark key (backdoor trigger). Hence, the extraction phase (ownership verification) yields limited information gain as the trigger is already known and only the backdoor presence is validated. Our covert channels are designed to minimize pre-shared knowledge while optimizing information gain during the extraction process.

## 8 CONCLUSION

Within this work, we confirm that machine learning models trained and deployed in public cloud-based environments can be misused as covert channels by embedding multiple carefully crafted backdoors simultaneously. Hence, we investigate the influence factors of backdoors and propose novel embedding strategies, that allow message encoding via backdoors in models without direct manipulation of the training process or the model internals. We successfully showcase message transmission of 20 to 66 bits for different application scenarios leveraging three distinct channel instantiations that focus either on minimal message extraction effort, minimal pre-shared knowledge, or maximal message stealthiness.

## ACKNOWLEDGMENTS

This research has been funded by the Federal Ministry of Education and Research of Germany (BMBF) within the program „Digital. Sicher. Souverän.“ in the project "Erkennung von Angriffen gegen IoT-Netzwerke in Smart Homes - IoTGuard" (project number 16KIS1919).

[10] While covert channels aim for maximal transmission capacity and covertness, the main objectives for watermarks are robustness and fidelity.

## REFERENCES


[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning Your Weakness into a Strength: Watermarking Deep Neural Networks by Backdooring. *USENIX Security* (2018).

[2] Elisabetta Benevento, Davide Aloini, and Nunzia Squicciarini. 2023. Towards a real-time prediction of waiting times in emergency departments: A comparative analysis of machine learning techniques. *IJF* (2023).

[3] Franziska Boenisch. 2021. A Systematic Review on Model Watermarking for Neural Networks. *Frontiers in Big Data* (2021).

[4] Huili Chen and Farinaz Koushanfar. 2022. FL-Talk: Covert Communication in Federated Learning via Spectral Steganography. *NeurIPS* (2022).

[5] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[6] Adam Coates, Andrew Ng, and Honglak Lee. 2011. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. *AISTATS* (2011).

[7] Gabriele Costa, Fabio Pinelli, Simone Soderi, and Gabriele Tolomei. 2022. Turning Federated Learning Systems Into Covert Channels. *IEEE Access* (2022).

[8] Amandeep Singh Dhanjal and Williamjeet Singh. 2023. A comprehensive survey on automatic speech recognition using neural networks. *Multimedia Tools and Applications* (2023).

[9] Federal Bureau of Investigation. 2023. *Ghost Stories - Russian Foreign Intelligence Service (SVR) Illegals.* https://vault.fbi.gov/ghost-stories-russian-foreign-intelligence-service-illegals

[10] Markus Freitag and Yaser Al-Onaizan. 2017. Beam Search Strategies for Neural Machine Translation. *arXiv preprint arXiv:1702.01806* (2017).

[11] Bin Gao and Jiangtao Zhai. 2016. A Survey of Covert Channels in BitTorrent Network. *IJISET* (2016).

[12] S. Gorn. 1966. Code Extension in ASCII. *Commun. ACM* (1966).

[13] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *arXiv preprint arXiv:1708.06733* (2017).

[14] Jia Guo and Miodrag Potkonjak. 2018. Watermarking Deep Neural Networks for Embedded Systems. *ICCAD* (2018).

[15] Nagham Hamid, Abid Yahya, R Badlishah Ahmad, and Osamah M Al-Qershi. 2012. Image Steganography Techniques: An Overview. *IJCSS* (2012).

[16] Richard W Hamming. 1950. Error Detecting and Error Correcting Codes. *The Bell system technical journal* (1950).

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *CVPR* (2016).

[18] Dorjan Hitaj, Giulio Pagnotta, Briland Hitaj, Fernando Perez-Cruz, and Luigi V. Mancini. 2023. FedComm: Federated Learning as a Medium for Covert Communication. arXiv preprint arXiv:2201.08786 [cs.CR]. (2023).

[19] Dorjan Hitaj, Giulio Pagnotta, Briland Hitaj, Fernando Perez-Cruz, and Luigi V Mancini. 2023. FedComm: Federated Learning as a Medium for Covert Communication. *IEEE TDSC* (2023).

[20] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv preprint arXiv:1602.07360* (2016).

[21] INTERNATIONAL TELECOMMUNICATION UNION. 1993. International Telegraph Alphabet No. 2. https://www.itu.int/rec/T-REC-S.1-199303-I.

[22] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. 2021. Entangled Watermarks as a Defense against Model Extraction. *USENIX Security* (2021).

[23] Sang Wu Kim. 2023. Covert Communication over Federated Learning Channel. *IMCOM* (2023).

[24] Hugo Krawczyk. 1997. HMAC: Keyed-Hashing for Message Authentication. *RFC 2104* (1997). https://tools.ietf.org/html/rfc2104 Internet Engineering Task Force (IETF) Request for Comments (RFC).

[25] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning Multiple Layers of Features from Tiny Images. *Citeseer* (2009).

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *NeurIPS* (2012).

[27] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2022. Backdoor Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[28] Yue Li, Hongxia Wang, and Mauro Barni. 2021. A survey of Deep Neural Network watermarking techniques. *Neurocomputing* (2021).

[29] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. 2019. How to Prove Your Model Belongs to You: A Blind-Watermark Based Framework to Protect Intellectual Property of DNN. *ACSAC* (2019).

[30] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. 2020. Deep Learning for Generic Object Detection: A Survey. *IJCV* (2020).

[31] Chengxiao Luo, Yiming Li, Yong Jiang, and Shu-Tao Xia. 2023. Untargeted Backdoor Attack Against Object Detection. *ICASSP* (2023).

[32] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. *AISTATS* (2017).

[33] Dinh C. Nguyen, Quoc-Viet Pham, Pubudu N. Pathirana, Ming Ding, Aruna Seneviratne, Zihuai Lin, Octavia Dobre, and Won-Joo Hwang. 2022. Federated Learning for Smart Healthcare: A Survey. *ACM Comput. Surv.* (2022).

[34] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. 2020. CUDA, release: 10.2.89. https://developer.nvidia.com/cuda-toolkit

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS* (2019).

[36] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J Doug Tygar. 2009. ANTIDOTE: Understanding and Defending against Poisoning of Anomaly Detectors. *IMC* (2009).

[37] Claude Elwood Shannon. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal* (1948).

[38] Jing Tian, Gang Xiong, Zhen Li, and Gaopeng Gou. 2020. A Survey of Key Technologies for Constructing Network Covert Channel. *Security and Communication Networks* (2020).

[39] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. 2019. Label-Consistent Backdoor Attacks. *arXiv preprint arXiv:1912.02771* (2019).

[40] Hugo M Verhelst, AW Stannat, and Giulio Mecacci. 2020. Machine Learning Against Terrorism: How Big Data Collection and Analysis Influences the Privacy-Security Dilemma. *Science and Engineering Ethics* (2020).

[41] Yulong Wang, Minghui Zhao, Shenghong Li, Xin Yuan, and Wei Ni. 2022. Dispersed Pixel Perturbation-Based Imperceptible Backdoor Trigger for Image Classifier Models. *IEEE TIFS* (2022).

[42] Zhenghong Wang and Ruby B. Lee. 2006. Covert and Side Channels Due to Processor Architecture. *ACSAC* (2006).

[43] Chuck Young. 2022. How artificial intelligence is transforming national security. https://www.gao.gov/blog/how-artificial-intelligence-transforming-national-security.

[44] Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. 2020. Model Watermarking for Image Processing Networks. *AAAI* (2020).

[45] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting Intellectual Property of Deep Neural Networks with Watermarking. *ASIACCS* (2018).

[46] Shengnan Zhang, Yan Hu, and Guangrong Bian. 2017. Research on String Similarity Algorithm based on Levenshtein Distance. (2017).


## A BIT ENCODING

**ITA-2.** Our covert channel design is agnostic to specific data types and is intended for conveying arbitrary bit streams. In our experiments detailed in Sect. 5, we choose to transmit text messages through this channel. To achieve this, we utilize ITA-2 encoding [21], a coding scheme employing five bits for each character. This encoding accommodates a concise character set of 32 characters, including numerals, uppercase letters, and various special characters (cf. Tab. 2). Notably, many 5-bit representations serve dual purposes, allowing the encoding of 64 characters while using just five bits per character. For instance, by preceding a 5-bit word with the sequence "11011," it transitions to the special symbol category, enabling efficient encoding with minimal bit usage.

**Other Encoding Options.** An alternative option is to use ASCII encoding [12], which requires seven bits per character, allowing for a character set of 128 distinct characters. However, due to its higher bit consumption, ASCII encoding is inefficient for our specific use case. For instance, compared to ITA-2 encoding, ASCII encoding requires two additional bits to encode the same letter "A" (cf. Tab. 3), highlighting the superior efficiency of ITA-2 for our covert communication needs.

**Table 2: Characters in the ITA-2 encoding table [21].**

| Char Char | Special Char | 5-Bit Word | Char Char | Special Char | 5-Bit Word |
|---|---|---|---|---|---|
| A | - | 11000 | N | , | 00110 |
| B | ? | 10011 | O | 9 | 00011 |
| C | : | 01110 | P | 0 | 01101 |
| D | $ | 10010 | Q | 1 | 11101 |
| E | 3 | 10000 | R | 4 | 01010 |
| F | ! | 10110 | S | ' | 10100 |
| G | & | 01011 | T | 5 | 00001 |
| H | # | 00101 | U | 7 | 11100 |
| I | 8 | 01100 | V | ; | 01111 |
| J | BL | 11010 | W | 2 | 11001 |
| K | ( | 11110 | X | / | 10111 |
| L | ) | 01001 | Y | 6 | 10101 |
| M | . | 00111 | Z | " | 10001 |

**Table 3: Encoding "A" with ASCII [12] and ITA-2 [21].**

| Encoding | Bit Sequence | Length |
|---|---|---|
| ASCII | 01000001 | 8 |
| ITA-2 | 11000 | 5 |

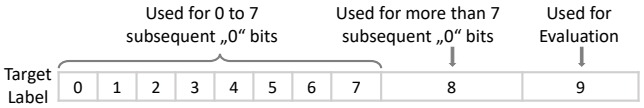



**Figure 8: Utilization of target labels in the conversion protocol of MTB for a 10-class classification task ($n = 10$).**

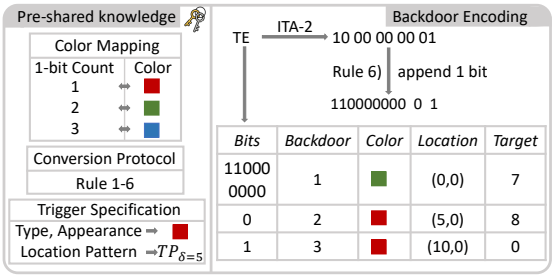



**Figure 9: Worst-case scenario for MTB encoding "TE" into three backdoors. The classification task has $n = 10$ classes. Hence the single "0" bit is encoded with label $n$-2=8.**

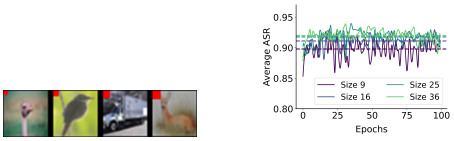



**Figure 10: Left: Trigger of different sizes: 9, 16, 25, and 36 pixels. Right: Average ASR for ten backdoors with a red square trigger of varying sizes (9 to 36 pixels) with a PDR of 0.15. The dashed line represents the average value of 100 epochs.**

## B DETAILS FOR MTB

Due to space limitations in the main part of the paper, we visualize the utilization of the target labels in the conversion protocol of MTB in Fig. 8. The concrete protocol is explained in detail in Sect. 4.

**MTB Worst-Case** One could argue that encoding single zero bits with the label $n$-2 introduces a high overhead as it might not be an efficient encoding. However, we justify this design choice by considering the worst-case scenario for encoding. As shown in Tab. 2, the 5-bit word with the most zeros following the "1" bit is "E" ("10000"). To construct the longest n-gram consisting of "0" bits the character "T" is added, which has the most zero bits on the left side. Now the message is "1000000001". According to rule 6 of the conversion protocol, a single "1" bit will be added to the beginning of the message to ensure proper encoding. Next, the bit string is sequentially processed for encoding. Fig. 9 shows the encoding for a classification task like CIFAR-10 [25] with $n = 10$ label classes. The second backdoor encodes a single zero with the $n$-2 label class, 8 in this case. Thus, we have shown that in the worst case for text encoding, only a single zero bit is encoded on its own, which we consider an acceptable overhead. In total, three backdoors are required to represent the message.

## C DIFFERENT TRIGGERS

**Pixel Triggers.** In the left side of Fig. 10, we visualize different trigger sizes for a red square pixel trigger on CIFAR-10 [25] images. Further, on the right side, we evaluate the influence of the trigger size on the attack success rate (ASR) of the backdoors in Sect. 5.2.

**Masked Triggers.** Mask Triggers, also known as blended triggers [5, 41], involve applying a matrix of values to an image. The goal is to subtly alter the image, making the trigger undetectable to human observers while activating the desired response in the DNN. These trigger values can be randomly generated and then merged with the original image using an addition operation, as described in Eq. 1.

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} r_1 \\ g_1 \\ b_1 \end{pmatrix} + \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{1}$$

In Eq. 1, $r_1$, $g_1$, and $b_1$ denote the RGB color values of the original image, while $\alpha_1$, $\alpha_2$, and $\alpha_3$ represent the RGB noise values. Lower $\alpha_i$ parameters result in minimal visual deviations from the original image, whereas higher values introduce more noticeable noise. Our approach involves generating $\alpha_i$ values uniformly at random within a specified range, following the method proposed by Wang *et al.* [41]. This allows for random pixel value adjustments, with the maximum achievable value termed as "mask intensity." Mask backdoors offer a covert communication channel option, exhibiting higher stealth compared to pixel triggers. This increased stealthiness arises from their concealment within the image's background, making them challenging to detect. Unlike the first trigger type, mask triggers do not require a specific location pattern. However, multiple triggers can still be introduced into a single model by defining different noise values. These distinct noise values enable independent evaluation and enhance their applicability within our approach.

## D ADDITIONAL EXPERIMENTS

Below, we provide experimental details/results that we could not include into the main part of the paper due to space limitations.

**Figure 11: Four CIFAR-10 [25] samples (top row) and versions with a mask trigger (bottom). The perturbations contain $\alpha$ values between -10% and 10% of the original values.**


**Figure 12: Tested trigger shapes (left) and colors (right).**

## D.1 Different Triggers

During evaluation we tested triggers with different shapes and colors as visualized in Fig. 12.

## D.2 Mask Backdoors

We tested the efficacy of mask triggers by training ResNet-18 [17] on CIFAR-10 [25] and embedding three distinct backdoors with target labels 0, 1, and 2, utilizing a PDR of 0.25. Each backdoor trigger employed a randomly generated mask with uniformly distributed values between -0.1 and 0.1, altering pixel values by a maximum of 10%. An example of such a trigger is shown in Fig. 11. Our results revealed a clear trend indicating limited effectiveness with the mean ASR ranging from 0.15 to 0.3 for the three backdoors. Mask triggers, providing more subtle and widespread image alterations compared to the precise and localized effects of pixel triggers, yielded weaker backdoors, especially when multiple backdoors were embedded. As a result, while our approach isn't restricted to a specific backdoor type, we adhere to using pixel triggers in our experiments due to their higher efficacy.

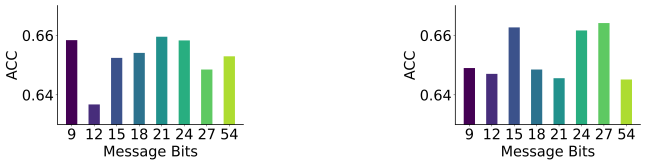
## D.3 Single-Trigger

Below, we provide details of the experiments for STB.

**Random Messages.** To establish a baseline for transmission capacity, we embedded messages with varying length inside the model. Here, we provide the exact messages ranging from 9 bits, corresponding to 3 backdoors (due to the 10 output classes), to 54 bits, which corresponds to 18 backdoors. The messages were generated randomly.

- $M_1$: 101000110 (9)
- $M_2$: 001101010011 (12)
- $M_3$: 000110101010001 (15)
- $M_4$: 111111001001010111 (18)
- $M_5$: 010111000101001011001 (21)
- $M_6$: 110001010000110101000111 (24)
- $M_7$: 101001100110001100110010100 (27)
- $M_8$: 000101101000000100011001010001 001101100101101111000100 (54)

**Covert Channel Performance.** In Tab. 4, we report an extended version of Tab. 1. Fig. 13 then shows the accuracy values for two different PDRs of the same experiments, highlighting, that no significant difference can be found for different message lengths. Note,




**(a) PDR 0.05** **(b) PDR 0.15**

**Figure 13: Impact on the mean test accuracy (ACC) over 100 epochs with messages with lengths ranging from 9 to 54 bits.**

note that the y-axis scale ranges from 63% to 67% instead of 0% to 100% in Fig. 13.

**Robustness & Reliability.** Due to space regulations in the main part of the paper, we provide Fig. 14 here. It shows the changes in BERs for PDRs 0.05, 0.10, 0.15, and 0.20.

**Different Evaluation Sample.** We studied the influence of different evaluation samples on the BER. We conducted an assessment of the BERs across all samples in the train set on a label-wise basis. The results are visualized in Fig. 15.

**Error Correction.** For STB, Hamming Codes, a well-established error-detection and correction mechanism can be applied. Hamming Codes entail the incorporation of parity bits, which are appended to the data at the receiver's end. These parity bits serve the purpose of rectifying single-bit errors during the decoding process at the receiver's end. To balance between communication overhead and error detection capability, we have opted for the (7,4) Hamming code. This specific Hamming code configuration encodes 4 data bits with the addition of 3 parity bits, culminating in a code word of a total length of 7 bits.[11] We conducted an experiment to evaluate the trade-off between the benefits of error correction and the additional overhead introduced by parity bits. In this experiment, we injected a 12-bit message (011100110000) into the CIFAR-10 [25] dataset and trained a ResNet-18 [17] model for 100 epochs with the PDR set to 0.05. The message was encoded using the (7,3) Hamming Code, resulting in a message length of 21 bits. For comparison, we injected the encoded message without Hamming Code into another model. The results, as depicted in Fig. 16, revealed that the average BER was 0.4058 (5 out of 12 bits) for the model without error correction and 0.4813 (10 out of 21 bits) for the model with error correction. This demonstrates that applying the Hamming error correction approach doubled the number of bit errors (+100%) while only increasing the message length by 75%. Based on these findings, we conclude that implementing error correction for the STB approach is not advantageous, as it slightly increases the error rate while also increasing the message length.

**Application Independence.** In Sect. 5.3, we report results about the application independence of STB. However, due to space limitations, we visualize the results here. Fig. 17 visualizes the label-wise BERs of an experiment showing the performance of STB for a SqueezeNet [20] model architecture as well as for the STL-10 [6] dataset.

[11] It is important to note that this approach is not constrained to encoding only 4 bits; it can effectively handle multiples of 4 bits. In cases where the message length is not divisible by 4, we implement zero padding to ensure compatibility with this encoding scheme.

**Table 4: Mean Backdoor ASRs and mean model accuracies (rounded) in percent for different message lengths and PDRs for STB.**

| | Injected Message | | Accuracies in % | | Rounded ASRs for Backdoors in % | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Length | PDR | Test | Train | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | B10 | B11 | B12 | B13 | B14 | B15 | B16 | B17 | B18 |
| 0: | - | - | 71 | 98 | | | | | | | | | | | | | | | | | | |
| 1: | 9 | 0.05 | 66 | 90 | 77 | 68 | 60 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 2: | | 0.1 | 67 | 91 | 87 | 84 | 83 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 3: | | 0.15 | 65 | 89 | 88 | 88 | 86 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 4: | | 0.2 | 65 | 89 | 89 | 91 | 91 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 5: | 12 | 0.05 | 64 | 88 | 77 | 73 | 80 | 71 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 6: | | 0.1 | 67 | 91 | 87 | 89 | 88 | 88 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 7: | | 0.15 | 65 | 89 | 90 | 89 | 89 | 87 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 8: | | 0.2 | 64 | 89 | 91 | 87 | 89 | 87 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 9: | 15 | 0.05 | 65 | 89 | 79 | 70 | 77 | 77 | 67 | - | - | - | - | - | - | - | | - - | - | - | - | - |
| 10: | | 0.1 | 66 | 91 | 90 | 87 | 86 | 89 | 79 | - | - | - | - | - | - | - | - | - | - | | - - | - |
| 11: | | 0.15 | 66 | 92 | 88 | 87 | 86 | 87 | 85 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 12: | | 0.2 | 64 | 90 | 91 | 90 | 88 | 91 | 90 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 13: | 18 | 0.05 | 65 | 89 | 65 | 31 | 22 | 50 | 74 | 56 | - | - | - | - | - | - | - | - | - | - | - | - |
| 14: | | 0.1 | 66 | 90 | 73 | 63 | 66 | 72 | 85 | 75 | - | - | - | - | - | - | - | - | - | - | - | - |
| 15: | | 0.15 | 65 | 90 | 83 | 73 | 71 | 79 | 86 | 82 | - | - | - | - | - | - | - | - | - | - | - | - |
| 16: | | 0.2 | 67 | 91 | 79 | 73 | 79 | 88 | 89 | 85 | - | - | - | - | - | - | - | - | - | - | - | - |
| 17: | 21 | 0.05 | 66 | 90 | 89 | 86 | 75 | 83 | 49 | 85 | 58 | - | - | - | - | - | - | - | - | - | - | - |
| 18: | | 0.1 | 66 | 90 | 88 | 87 | 85 | 85 | 79 | 88 | 77 | - | - | - | - | - | - | - | - | - | - | - |
| 19: | | 0.15 | 65 | 91 | 90 | 92 | 90 | 89 | 83 | 89 | 80 | - | - | - | - | - | - | - | - | - | - | - |
| 20: | | 0.2 | 66 | 92 | 89 | 91 | 87 | 89 | 88 | 91 | 83 | - | - | - | - | - | - | - | - | - | - | - |
| 21: | 24 | 0.05 | 66 | 90 | 75 | 78 | 80 | 61 | 63 | 79 | 60 | 76 | - | - | - | - | - | - | - | - | - | - |
| 22: | | 0.1 | 65 | 90 | 87 | 90 | 86 | 80 | 83 | 86 | 80 | 81 | - | - | - | - | - | - | - | - | - | - |
| 23: | | 0.15 | 66 | 91 | 86 | 90 | 88 | 84 | 84 | 90 | 79 | 87 | - | - | - | - | - | - | - | - | - | - |
| 24: | | 0.2 | 66 | 92 | 86 | 89 | 90 | 84 | 86 | 90 | 87 | 89 | - | - | - | - | - | - | - | - | - | - |
| 25: | 27 | 0.05 | 65 | 89 | 83 | 59 | 66 | 62 | 59 | 55 | 62 | 76 | 61 | - | - | - | - | - | - | - | - | - |
| 26: | | 0.1 | 65 | 89 | 87 | 74 | 80 | 69 | 68 | 69 | 70 | 81 | 69 | - | - | - | - | - | - | - | - | - |
| 27: | | 0.15 | 66 | 93 | 88 | 87 | 85 | 81 | 83 | 81 | 82 | 86 | 81 | - | - | - | - | - | - | - | - | - |
| 28: | | 0.2 | 65 | 90 | 90 | 91 | 87 | 87 | 88 | 86 | 85 | 89 | 87 | - | - | - | - | - | - | - | - | - |
| 29: | 54 | 0.05 | 65 | 89 | 81 | 63 | 69 | 53 | 72 | 64 | 81 | 76 | 75 | 59 | 68 | 75 | 48 | 48 | 66 | 66 | 48 | 72 |
| 30: | | 0.1 | 65 | 90 | 89 | 76 | 79 | 86 | 88 | 80 | 87 | 86 | 86 | 76 | 87 | 83 | 60 | 57 | 81 | 83 | 74 | 79 |
| 31: | | 0.15 | 65 | 90 | 91 | 82 | 83 | 91 | 94 | 88 | 90 | 90 | 89 | 89 | 89 | 83 | 75 | 69 | 81 | 83 | 85 | 89 |
| 32: | | 0.20 | 64 | 90 | 92 | 83 | 81 | 90 | 92 | 88 | 90 | 93 | 90 | 90 | 92 | 82 | 73 | 74 | 82 | 84 | 85 | 86 |

**Table 5: Mean Backdoor ASRs and model accuracies (rounded) in percent for different message lengths and PDRs for MTB.**

| | Injected Message | | Accuracies in % | | Rounded ASRs for Backdoors in % | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Length | PDR | Test | Train | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | B10 | B11 | B12 | B13 | B14 | B15 | B16 | B17 | B18 | B19 |
| 0: | - | - | 71 | 98 | | | | | | | | | | | | | | | | | | | |
| 1: | 9 | 0.05 | 66 | 89 | 56 | 13 | 11 | 54 | 11 | 12 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 2: | | 0.10 | 65 | 89 | 88 | 76 | 55 | 84 | 59 | 60 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 3: | | 0.15 | 66 | 89 | 91 | 86 | 82 | 89 | 70 | 80 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 4: | | 0.20 | 65 | 91 | 93 | 90 | 85 | 90 | 80 | 84 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 5: | 12 | 0.05 | 66 | 89 | 19 | 57 | 46 | 28 | 19 | 37 | 60 | 15 | - | - | - | - | - | - | - | - | - | - | - |
| 6: | | 0.10 | 66 | 90 | 58 | 84 | 78 | 77 | 62 | 75 | 75 | 67 | - | - | - | - | - | - | - | - | - | - | - |
| 7: | | 0.15 | 65 | 90 | 71 | 84 | 83 | 77 | 53 | 82 | 85 | 75 | - | - | - | - | - | - | - | - | - | - | - |
| 8: | | 0.20 | 65 | 91 | 83 | 90 | 86 | 83 | 71 | 86 | 89 | 81 | - | - | - | - | - | - | - | - | - | - | - |
| 9: | 15 | 0.05 | 64 | 88 | 24 | 50 | 52 | 12 | 15 | 19 | 50 | 40 | 49 | - | - | - | - | - | - | - | - | - | - |
| 10: | | 0.10 | 66 | 91 | 58 | 77 | 72 | 53 | 49 | 48 | 66 | 65 | 69 | - | - | - | - | - | - | - | - | - | - |
| 11: | | 0.15 | 66 | 92 | 64 | 89 | 84 | 63 | 69 | 62 | 79 | 81 | 80 | - | - | - | - | - | - | - | - | - | - |
| 12: | | 0.20 | 65 | 91 | 79 | 91 | 85 | 75 | 74 | 63 | 82 | 87 | 81 | - | - | - | - | - | - | - | - | - | - |
| 13: | 18 | 0.05 | 66 | 90 | 12 | 14 | 21 | 42 | 16 | 10 | 11 | 16 | 14 | - | - | - | - | - | - | - | - | - | - |
| 14: | | 0.10 | 66 | 88 | 55 | 67 | 65 | 74 | 69 | 31 | 42 | 38 | 49 | - | - | - | - | - | - | - | - | - | - |
| 15: | | 0.15 | 66 | 90 | 71 | 85 | 73 | 77 | 87 | 65 | 55 | 59 | 77 | - | - | - | - | - | - | - | - | - | - |
| 16: | | 0.20 | 65 | 89 | 74 | 81 | 86 | 84 | 85 | 61 | 62 | 72 | 79 | - | - | - | - | - | - | - | - | - | - |
| 17: | 21 | 0.05 | 66 | 90 | 29 | 16 | 11 | 11 | 53 | 10 | 31 | 12 | 25 | 42 | 16 | - | - | - | - | - | - | - | - |
| 18: | | 0.10 | 67 | 91 | 63 | 51 | 28 | 43 | 73 | 19 | 63 | 33 | 45 | 69 | 59 | - | - | - | - | - | - | - | - |
| 19: | | 0.15 | 64 | 88 | 77 | 74 | 69 | 54 | 74 | 48 | 81 | 69 | 62 | 87 | 77 | - | - | - | - | - | - | - | - |
| 20: | | 0.20 | 65 | 90 | 86 | 79 | 70 | 79 | 85 | 60 | 85 | 71 | 68 | 85 | 84 | - | - | - | - | - | - | - | - |
| 21: | 24 | 0.05 | 65 | 87 | 78 | 38 | 48 | 37 | 29 | 22 | 17 | 31 | 35 | 27 | - | - | - | - | - | - | - | - | - |
| 22: | | 0.10 | 65 | 88 | 85 | 66 | 64 | 62 | 72 | 43 | 54 | 57 | 59 | 69 | - | - | - | - | - | - | - | - | - |
| 23: | | 0.15 | 66 | 92 | 88 | 80 | 80 | 74 | 67 | 74 | 64 | 73 | 77 | 74 | - | - | - | - | - | - | - | - | - |
| 24: | | 0.20 | 64 | 90 | 90 | 88 | 84 | 87 | 79 | 88 | 81 | 86 | 87 | 88 | - | - | - | - | - | - | - | - | - |
| 25: | 27 | 0.05 | 65 | 88 | 13 | 22 | 9 | 14 | 20 | 65 | 23 | 29 | 10 | 12 | 12 | - | - | - | - | - | - | - | - |
| 26: | | 0.10 | 66 | 90 | 68 | 65 | 44 | 67 | 48 | 77 | 55 | 54 | 58 | 43 | 61 | - | - | - | - | - | - | - | - |
| 27: | | 0.15 | 64 | 89 | 74 | 79 | 81 | 76 | 62 | 84 | 72 | 64 | 79 | 61 | 78 | - | - | - | - | - | - | - | - |
| 28: | | 0.20 | 65 | 90 | 78 | 84 | 73 | 79 | 73 | 84 | 73 | 74 | 76 | 68 | 82 | - | - | - | - | - | - | - | - |
| 29: | 54 | 0.05 | 66 | 90 | 66 | 25 | 52 | 24 | 11 | 37 | 32 | 28 | 13 | 19 | 12 | 12 | 32 | 12 | 15 | 37 | 16 | 13 | 58 |
| 30: | | 0.10 | 66 | 90 | 79 | 44 | 73 | 52 | 13 | 77 | 63 | 48 | 19 | 44 | 41 | 15 | 53 | 18 | 21 | 59 | 43 | 38 | 81 |
| 31: | | 0.15 | 65 | 90 | 89 | 63 | 83 | 85 | 20 | 85 | 81 | 64 | 51 | 72 | 72 | 26 | 68 | 42 | 33 | 77 | 70 | 65 | 91 |
| 32: | | 0.20 | 63 | 89 | 90 | 77 | 85 | 83 | 58 | 88 | 82 | 76 | 71 | 78 | 75 | 64 | 76 | 71 | 74 | 82 | 80 | 74 | 91 |


**Figure 14: BERs for one evaluation sample over 100 epochs, with PDRs 0.05, 0.10, 0.15, and 0.20 from left to right.**

## D.4 Multi-Trigger

**Different Messages.** For MTB, we provide the same table as for STB, showing mean backdoor ASRs and model accuracies. Tab. 5 reports the results for the same messages as reported for STB in Tab. 4. The most significant difference that we can observe ist, that for low PDRs, e.g., PDR of 0.05, we get low ASRs for the backdoors.

**Table 6: Injected ITA-2 [21] encoded messages.**

| Text | Bit Encoding | Length (in bits) |
|---|---|---|
| HELLOWORLD | 1001011000001001010010001111001000110101001001100010 | 51 |
| JASPERSTANG | 1110101100010100011011000001010101000000111000001 1001011 | 56 |
| TORSTENKRAUSS | 1000010001101010101000000110000001101111001010110001110010100100100 | 66 |

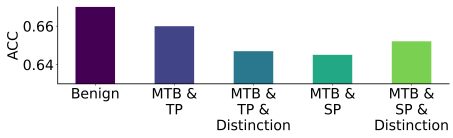
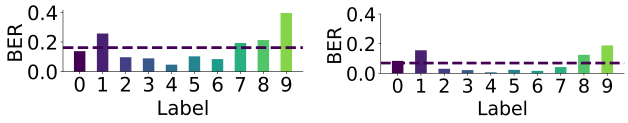



**Figure 18: Mean accuracy on the test set (ACC) for a 66-bit message embedded on a ResNet-18 [17] trained on CIFAR-10 [25]. The figure shows the influence of TP and SP location patterns and the "color distinction" dataset poisoning strategy for MTB.**




**(a) PDR 0.05** **(b) PDR 0.15**

**Figure 15: Training dataset BERs per ground-truth label for a message length of 54 (dashed line indicates mean value).**


**(a) No Hamming Code.** **(b) Hamming Code.**

**Figure 16: Impact of Hamming Code Error Correction for a message of length 12 and a PDR of 0.05.**


**(a) SqueezNet [20]** **(b) PDR 0.15**

**Figure 17: (a) Label-wise BERs of SqueezeNet [20] trained on CIFAR-10 [25]. A 54-bit message was embedded with a PDR of 0.15. The dashed line indicates the mean BER over all labels of 0.029. (b) ResNet-18 [17] trained on STL-10 [6]. The same message was embedded with a PDR of 0.20 resulting in a mean BER of 0.1126.**

However, for PDR of 0.15 and 0.20, we can report high ASRs similar to the STB approach.

**Text Messages.** To test the MTB approach, we report the results for the longest out of three tested messages in the main part of the paper. However, all three messages yielded similar results. Specifically, we injected the ITA-2 [21] encoded messages in Tab. 6.

**Fidelity.** In Fig. 18, we visualize the reported main task test accuracy values reported in Sect. 5.4 showing the influence of MTB compared to a clean model.