Verify your Labels! Trustworthy Predictions and Datasets via Confidence Scores

Torsten Krauß
University of Würzburg

Jasper Stang
University of Würzburg

Alexandra Dmitrienko
University of Würzburg

Abstract

Machine learning is a rapidly evolving technology with manifold benefits. At its core lies the mapping between samples and corresponding target labels (SL-Mappings). Such mappings can originate from labeled dataset samples or from prediction generated during model inference. The correctness of SL-Mappings is crucial, both during training and for model predictions, especially when considering poisoning attacks.

Existing standalone works from the dataset cleaning and prediction confidence scoring domains lack a dual-use tool offering an SL-Mappings score, which is impractical. Moreover, these works have drawbacks, e.g., dependence on specific model architectures and reliance on large datasets, which may not be accessible, or lack a meaningful confidence score.

In this paper, we introduce LabelTrust, a versatile tool designed to generate confidence scores for SL-Mappings. We propose pipelines facilitating dataset cleaning and confidence scoring, mitigating the limitations of existing standalone approaches from each domain. Thereby, LabelTrust leverages a Siamese network trained via few-shot learning, requiring minimal clean samples and is agnostic to datasets and model architectures. We demonstrate LabelTrust's efficacy in detecting poisoning attacks within samples and predictions alike, with a modest one-time training overhead of 34.56 seconds and an evaluation time of less than 1 second per SL-Mapping.

1 Introduction

Machine learning (ML) is a rapidly evolving technology with many benefits, e.g., in task automation [6], decision-making [5], and recommendation systems [8]. These systems rely on large datasets, often employing Deep Neural Networks for pattern extraction during training. Supervised learning [31], the prevalent ML approach, involves data samples labeled with ground truth, allowing models to learn patterns for accurate predictions. Following training, models utilize these learned patterns to infer predictions for new data.

At the heart of an ML system lies the mapping between a sample and its label, which we term Sample Label Mapping (SL-Mapping). The correctness of this mapping is crucial regardless of its origin, e.g., from domain experts in the dataset labeling phase or from predictions by a trained model during inference. Quality in both, the sample and the assigned target label is imperative. For labeled train datasets, the efficacy of a trained model depends heavily on the quality of samples and assigned labels. Conversely, during model inference, trustworthy and accurate predictions are important, as erroneous predictions could have significant consequences, particularly in applications like autonomous driving systems [6].

Problem Statement. The challenge of SL-Mappings, prevalent in training datasets but also during model inference, stems from potential mismatches, e.g., false labels. These discrepancies may arise from various sources, including unintentional errors and deliberate false mappings. In labeled datasets, errors can occur due to mistakes by experts or manipulation by adversaries, who may poison benign datasets with incorrect SL-Mappings [7, 13, 26, 27, 40]. Similarly, during model inference, models may exhibit low prediction performance or be trained on poisoned datasets, leading to erroneous predictions. Regardless of the source and effects of these inaccuracies, there is a clear need for a system capable of detecting invalid or erroneous mappings and providing corresponding notifications. Existing works tackling similar challenges primarily fall within dataset cleaning or prediction confidence scoring.

Existing SL-Mapping Verification Methods. Dataset cleaning [12, 17, 33, 35–38, 47] aims to identify low-quality or poisoned samples within a dataset, separating them from clean data to ensure untainted training. However, existing approaches [12, 35, 38] often involve intensive computations on the entire dataset, such as training auxiliary models, or rely on a sizable fraction of clean samples, which may not always be available. The closest work to ours from this domain, Huang *et al.* [17], relies on a specific model architecture and is designed for a particular pre-trained model instance. Hence, if the existing trained model is upgraded, e.g., with a more efficient model architecture, the system must be retrained to accommodate the new architecture. Retraining requires 1%

clean data, which can pose challenges with larger datasets as obtaining this portion may require substantial human effort.

Confidence scoring [9, 19, 28] evaluates the trustworthiness of predictions made by pre-trained models. However, existing works [9, 19, 28] do not consider poisoning attacks. Moreover, these methods depend on the entire untrusted train set, making it challenging to ensure the absence of poisonings, potentially compromising their reliability. The closest related work, Corbière *et al.* [9], employs a secondary model to generate confidence scores for predictions and relies on intermediate layer outputs from a pre-trained primary model, thereby being dependent on a specific trained model.

In general those standalone methods are specifically tailored to one specific domain, necessitating the implementation of separate tools with distinct assumptions, prerequisites, and performance characteristics in different use cases. Moreover, dataset cleaning may not achieve complete effectiveness, potentially leaving leftover poisoned samples in the cleaned training dataset, thus underscoring the importance of generating trust scores for predictions during inference.

Desired System. Hence, there is a need for a system capable of both, dataset cleaning and confidence scoring. As both scenarios evaluate SL-Mappings, the system should be a general tool providing high flexibility in terms of use cases. This system should only rely on a small fraction of clean samples from the train set and should be independent of a specific trained model or it's architectures, while detecting false ST-Mappings and yielding a meaningful confidence score.

Approach. We introduce LabelTrust, a scoring system designed for SL-Mappings, that bridges the gap in the absence of a general tool for SL-Mapping quality assessment. Serving multiple purposes in dataset cleaning and prediction confidence scoring, LabelTrust can be applied to multiple use-cases simultaneously without dual effort. Furthermore, it addresses limitations of existing dataset cleaning and confidence scoring methods, mainly the dependence on specific model instances or architectures as well as the reliance on extensive datasets. LabelTrust yields a meaningful confidence score for the label associated with a sample, essentially providing a measure of trust for the SL-Mapping. Thereby, LabelTrust is efficient in identifying low-quality samples but also poisonings stemming form, for instance, poisoning attacks.

Contributions. This paper makes the following contributions:

- We propose LabelTrust, a general tool to generate a confidence score for an SL-Mapping. Rooted in a Siamese network [25] architecture, LabelTrust discerns whether two samples originate from the same label class. By employing few-shot learning, LabelTrust only requires a minimal number of clean samples for operation. The system's architecture includes a refeed loop, facilitating continuous performance enhancement.

- We suggest LabelTrust for dataset cleaning to effectively filtering samples from datasets. This use case aids experts in identifying not only poisoned samples but also low-quality or underrepresented ones, significantly reducing the time required for dataset examination.

- We propose using LabelTrust for prediction confidence scoring. LabelTrust provides trust scores for predictions made by trained models during inference, enabling users to determine the reliability of predictions or to initiate further examination by human experts.

- We conduct a comprehensive study demonstrating the robust performance of LabelTrust. Our findings indicate that LabelTrust can detect poisoning attacks in both the dataset cleaning and the confidence scoring domain. Notably, LabelTrust successfully detected four different poisoning attacks and exhibits independence from datasets and model architectures, requiring minimal clean samples, between 2 to 35 samples per label class in this paper. We found that, equipped with LabelTrust, a domain expert only needs to manually inspect around 0.0058% of the data for cleaning 83.73% of a dataset. The runtime overhead of LabelTrust is also manageable, with a one-time effort of 34.56 seconds required to train the Siamese network and a negligible processing time of below 1 second for evaluation of one SL-Mapping.

In summary, this paper presents LabelTrust, a general tool designed to generate confidence scores for SL-Mappings, applicable to both dataset cleaning and prediction confidence assessment. Leveraging few-shot learning with a Siamese network and a limited set of clean samples, LabelTrust's performance can be enhanced over time through a refeed loop. Addressing shortcomings of existing methods, LabelTrust overcomes challenges related to dataset size, model dependencies, lack of meaningful confidence scores, and missing consideration of poisoning attacks.

Outline. In Sect. 2, we first provide background information, before we present our approach in Sect. 3, followed by a evaluation study in Sect. 4. Sect. 5 discusses aspects of LabelTrust and Sect. 6 delimits LabelTrust from related works, before Sect. 7 concludes the paper.

2 Background

This section provides background information that is necessary to understand our approach.

Samples in Machine Learning. In the context of machine learning (ML), a sample, such as an image in an image classification [5] task, is treated as an input to the ML model. For supervised ML [31], a respective dataset comprises samples associated with target labels, representing the ground truth assigned by a domain expert. In this paper, we refer to this assignment as Sample Label Mapping (SL-Mapping). The term

"sample" often implicitly encompasses the associated label in the training context. However, during inference, the sample refers exclusively to the input data (e.g., the image), as the model is expected to generate the corresponding prediction or class label without direct access to the ground truth. Note, that we also refer to the sample with the prediction assigned by the model as an ST-Mapping. It is noteworthy that a sample may not be limited to a single data record; for instance, Siamese networks [25], which we will introduce below, require two data records, e.g., two images, simultaneously, forming a composite sample for the network, as illustrated in Fig. 1.

A dataset typically comprises multiple samples paired with corresponding target labels, serving as the training data for the model. The efficacy of the trained model is heavily dependent on the quality of the dataset [41]. Firstly, the included samples should exhibit high quality, portraying clear scenarios without significant noise obscuring the actual content. Secondly, the dataset should demonstrate diversity, encompassing instances of the same scenario from slightly varied angles or under different circumstances. This diversity facilitates better generalization by enabling the model to extract generalized features rather than memorizing training samples. Additionally, class balance within the dataset [16] is crucial for optimal prediction performance across all classes, as an imbalanced dataset might lead the model to focus on one class and might diminish the model's generalization capabilities. This consideration is particularly pertinent for classes with diverse shapes, viewed as subclasses within one class (e.g., various types of fonts in a written digit dataset). Thus, achieving balance within classes is pivotal for effective generalization. If a sample class or subclass is underrepresented, e.g., the amount of samples for this subclass is lower compared to others, the model might not adequately focus on it, leading to suboptimal generalization. In such cases, it might be beneficial to remove the samples or add more samples of the same (sub)class.

In this paper, our solution can identify erroneous SL-Mappings, originating from intentional poisonings, as well as from low-quality or underrepresented samples.

Dataset Cleaning. An ST-Mapping is an integral component of a labeled dataset created before model training in an offline setting, which is subsequently used for model training. Typically, such datasets are curated by domain experts who carefully examine selected samples and assign labels to each sample individually. This assignment process can be carried out either entirely manually or with the assistance of a machine, such as a rule-based algorithm. The resulting ST-Mapping essentially constitutes a labeled dataset suitable for model training. Dataset cleaning approaches split the dataset into two parts: The clean dataset, comprises unpoisoned samples, and filtered samples that are potentially poisoned. The final clean dataset will then be used for subsequent model training. Labeling mismatches may arise due to unintentional human error, particularly when a domain expert makes an invalid assignment. These mismatches can also be intentional if the domain expert is an adversary. The resulting ST-Mapping within the labeled dataset, manipulated by a malicious expert, can exert adverse effects during the subsequent model training phase. This is a form of poisoning attack, which we discuss in detail below, namely data poisoning, which represents a method to manipulate the model training process. In this work, we propose a general tool for ST-Mapping verification, which can be used for dataset cleaning.

Confidence Scoring. After model training in an online setting, previously unseen data samples can be input into an already trained and deployed model, resulting in corresponding predictions. The consolidation of these samples and their respective predictions once more form an ST-Mapping. A confidence scoring system should be able to provide a confidence score for this mapping. In this context, the confidence score serves as a direct indicator of prediction trustworthiness. This trust score can be interpreted as a measure of reliability for the predictions. During model inference, the accuracy of predictions is contingent upon the quality of the utilized model. Consequently, a model with suboptimal performance, such as one trained on a low-quality dataset, is prone to generating more erroneous predictions compared to a model trained on a substantially diverse and high-quality dataset. False mispredictions due to low model performance are a form of unintentional erroneous SL-Mappings. However, an adversary can also disrupt the training process through model poisoning attacks. In such cases, the adversary aims to create a poisoned model that is subsequently deployed, leading to intentionally false SL-Mappings during inference. Both, intentional and unintentional false mappings should be assigned with a low confidence score, while correct SL-Mappings should yield a high confidence score. In this paper, a general tool for confidence scoring, is developed.

Siamese Networks. A Siamese network [11, 25] represents a model architecture employed for assessing the similarity between two data records, such as two images. Common applications of Siamese networks include image recognition [29], signature verification [2], and face recognition [49], where the objective is to compare the similarity of two records. Contrary to traditional classification networks, which assign samples of single data records to specific labels, a Siamese network produces a similarity score, indicating the similarity between two data records. Typically, one part is the record under examination, and the other one is a record from the claimed associated label to be verified. Thereby, Siamese Networks are based on few-shot learning [48], meaning that only a few high-quality samples per class are necessary for training.

The fundamental concept of a Siamese network is depicted in Fig. 1 and involves guiding the two data records through identical networks, each sharing the same architecture and weights - effectively functioning as "siamese twins". These networks generate compressed representations, so-called embeddings, of the records, essentially forming feature vectors

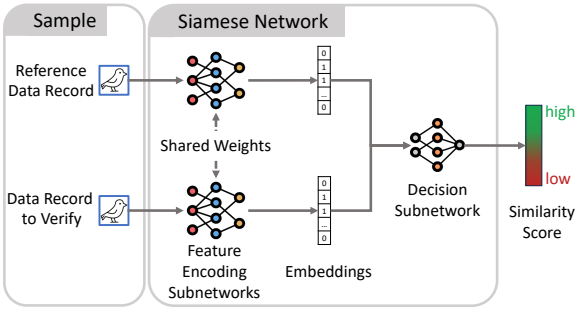

Figure 1: Visualization of a Siamese network.

and are hence named feature encoding subnetworks in Fig. 1. These embeddings then undergo processing in a subsequent decision subnetwork (cf. Fig. 1), which calculates the similarity score between the two embeddings. The decision subnetwork can utilize the embeddings in concatenated form or explore various combinations, including engineered features such as employing a distance metric like cosine similarity between the two vectors. Generally, the final decision network usually comprises a few linear layers, culminating in a singular output value, visualized as similarity score in Fig. 1.

Given that the network produces a similarity score for two data records, its training involves data record pairs. In a typical scenario, various pairs of records from the same target label class and records from different classes are paired together. These pairs are then labeled with 1 or 0, indicating whether they originate from the same class or different classes, essentially forming an input sample for the Siamese network.

For a straightforward binary classification task, a Siamese network can be trained using the Binary Cross Entropy Loss (BCELoss, cf. Eq. 1) processing the last layers output with a sigmoid function beforehand to produce a score between 0 and 1. In the equation, N is the number of samples, y_i is the target label (1 or 0 indicating if the two data records belong together or not) and $p(y_i)$ is the output of the decision network processed by a sigmoid function, e.g., the probability that the sample belongs to the label 1, meaning that both data records are from the same class. Notable alternatives to BCELoss include the Contrastive loss and the Triplet loss [25], which we discuss in App. 7.2.

$$BCE = \frac{1}{N}\sum_{i=1}^{N} y_i \ log(p(y_i)) + (1 \quad y_i) \ log(1 \quad p(y_i)) \quad (1)$$

In this paper, we create a Siamese network on a small examined dataset mainly trained with BCEloss, which serves as a central part of LabelTrust to produce a confidence score for mappings between samples and labels.

Poisoning Attacks. Poisoning attacks are strategic attempts to undermine the predictive performance of an ML model. Such attacks can be untargeted [27, 40] or targeted [7, 13, 26]. Untargeted attacks are designed to diminish the overall model performance, without introducing new behaviors but rather

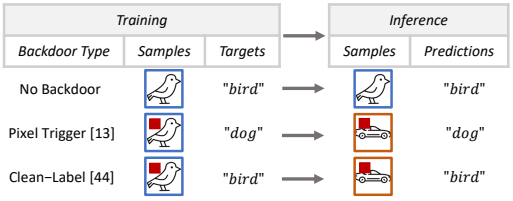

Figure 2: Visualization of different backdoor attacks.

disrupting the model's functionality. On the other hand, targeted attacks, often referred to as backdoors, aim to stealthily introduce hidden adversarial behavior to the model while maintaining the regular model performance. In a targeted attack, a backdoor comprises a trigger and a target class, both selected by the attacker. The trigger is injected into a sample and the label is changed towards the target class, causing the model to associate this trigger with the backdoor's target class, irrespective of the actual content of the sample. Consequently, a trigger placed on a sample from any class during inference would result in misclassification towards the backdoor's target. Triggers can take various forms, such as a pixel pattern [13], as visualized in the second line of Fig. 2, and leads to a classification towards the backdoor's target class ("dog") during inference. Another alternative backdoor would be adding random noise across the entire image [7], to name another example.

As visualized in the third line of Fig. 2, one of these backdoor attacks is a clean-label backdoor [44], wherein the trigger is exclusively placed on samples from the target class, "bird" in this case, and the label itself is not altered. Nevertheless, during training, the model associates the trigger with the target class and subsequently classifies any sample containing the trigger towards that class during inference.

Poisoning attacks can be executed through either data poisoning or model poisoning methods. In data poisoning, the attacker manipulates the dataset used for training by introducing triggers to samples and altering labels. In model poisoning, the adversary interferes with the training process itself, altering training methods, loss functions, or hyperparameters. Additionally, the attacker may manipulate the model parameters directly, either during training or post-training.

This paper proposes a solution for the identification and filtration of poisoned data samples during the dataset cleaning process. Furthermore, the solution is utilized to detect poisoning attacks during the inference phase.

3 Approach

We aim to tackle the challenge inherent in the machine learning domain, where the accurate mapping of a label to a sample (SL-Mapping) is uncertain, irrespective of the method employed for assignment. An incorrect SL-Mapping can either be unintentional or intentional stemming from poisoning

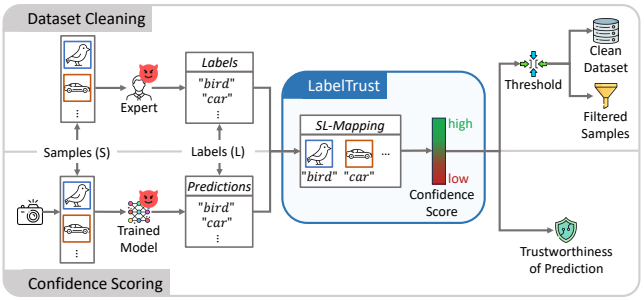

Figure 3: Visualization of the considered scenarios.

attacks, as elaborated in Sect. 2. This uncertainty demands a system that assesses the trustworthiness of SL-Mapping. The proposed system, LabelTrust, as illustrated in the center of Fig. 3, should yield a confidence score for an SL-Mapping, allowing to identify false mappings. This score is relevant in two specific scenarios, which we consider in this paper: Dataset cleaning, as visualized in the upper half of Fig. 3, and confidence scoring on predictions of trained models on unseen data, as visualized in the lower half of Fig. 3.

Application Scenarios. For dataset cleaning, a labeled dataset from an untrusted source can be obtained. Hence, it is unclear if the dataset contains any unintentionally mislabeled or low quality samples[1], or poisoned samples, that were intentionally mislabeled by an adversary to introduce a poisoning attack (cf. Sect. 2). LabelTrust generates confidence scores for each sample, enabling the creation of a "clean" subset for benign model training. In particular, a threshold based on the desired safety level can be applied to the confidence score, thus, identifying and removing potentially poisoned or low-quality samples.

For the confidence scoring setup, a trained and hence untrusted model can either be obtained from third-party sources, or the model can be trained locally on an untrusted dataset. Feeding unseen samples to the trained model yields a prediction, essentially forming an SL-Mappings with the input sample, which can then be assessed. LabelTrust's generated confidence score can then be used directly as a measure of trust for the prediction. In essence, LabelTrust detects intentional mispredictions (backdoors) and unintentional errors due to poor model performance by assigning low trust scores.

Combining both scenarios further enhances trust in the model. A dataset can be cleaned using LabelTrust before training a model. Subsequently, LabelTrust can evaluate the model's predictions during real-time operation, adding another layer of security without necessitating additional effort.

Threat Model. For dataset cleaning, we consider an adversary capable of compromising a dataset intended for training purposes. This adversary holds the capability to implement various poisoning attacks through data poisoning, including targeted poisoning attacks (cf. Sect. 2). In particular, the adversary might make arbitrary modifications to the dataset, e.g., change the samples, alter target labels, or add new samples. Those modifications can either be intentional with the goal of embedding a poisoning attack or unintentional due to human error.[2] To manipulate the predictions during model inference in the domain of confidence scoring, the adversary can generate a poisoned model utilizing dataset poisoning or model poisoning techniques. Such model manipulation takes place before model deployment. Further, a model can also yield false predictions due to a generally low model performance. It is important to note that the execution of the desired system is assumed to take place within a secure environment, thus safeguarding it against any adversarial interactions or interference with the process. As a result, the desired system is considered benign within this secure context.

Below, we describe the core concept of LabelTrust in detail. Sect. 3.1 presents the inner workings of LabelTrust, followed by two application scenarios in Sect. 3.2.

3.1 Details of LabelTrust

The general functionality of LabelTrust is to generate confidence scores for mappings between samples and labels (SL-Mappings) within a machine learning environment. Hence, these mappings, which will be inspected, can originate from either an existing labeled dataset created during a manual labeling process or from predictions made by an already trained and deployed model when provided with an unseen sample. The confidence score produced by LabelTrust serves as an evaluative measure of the legitimacy of the assignment between a sample and its label, facilitating the identification of irregular mappings indicated by low confidence scores.

To derive an SL-Mapping confidence score, LabelTrust leverages the expertise of a domain expert who selects a small clean subset of the dataset. For dataset cleaning, this subset is selected from the entire set of samples to be inspected, while for confidence scoring, the subset is selected from the dataset used for model training. The expert selects a small high-quality subset consisting of x samples for each of the n label classes $(l_1, \ldots, l_n)$, ensuring their unpoisoned nature and correct labeling, effectively forming a small benign SL-Mapping as a ground truth, which we name reference dataset.

These samples are then used to train a Siamese network via few-shot learning. The Siamese network is fed with two samples simultaneously, and trained to discern whether they belong to the same label class. The resulting output of the Siamese network consists of a value between 0 and 1, where 0 indicates a low probability of belonging to the same label class, and 1 indicates a high probability. This probability

[1] Low-quality samples are correctly labeled, but can potentially be misinterpreted, such as a 1 that might look like 7 in MNIST [10].

[2] In the dataset cleaning scenario, our focus does not encompass clean-label backdoor attacks. However, we tackle these attacks in the use case of prediction confidence scoring. This however does not pose a limitation, as our tool is crafted for dual-use in both scenarios.

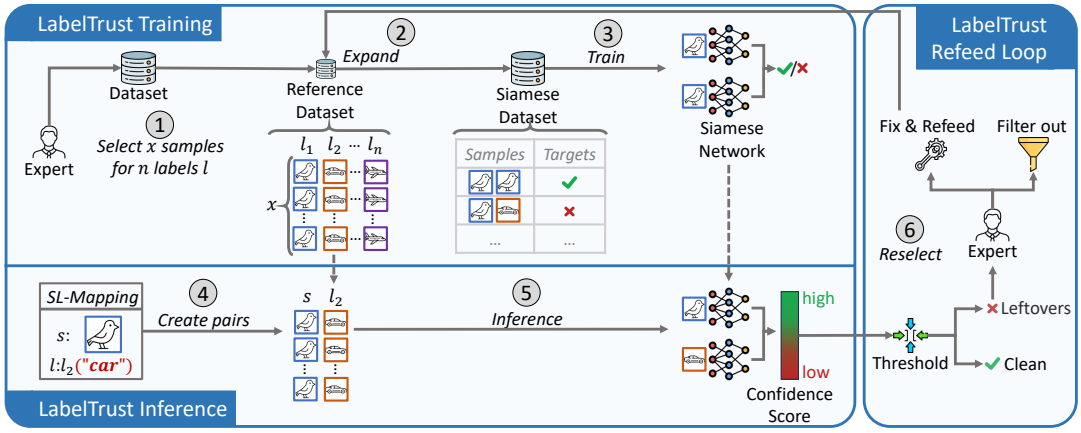

Figure 4: Detailed visualization of the six life cycle steps of LabelTrust.

serves as the confidence score. Consequently, after training on the reference dataset, the Siamese network can verify the labelings of SL-Mappings. Importantly, the resulting decision is grounded in the small reference dataset that was used to train the Siamese network. By adding more clean samples into the reference dataset in a refeed loop over time, the quality of the Siamese network can be increased over time.

Life-Cycle Steps of LabelTrust. Below, we explain the steps of LabelTrust, which are visualized in Fig. 4. The initial three steps, constituting LabelTrust's training phase, must be executed before applying LabelTrust for SL-Mappings analysis. Steps 4 and 5 are the central parts for the generation of confidence scores, and the concluding step 6 establishes a refeed loop designed to enhance the performance of LabelTrust.

1. *Selection of Reference Samples.* The domain expert first selects x samples for each of the n target label classes $(l_1, \ldots, l_n)$ within the dataset. It is imperative that the expert verifies the absence of poison, e.g., backdoor triggers, in these samples, designating them as a small reference dataset consisting of benign data with correct target labels, serving as the ground truth for LabelTrust.

2. *Expansion of Reference Dataset.* The reference dataset undergoes expansion to construct a Siamese dataset, enabling the training of a Siamese network. To construct this dataset, each sample is comprised of two instances from the reference dataset and a corresponding label, signifying whether the two samples originate from the same class within the reference datasets. For example, in Fig. 4, the first sample consists of two blue images from the l_1 class ("bird") and is thus positively labeled as 1 indicated with a green checkmark. Conversely, the second sample contains a blue sample from l_1 and a brown sample from l_2, resulting in a negative label of 0 visualized as a red cross.

3. *Training of Siamese Network.* The Siamese dataset is used to train a Siamese Network. The network learns to generate meaningful representations (embeddings) for both inputs and subsequently learns to classify these embeddings toward the positive or negative label, reflecting the similarity of both inputs. After this training step, LabelTrust can be used to evaluate unseen SL-Mappings.

4. *Construction of Valid Input for Siamese Network.* To assess the quality of an SL-Mapping, consisting of a sample s and a label l, LabelTrust constructs valid inputs for the Siamese network. This involves pairing the x samples from the label class of the SL-Mapping (e.g., l_2 in Fig. 4) from the reference dataset with the input sample s. These pairs are similar to the ones in the Siamese dataset from step 2. With this pairing, the Siamese network can validate if the unseen sample s is similar to any of the ground truth samples from the predicted class of s.

5. *Calculation of Confidence Score.* The constructed pairs are fed into the Siamese network trained in step 3, yielding a confidence score for each pair. A high score indicates similarity to the reference dataset, while a low score suggests that the SL-Mapping deviates unusually based on the reference dataset.[3] The final score is based on averaging the confidence scores of all evaluated pairs.[4]

6. *Reefeed Loop.* By applying a threshold to the final confidence score, all tested SL-Mappings can be categorized into clean samples and leftovers, stemming from potentially poisoned, underrepresented, or low-quality samples. A high threshold categorizes samples as clean only if LabelTrust is highly confident in their cleanliness. Conversely, a low threshold can identify samples likely

[3] A high-quality SL-Mappings has high similarities to samples of the respective class within this reference dataset, while low-quality SL-Mappings are not contained in the reference dataset.

[4] If samples within one class significantly differ, we introduce new (sub)classes (cf. Sect. 5). Hence, outliers are very unlikely to occur.

to be undesired, such as potentially poisoned ones.[5] Regardless of the application scenario, a domain expert can then inspect the leftovers.The expert can refeed clean samples from the leftovers, that were not considered in the ground truth so far into the reference dataset and rerun the whole process, including Siamese model training and confidence scoring of the whole dataset. Our intuition, which we verify in Sect. 4.3 is that the leftovers will be iteratively reduced while maintaining a high detection rate of poisonings and simultaneously reducing the number of poisonings that are falsely identified as clean, essentially increasing the performance of LabelTrust over time.

3.2 Application Scenarios

In the following, we will elaborate on the practical applications of LabelTrust within two distinct scenarios: dataset cleaning and the generation of a prediction confidence scores.

Dataset Cleaning. For a manual dataset-cleaning procedure before model training (cf. Sect. 2), the conventional approach would necessitate exhaustive individual inspection of all samples by a domain expert. Leveraging LabelTrust, the expert can optimize this process by selecting a small representative subset of samples for each label class as reference dataset (step 1 in Fig. 4). Subsequently, the Siamese network trained on these samples can be employed to check all samples from the original dataset. Clean samples will exhibit high similarity to the small ground-truth in the reference dataset.

Precisely, intentionally mislabeled samples, e.g, samples with a pixel trigger [13], will not be classified toward the attacker-chosen false label class by the Siamese Network, but towards the benign class and hence yield a low confidence score on the assigned false label. Similarly, unintentionally mislabeled data, e.g., low-quality samples, will also yield a low confidence score on the assigned label, as the samples differ from the ground truth in the reference dataset. For both, intentional and unintentional mislabels, the low confidence score yielded by LabelTrust allows sample filtering, essentially also detecting poisoning attacks. Unpoisoned filtered samples are either mislabeled and thus undesirable in the dataset or can be added to the reference dataset (step 6 in Fig. 4) as they provide valuable insights that have not been represented by the ground truth yet. Thus, the domain expert can resample benign sets from the filtered irregular mappings (leftovers in Fig. 4), and refeed them into the reference dataset (step 6 in Fig. 4). By iteratively repeating this process, e.g., performing multiple dataset cleaning rounds, LabelTrust results in a significant reduction in the number of samples requiring manual verification, which we show in Sect. 4.3.

Noteworthy, in a new dataset cleaning round, all samples are reevaluated by the new Siamese network, instead of just analyzing the leftovers. The reason for this design choice is, that the Siamese network's capability to distinguish clean samples will increase with further rounds. Thus, samples that might have been falsely categorized as clean in the first round will be correctly classified as leftover in subsequent rounds due to increased knowledge. Hence, we assume that the networks's increased capabilities will diminish the size of the leftover over time. We confirm this assumption through empirical evaluation in Sect. 4.3.

The dataset cleaning process can be concluded either when the expert has inspected all samples, discarding or refeeding them, or when a predetermined fraction of filtered samples is reached, that attains a high, yet poison-free model performance, reflecting the compromise the domain expert is willing to make for the sake of a clean dataset.

Confidence Scoring. The same procedure of LabelTrust is applicable for confidence scoring of predictions of a primary pre-trained model. In this scenario, the labels for the samples are obtained by using the samples' prediction yielded by the pre-trained model. The reference dataset can be constructed in two ways: Either access to the trainset is available, e.g., model creators provide LabelTrust. Alternatively, model creators can provide small clean datasets. If no data access is given, the domain expert can observe inference samples and construct the reference dataset (via refeed loop). Hence, LabelTrust can work without access to trainsets. Irregular mappings, essentially arising from mispredictions of the primary model, may stem from either a low-performing or poisoned primary model. Consequently, the confidence score derived from the Siamese network, which is based on the clean reference dataset used to train LabelTrust's Siamese network, can provide insights into the trustworthiness of the primary model's predictions. For samples dissimilar to reference-dataset samples, the confidence score is (and should be) low. Such samples are validated by the domain expert and can be added to the reference dataset via refeed loop. The Siamese network does not achieve the same classification performance as the primary model trained on a large, untrusted dataset. Hence it cannot duplicate the classification task but can asses the primary model's predictions. This advantage comes from training the Siamese network on a smaller, carefully curated reference dataset, free from poisoned data. Hence, both models, primary model and Siamese network have different purposes and complement each other.

The yielded confidence score enables an user to decide if a model prediction is trustworthy. For a low score the user should verify the prediction, while for a high score, the system's prediction is confirmed. Furthermore, the performance of LabelTrust can also be improved in the confidence scoring setting, as unseen new samples could be integrated into the reference dataset after careful inspection by a domain expert

[5] LabelTrust doesn't rely on fixed thresholds. A high threshold will identify very trustworthy mappings while a low threshold will identify untrustworthy mappings in the same setting. The threshold is selected by the user depending on his goal and not depending on the scenario itself, e.g., model architecture or dataset.

(step 6 in Fig. 4). In particular, samples that received low confidence scores are considered most useful as they differ from the existing ground truth and lead to a more diverse Siamese training dataset. After the integration of these samples, a new Siamese model could be trained that replaces the existing one. Hence, similar to dataset cleaning, LabelTrust could improve iteratively over time with the help of a domain expert.

Summary. LabelTrust serves as a versatile tool for confidence score generation that assesses the legitimacy of a label assigned to a sample, relying on a small set of thoroughly examined samples by an expert. The system is based on few-shot learning, specifically using a Siamese network. A high confidence score is indicative of a sample's similarity to an already examined sample and its correct label assignment. Conversely, if a sample falls within a subset that has not yet undergone expert examination, and hence is potentially poisoned or of low quality, LabelTrust will yield a low confidence score. While LabelTrust is a general tool applicable to all use cases with the need for SL-Mapping evaluation, we show its utilization for dataset cleaning and prediction confidence scoring. LabelTrust's performance is iteratively enhanced by refeeding negatively evaluated samples to the small reference dataset and retraining the Siamese network. This process ensures the continual improvement of LabelTrust over time.

4 Evaluation

Datasets & Model Architectures. We use common datasets mainly focusing on image classification with MNIST [10], Fashion-MNIST (FMNIST) [50], and Intel Image Classification (IIC) [3] trained on the ResNet-18 [15] model architecture. The architecture of the feature encoding subnetwork of the Siamese network is identical to the ResNet-18 [15], however, the last layer is removed as the network should not yield a prediction but an embedding of size 256. Further, the first layer is adjusted depending on the utilized dataset. We added a decision network consisting of two linear layers equipped with a ReLU [1] activation function. Furthermore, we evaluated a second Siamese architecture that uses a smaller CNN model as feature encoding subnetwork combined with the same decision network. This CNN comprises two consecutive convolutional layers, each followed by a batch normalization layer and a ReLU [1] activation function, as well as a 2D pooling layer. The experimental setup including hardware specifications is reported in App. 7.1.

4.1 General Functionality

First, we show the general functionality of LabelTrust's core, the Siamese network. Therefore, we use MNIST [10] and the ResNet-18 [15] architecture for the Siamese network.

Siamese Training. In step 1 of LabelTrust (cf. Sect. 3.1 and Fig. 4), involving the selection of reference samples, we randomly extract $x = [2, 5, 10, 15, 20]$ samples for each of the $n = 10$ classes from the MNIST [10] train dataset, forming a reference dataset comprising 20, 50, 100, 150, and 200 samples for LabelTrust. To give an impression of the reference dataset, we visualize the samples for $x = 10$ in App. 7.4. Opposed to the intended use-case of experts selecting high-quality samples, we select correctly labeled samples at random to showcase the behavior under a semi-optimal expert. For the subsequent expansion of the reference dataset in step 2, we utilize these samples to generate a Siamese dataset mirroring the size of the original MNIST [10] dataset, which comprises 60,000 samples. To achieve this, we repeat the following process 60,000 times: We iteratively generate a positive and negative pair. For a positive pair, we randomly select a class and one of the x samples of this class and pair it with a different random sample from the same class. For a negative pair, we chose two different random classes and one sample out of the x samples from each class to generate the pair. For both, a positive and a negative pair, those two selected samples serve as data records to form a new sample for the Siamese network (cf. Sect. 2). Each new sample, composed of two data records, is assigned a target label of 1 if the records originate from the same class and 0 otherwise. Subsequently, this Siamese dataset is employed in step 3 of LabelTrust to train the Siamese network. For training, we employ the BCELoss (cf. Sect. 2), an Adadelt [52] optimizer with learning rate 1.0, and train one epoch with a batch size of 64.

Evaluation Method. We use three distinct test sets with unseen data to evaluate the performance of the Siamese network:

1. *ACC-Testset.* Utilizing the MNIST [10] test set, which are unseen data for the Siamese network, we replicate the method outlined in LabelTrust's step 2 to construct a Siamese test dataset. This test dataset enables us to derive a comprehensive measure of model performance, quantified by accuracy (ACC). We present the ACC across varying confidence score thresholds of 0.99, 0.5, and 0.01. A high threshold of 0.99 allows to identify the samples above 0.99, which have a very high similarity to the reference dataset and hence are most likely clean. A low threshold of 0.01 allows to identify the samples below 0.01, which yield minimal similarity to the reference dataset and hence are not clean with a high probability. The threshold of 0.5 reflects the most balanced trade-off between both settings. While the threshold can be arbitrarily chosen, we showcase the results for the three thresholds, as most application scenarios would most likely use one of these three thresholds.

2. *Unpoisoned-Testset.* Leveraging all samples from the MNIST [10] test set as SL-Mappings for verification, we expect the Siamese network to accurately assess the assigned target labels of evaluation samples. We report the ACC, ideally high, along with the fraction of SL-

Table 1: Performance of the Siamese network for MNIST [10] trained on ResNet-18 [15].

	Poison Type	x	**ACC-Testset** with confidence threshold...			**Unpoisoned-Testset**, with confidence threshold...						**Poison-Testset**			
						0.99		**0.5**		**0.01**		**AVG**	**TRR**, with threshold...		
			0.99	**0.5**	**0.01**	**ACC**	**FRR**	**ACC**	**FRR**	**ACC**	**FRR**	**Score**	**0.99**	**0.5**	**0.01**
(1)	Pixel [13]	2	59.56	69.65	74.45	92.52	72.98	92.34	55.16	89.60	29.90	0.0088	100.00	100.00	92.04
(2)	Pixel [13]	5	72.44	79.53	81.37	95.30	41.03	95.03	28.83	92.10	19.45	0.0074	99.75	99.31	98.43
(3)	Pixel [13]	10	75.50	82.43	85.03	96.29	32.07	96.18	21.45	94.01	13.84	0.0042	99.93	99.61	98.69
(4)	Pixel [13]	15	80.88	88.65	91.52	97.46	22.80	97.82	12.25	96.26	6.39	0.0047	99.87	99.59	98.39
(5)	Pixel [13]	20	81.74	89.17	91.91	97.36	22.94	97.74	12.88	96.52	6.59	0.0010	99.95	99.90	99.68
(6)	Blend [7]	10	75.50	82.43	85.03	96.29	32.07	96.18	21.45	94.01	13.84	0.0013	99.86	98.81	95.20
(7)	Clean-Label [44]	10	75.50	82.43	85.03	96.29	32.07	96.18	21.45	94.01	13.84	0.0041	99.94	99.61	98.69
(8)	Random-Label [4,21]	10	75.50	82.43	85.03	96.29	32.07	96.18	21.45	94.01	13.84	0.0179	99.52	98.37	95.32

Mappings falsely assigned with a confidence score below the threshold, indicated by the False-Rejection-Rate (FRR), again using thresholds of 0.99, 0.5, and 0.01.

3. *Poison-Testset.* Using the MNIST [10] test set, we introduce backdoors into each sample, for instance a pixel trigger [13], and subsequently verify the resultant SL-Mappings. In an optimal scenario, the Siamese network should assign low confidence scores to all poisoned samples. We present the average score over all samples in the test set and the fraction of poisoned samples with a confidence score above the threshold, denoted by the True-Rejection-Rate (TRR), again employing thresholds of 0.99, 0.5, and 0.01.

Results. The results of the experiments are listed in Tab. 1. Lines (1) to (5) showcase the results when utilizing the pixel trigger [13]. We begin with $x = 2$ samples from each class in line (1) and progressively increase the number of samples in the reference dataset to $x = 20$ in line (5). Regarding the accuracy on the Siamese test set (ACC-Testset), we observe an increase in accuracy values across all confidence score thresholds of 0.99, 0.5, and 0.01 with a larger number of ground truth samples. For instance, at the 0.99 threshold, the ACC for $x = 2$ is merely 59.56, whereas it rises to 81.74 with $x = 20$ samples. This behavior aligns with our expectation that samples are assessed based on the knowledge within the clean reference dataset, indicating that reintroducing new samples into the reference dataset (step 6 of LabelTrust) will enhance performance. The improvement between $x = 15$ and $x = 20$ in lines (4) and (5) is negligible, suggesting performance plateauing. This supports our assumptions about the efficiency of the few-shot learning process, as apparently for this setup no more samples than $x = 15$ are necessary for optimal performance.

When evaluating the Siamese network's performance in verifying benign data (Unpoisoned-Testset), a similar trend is observed across all three confidence thresholds. For instance, at the 0.99 threshold, the ACC starts at 92.52 for $x = 2$ and increases to 97.36 for $x = 20$. Conversely, the corresponding FRR demonstrates the opposite trend: Initially, with $x = 2$, numerous benign samples are assigned confidence scores below the threshold, resulting in an FRR of 72.98 for the 0.99 threshold, essentially ensuring only samples with high similarity to the reference dataset are considered positive. This FRR diminishes to 22.94 for $x = 20$. Depending on the application context, users may opt for a low threshold, such as 0.01, yielding an FRR of only 6.59 for $x = 20$ samples, ensuring LabelTrust identifies only significantly deviant samples from the reference dataset.

When analyzing the detection performance on poisoned samples (Poison-Testset), all experiments exhibit exceedingly low average confidence scores, e.g., 0.0088 already for $x = 2$, effectively countering pixel trigger backdoors. Across all thresholds, the TRR nearly reaches 100.0.

Summary. The Siamese network of LabelTrust operates as intended, demonstrating improved outcomes with an increased number of samples within the reference dataset. While achieving favorable results across all thresholds, users have the flexibility to tailor the threshold to suit specific scenarios. The user can opt to have LabelTrust identify highly confident SL-Mappings by setting a threshold such as 0.99, or label mappings that are deemed highly unlikely to be correct with a threshold like 0.01. For subsequent experiments assessing LabelTrust's, we maintained $x = 10$ samples as our initial reference dataset size.

4.2 Generalizability

Poisoning Attack Independence. Initially, we assess LabelTrust's generalizability by replicating the experiment detailed in line (3) of Tab. 1 across different poisoning attacks. Specifically, we conduct the experiment with a Blend [7] backdoor, followed by a Clean-Label [44] pixel trigger backdoor, and lastly, an untargeted attack, which randomly assigns labels to samples (Random-Label) [4, 21].

The outcomes, outlined in lines (6), (7), and (8) of Tab. 1, exhibit comparable results to line (3).[6] Consequently, with detection rates consistently surpassing 95.20 across all scenarios, we affirm that LabelTrust functions independent of the poisoning attack in place and detects false SL-Mappings, e.g., due to poisoning attacks, reliably. Note, that the Random-Label attack mirrors the effect of a mislabeling due to human error. Hence, the results also show the performance of LabelTrust for unintentional mislabelings.

[6] Note, that the results for ACC-Testset and Unpoisoned-Testset are completely equal, as the same samples were picked for Siamese training due to the same random seed and the same amount of poisoned samples. Only the results for Poison-Testset vary depending on the new backdoor.

Table 2: Performance of the Siamese network for a pixel trigger [13] backdoor with $x = 10$ samples.

	Dataset	Model	**ACC-Testset** with confidence threshold...			**Unpoisoned-Testset**, with confidence threshold...						**Poison-Testset**			
						0.99		**0.5**		**0.01**		**AVG**	**TRR**, with threshold...		
			0.99	**0.5**	**0.01**	**ACC**	**FRR**	**ACC**	**FRR**	**ACC**	**FRR**	**Score**	**0.99**	**0.5**	**0.01**
(1)	FMNIST [50]	ResNet-18 [15]	70.79	79.29	83.01	93.71	51.60	93.50	30.42	90.11	14.69	0.0018	99.98	99.88	98.78
(2)	IIC [3]	ResNet-18 [15]	52.06	58.30	60.26	84.62	85.10	77.60	49.70	67.03	30.40	0.0132	99.48	98.77	96.55
(3)	MNIST [10]	CNN	58.98	75.71	81.73	93.60	63.58	95.80	28.79	88.73	11.06	0.0204	100.0	98.78	87.26

Dataset Independence. To demonstrate the dataset independence of LabelTrust, we replicated the experiments outlined in line (3) of Tab. 1 using alternative datasets: FMNIST [50], offering different image content, and IIC [3], featuring colored samples. The results, presented in lines (1) and (2) of Tab. 2, reveal noteworthy findings. With FMNIST [50], depicted in line (1), we observed comparable outcomes to MNIST [10], with notably high ACC values, such as 90.11 even at the 0.01 threshold for the Unpoisoned-Testset, and a modest FRR of 14.69. Remarkably, the detection rate for poisonings remained consistently above 98.78 for all thresholds. Conversely, the colored IIC [3] dataset posed a greater challenge for LabelTrust due to its increased sample diversity within each class. However, even with just $x = 10$ samples, LabelTrust achieved notable results. While the accuracies were slightly lower compared to FMNIST [50], the detection ratio for poisonings remained above 96.55 for all thresholds, with an average score of 0.0132. Thus, we assert that LabelTrust demonstrates robustness across different datasets, suggesting its dataset independence.

Model Independence. To demonstrate LabelTrust's independence from model architecture, we replicated the experiment outlined in line (3) of Tab. 1 using a Convolutional Neural Network (CNN) with a simplified structure. Due to the reduced complexity of the network, the learning rate of the Adadelt [52] optimizer was adjusted to 0.1. The results presented in line (3) of Tab. 2 demonstrate that comparable outcomes were achieved, despite the simpler architecture. While the accuracy was slightly lower compared to this attained with the ResNet-18 [15] architecture in line (3) of Tab. 1, the detection performance exhibited only a slight decrease, with values remaining notably high above 87.26 for all thresholds (reaching 100.0 for the 0.99 threshold), and an average confidence score of 0.0204 compared to 0.0042 for ResNet-18 [15]. These findings underscore LabelTrust's capability to operate effectively across diverse model architectures, affirming its architectural independence. It's noteworthy that in practical applications, such as confidence scoring of predictions of a primary model, LabelTrust's Siamese network and the primary model are inherently independent by design.

Summary. In summary, LabelTrust emerges as a versatile tool, consistently demonstrating uniform and reliable behavior across various configurations. Precisely, the evaluation has shown high performance across various poisoning methods, datasets, and Siamese architectures. Its performance remains unaffected by the type of poisoning method employed, the dataset's characteristics, and the Siamese network's architecture. Consequently, LabelTrust stands as a general-purpose solution applicable across numerous scenarios. Our subsequent evaluations will focus on its effectiveness in dataset cleaning and confidence score prediction.

4.3 Offline Dataset Cleaning

Setup. To demonstrate LabelTrust's effectiveness as an offline dataset cleaning tool, we use MNIST [10] and introduced a pixel trigger [13] backdoor, contaminating 10% of the samples in each batch resulting in 5625 poisoned samples.[7] Initially, we trained a ResNet-18 [15] model on this poisoned dataset and evaluated its accuracy on both the main task (MA) using the MNIST [10] test set and the backdoor task (BA) on the poisoned MNIST [10] test set. The results, presented in the first two columns of Tab. 3, show 98.36 MA and 100.0 BA, indicating a highly effective backdoor. Subsequently, we used LabelTrust to clean the dataset, opting for a 0.99 threshold, aiming to identify samples with high confidence scores while double-checking the remainder, simulating expert validation. Starting with a reference dataset of $x = 10$ randomly selected clean samples, we trained our LabelTrust version, which utilizes ResNet-18 [15] as the feature encoding subnetwork in the Siamese network. LabelTrust then categorized samples into clean and leftover subsets. We used the clean samples to retrain a ResNet-18 [15] model from scratch, aiming for an unbackdoored model. The iterative process continued with LabelTrust's refeeding step, where five additional randomly chosen clean samples from the leftovers were incorporated into the reference dataset in each subsequent round.[8] The updated reference dataset of each round is then used to retrain the Siamese network from scratch and reevaluate the whole dataset, resulting in a new set of clean and leftover samples. Opposed to the intended use-case of experts selecting high-quality samples in the refeed loop we select them at random to showcase the behavior under a semi-optimal expert.

Results. As reported in Tab. 3, the initial round reported in line (1) identified 37,717 clean samples and 22,283 leftovers, distinguishing 5,616 real poisoned samples and 16,667 benign samples with confidence scores below the threshold. The seemingly low accuracy of 71.86 is caused primarily by the unpoisoned samples falsely classified as leftovers. This effect

[7] We poison 10% of the 978 batches, each containing 64 samples, resulting in six poisonings per batch. The last batch, with 32 samples, has three poisonings due to the total dataset size of 60,000 samples.

[8] Note that Siamese dataset has the same amount of samples and is continuously trained for one epoch to produce comparable results. However, one could further enhance the training as discussed Sect. 5.

Table 3: Dataset cleaning statistics over six refeed rounds of LabelTrust for 60,000 samples containing 5,625 poisonings.

	Pre-LabelTrust		Post-LabelTrust		Refeed	x	Detection Statistics					
	MA	BA	MA	BA	Round		clean	leftovers	true leftovers	false leftovers	TRR	ACC
(1)	98.36	100.0	92.97	3.42	-	10	37,717	22,283	5,616	16,667	99.84	72.20
(2)	98.36	100.0	92.50	8.76	1	15	41,199	18,801	5,608	13,193	99.69	77.98
(3)	98.36	100.0	94.25	50.72	2	20	47,202	12,798	5,588	7,210	99.34	87.92
(4)	98.36	100.0	97.63	20.56	3	25	47,038	12,962	5,591	7,371	99.39	87.65
(5)	98.36	100.0	96.89	2.72	4	30	47,027	12,973	5,609	7,364	99.71	87.70
(6)	98.36	100.0	98.14	6.80	5	35	50,009	9,981	5,615	4,366	99.82	92.70

is caused by the high threshold of 0.99 and will diminish in later stages. The backdoor in line (1) was successfully filtered, achieving a BA of 3.42 after model retraining, with a marginal performance loss, reaching 92.97 MA. Notably, the TRR is 99.84, showing that LabelTrust detects a high fraction of poisonings already in the first round. This result reassures the results from the previous sections. In subsequent rounds (2) and (3) in Tab. 3, a discernible trend emerged: the Siamese network consistently identified more samples as clean, e.g., 41,199 in line (2) and 47,202 in line (3), while maintaining a very high TRR, indicating effective and reliable filtering of poisonings. Consequently, false leftovers decreased to 7,210, resulting in higher accuracies of up to 87.92 in line (3). Hence, we can see, that the randomly selected samples in the refeed loop increased the quality of the reference dataset, essentially leading to increased Siamese performance.[9] In lines (4) and (5), we can observe, that the quality of the system plateaued. Hence, the randomly selected clean samples from refeed rounds three and four do not contribute to the classification performance of the Siamese network. This behavior is expected, as the random selection (semi-optimal expert) might have selected low-quality samples. Nevertheless, if the process is continued, the original trend continues in line (6) with already 50,009 clean samples and only 9,981 leftovers reaching an accuracy of 92.70 and a TRR of 99.82. Hence, we can reason, that the choice of the clean samples is important and affects the performance of LabelTrust in terms of minimizing the number of rounds, but is not imperative as the dataset cleaning process will converge over time. Our evaluation has confirmed our assumption cf Sect. 3.2 that the dataset cleaning process converges over time resulting in a diminishing number of leftovers while maintaining a high backdoor detection (TRR). Tuning the Siamese network to better performance might yield even better results and is discussed in Sect. 5, but is out of the scope of this paper, as we focus on comparable results over multiple experiments and the depiction of the general functionality of LabelTrust.

Summary. In summary, LabelTrust functioned as intended, with the refeed loop enhancing system performance over time. Notably, even in the initial round, LabelTrust successfully removed the backdoor, with subsequent rounds yielding results akin to those observed with increasing x values in Tab. 1.

[9]Note, that all samples of the dataset get reevaluated in each round by design (cf. Sect. 3.2) to address potential mistakes made by LabelTrust in earlier rounds, especially detecting poisoned samples falsely mapped as clean.

Following the five refeed rounds detailed in Tab. 3, an expert would have reviewed only 350 samples manually, a fraction of 0.0058% compared to 60.000 samples that would have been reviewed without assistance from LabelTrust, and would only sacrifice 9,981 samples (16,63%) as leftovers, if he stops after refeed round five (line (6) in Tab. 3). Hence, LabelTrust effectively identified 50,009 samples as clean, essentially rendering a valuable tool for dataset-cleaning tasks.

4.4 Online Confidence Scoring

LabelTrust, by its design, is not supposed to filter out clean-label backdoors [44] during dataset cleaning, as these attacks rely on clean SL-Mappings during the poisoning process (cf. Sect. 2).[10] However, such attacks can be identified during model inference, as demonstrated in line (7) of Tab. 1. This motivates the combined usage of LabelTrust also in online confidence scoring application scenarios in Sect. 4.4.[11]

Setup. To demonstrate the effectiveness of LabelTrust in an online confidence scoring application, we trained a ResNet-18 [15] on MNIST [10] with a pixel trigger [13] backdoor injected on 10% of the data of each batch. We then evaluated the model's performance using both the MNIST [10] test set and a fully poisoned version of the test set, as previously used in the experiments in Sect. 4.3. We analyzed the misclassified samples from both test sets to identify instances where predictions should not be considered trustworthy. Specifically, we flagged samples misclassified by the model, whether due to poor performance or the presence of a backdoor, as potential alerts for untrustworthy predictions. We then examined the confidence scores provided by LabelTrust for these samples. For LabelTrust, we trained a Siamese network with $x = 10$, similar to the setup reported in line (3) of Tab. 1.

Results. In this evaluation, we focus on the detection of false or malicious predictions and hence SL-Mappings, as benign predictions do not pose a harmful threat to the user and hence do not need to raise a respective alarm. Benign predictions would yield similar results as for the Unpoisoned-Testset in the previous experiments and are thus omitted

[10]We conducted a respective experiment validating this assumption leading to a TRR of 4.64 and an ACC of 63.94.

[11]If LabelTrust is used with a model with high performance, e.g., 80% ACC, the final predictions' trustworthiness is still unclear. This can be provided with LabelTrust, which can be trained with starting from 2 samples per class. One pre-requirement is, that the dataset for the pre-trained model is not identical to the reference dataset, such that new insight is provided.

here. Mispredictions from the benign test set yielded a mean and median confidence score of 0.30 and 0.0018, while those from the backdoored test set yielded 0.0052 and 5.83 10 ^{7}. We achieved similar results for experiments utilizing Blend [7] and Clean-Label [44] backdoors. Further, we use the FMNIST [50], and IIC [3] dataset and report the results in App. 7.3. These results indicate that LabelTrust reliably assigns low confidence scores, alerting the user by indicating a lack of trustworthiness in the predictions, particularly for backdoored samples. Conversely, non-poisoned samples receive more moderate negative confidence score values.

Summary. Summarized, LabelTrust shows reliable performance in online confidence scoring application scenarios, providing valuable insights into prediction trustworthiness.

4.5 Runtime

In terms of runtime, we present three noteworthy findings derived from ten runs of the experiment outlined in line (3) of Tab. 1, with the times averaged. First, the training duration for the Siamese network, constituting step 3 of LabelTrust, amounted to 34.56 seconds. Subsequently, the sample pairing process, encompassing step 4 of LabelTrust, required a mere 7.62 10 ^{4} seconds. Finally, the inference process for the ten samples in this case consumed 0.0056 seconds. Consequently, we assert that the additional computational burden imposed by LabelTrust is acceptable in the offline phase and negligible in the online phase. While there may be critical considerations in application scenarios such as autonomous driving, we believe that optimization of code and utilization of specialized hardware can reduce the overhead even more.

5 Discussion

Complicated Application Scenarios. As LabelTrust relies on a Siamese network, it encounters similar challenges to other few-shot learning-based networks, particularly in complex application scenarios where representative samples for all classes are required as ground truth. In instances where subclasses exist within classes, such as different shapes of objects like cars, it's advisable to either designate a class for each subclass in the Siamese network or increase the number of samples per class in the reference dataset. Moreover, longer training periods (more epochs) are typically necessary for handling more complex application scenarios and datasets. Ultimately, the user must determine the threshold in LabelTrust's pipeline based on whether they aim to identify samples likely to be clean or potentially poisoned/low-quality. In this work, we present the general functionality and applicability over multiple application scenarios and, hence, fixate the training periods and dataset sizes to get comparable results. Further, we use experimental setups, that are useful to see the effects of LabelTrust, e.g., for different thresholds, rounds, input types, and other hyperparameters.

Optimizations. Further optimization of the Siamese training, architecture, and hyperparameters might lead to even better performance. However, we did not focus on the optimization of few-shot learners in this work but focused on their application of confidence scoring. We believe that optimizing the performance of LabelTrust can be achieved through optimizing the performance of the Siamese network, mainly by four key measures. Firstly, employing a different loss function, such as Triplet Loss (cf. App. 7.2), may yield superior performance depending on the specific application scenario. Secondly, while LabelTrust is designed to be architecture-agnostic, customizing the Siamese network architecture to the specific application can enhance performance. Thirdly, ensuring high-quality sampling by involving a knowledgeable domain expert is crucial, as the quality of the reference dataset significantly impacts LabelTrust's performance. Further, the generation mechanism for the Siamese dataset can be optimized to generate highly effective pairs.

Adaptive Adversaries. Adapting to LabelTrust requires knowledge of expert-selected samples, which is not given. Neither the Siamese network nor the reference dataset is accessible to adversaries, preventing adaption, e.g., crafting adversarial examples [42]. LabelTrust evaluates SL-Mappings after poisoning attacks, such as dataset-poisoning or poisoned model training, have already been conducted.

Dependency on Data Distributions. Conceptually, the training dataset distribution is irrelevant for LabelTrust. We assume a minimal sample amount in the reference dataset, starting from 2 samples per class. Certainly, the Siamese network creator and domain expert should create a balanced dataset, which generally yields better performance. However, within LabelTrust, each SL-Mapping is evaluated individually without dependencies on other samples or pre-trained models.

6 Related Work

The related work for this paper draws from several interconnected domains. Initially, we examine research in the field of confidence scoring. Subsequently, we delineate our investigation against methodologies for dataset cleaning. Thirdly, we explore studies focused on mitigating data poisoning within datasets utilized by machine learning models.

6.1 Confidence Scores

Confidence Scoring generates scores that assess the reliability of predictions generated by a trained model.

Corbière *et al.* [9] employ a secondary model to generate a confidence score for predictions, relying on intermediate layer outputs from a pre-trained primary model. Consequently, its

efficacy is contingent not only on a specific model architecture but also on the characteristics of a particular model. This confidence scoring model undergoes training using the entire training dataset. A similar methodology is adopted by Luo *et al.* [28], which introduce a novel loss function within the same framework. LabelTrust instead is a general tool for ST-Mapping verification and hence can be used to evaluate the prediction outcome of any primary model, essentially being independent of a specific model instance or architecture.

Jiang *et al.* [19], on the other hand, utilize the entire training dataset to construct sets of samples with high density for each class through the nearest neighbor algorithm. For a new prediction, the distance to these sets is computed, resulting in a lower prediction confidence if the prediction deviates from the nearest distance. However, as only a straightforward distance metric is employed, this approach does not analyze the precise content of the sample, such as the scenery depicted in an image. Additionally, the computational complexity during sample evaluation increases with the number of classes. Contrary, leveraging a Siamese network with convolutional layers, LabelTrust considers the specific features of a sample's content, rather than solely relying on distance metrics. The features extracted by a similarity metric might not capture the full complexity of visual information, leading to inaccurate similarity assessments. Neural networks, on the other hand, can learn their own feature representations directly from the image data, potentially capturing more nuanced relationships between images. Further, the effort to evaluate one ST-Mapping is constant, e.g., one model inference of the Siamese network. Additionally, the efficacy of the methods lies in analyzing the entire training dataset, assuming the presence of clusters of benign samples, thus presupposing a majority of clean samples. In contrast, LabelTrust relies on a small number of clean samples, allowing it to effectively manage datasets with a high degree of compromised data.

Moon *et al.* [30] propose a model-dependent scoring mechanism embedded during training via custom loss and does not consider poisonings. LabelTrust instead does not interfere with the training of the pre-trained model.

Jha *et al.* [18] identify important features in an input sample. Iteratively, the approach changes those features and observes the model output of the modified sample over multiple inference rounds to yield a confidence score. As the approach relies on a trained model, it cannot be applied to dataset cleaning and induces significant computational overhead, due to multiple inference steps.

Guillory *et al.* [14] predict general model performances under distribution shift instead of scoring individual samples, which is an orthogonal problem motivating LabelTrust.

Overall, while the primary focus of most of these studies [9, 19, 28, 30] is to furnish scoring mechanisms for predictions, they do not address or prioritize defense against poisoning attacks, such as backdoors, or introduce high overhead during inference [18]. In contrast, LabelTrust is designed to operate within a more realistic scenario, accounting for adversarial attacks. Further, LabelTrust only needs a fraction of the training samples for training. Hence, the confidence score of those works is based on large untrusted datasets as references, whereas LabelTrust's confidence score relies on a small subset of high-quality samples resulting in high trustworthiness.

6.2 Dataset Cleaning

Dataset cleaning aims at identifying poisoned samples within a dataset essentially separating poisoned from clean samples to allow for unpoisoned training.

Huang *et al.* [17] utilize outputs from specific layers (activations) of clean samples to train a secondary network tasked with scoring predictions. By relabeling and reintroducing samples based on these scores, they assert that only 1% of training data must be clean. [17] can be considered the closest related work to LabelTrust. However, it is reliant not only on model architecture but also on a specifically trained model instance, unlike LabelTrust, which remains agnostic to such constraints. Furthermore, by leveraging a Siamese network and few-shot learning, LabelTrust can operate with even fewer samples than the 1% required in [17].

Paudice *et al.* [35] detects label-flipping attacks and simultaneously offers a method to adjust previously flipped labels. Peri *et al.* [36] focuses on the detection of clean-label backdoors. Both works, use a similar principle as Jiang *et al.* [19], which results in the same downsides that we mentioned above.

Gao *et al.* [12] developed a model aimed at partitioning a suspicious dataset into clean and poisoned subsets. The loss generated by individual samples provides insights into their potential contamination. Ramzi *et al.* [38] employ an auto-encoder architecture to identify data poisonings, leveraging reconstruction errors to distinguish between clean and contaminated samples. Pan *et al.* [33] introduces a sample filtering method specially designed for backdoored samples based on the optimization of a detection model against multiple objectives. Based on a small clean dataset, which is used for training first, the loss of samples provides insight if the samples are poisoned. While [12, 33, 35, 38] focus solely on offline dataset cleaning, LabelTrust simultaneously addresses both offline dataset cleaning and online prediction scoring.

Other papers such as [22, 23, 39], focused on dataset cleaning, concentrate solely on identifying [22, 23] and potentially repairing [39] low-quality (dirty) training samples, without addressing poisoned data, thus diverging from our work.

6.3 Poisoning Prevention

Poisoning prevention aims at making a model robust to the existence of poisoned samples within a dataset, essentially training a benign model on a poisoned dataset.

Wang *et al.* [46] propose a system to safeguard the feature extraction component of a model against backdoor feature learning. The authors achieve this by retraining the classification part (the final layers) on a clean dataset, constituting 10% of the entire dataset. Thus, rather than cleaning the dataset, the approach modifies the architecture to withstand backdoor attacks, making it reliant on specific model architectures. Instead, LabelTrust is independent of the model architecture and only needs a small fraction of data for few-shot learning.

Li *et al.* [24] introduce a system that detects backdoors by identifying poisoned samples based on their loss values (similar to [33]) and assigning random labels to these samples to disrupt the connection to the backdoor target class. Although this system is not designed to isolate poisoned samples, it could potentially be repurposed for this task. However, the approach relies on an empirically determined threshold and necessitates altering the mapping of samples, which could degrade the model performance similar to untargeted poisoning, despite its effectiveness against backdoors. LabelTrust does not negatively impact the performance of a trained model by filtering the poisoned samples out instead of relabeling them. Further, we provide a meaningful confidence score allowing for versatile operation.

Building upon Li *et al.* [24], Ying *et al.* [51] propose a training pipeline incorporating unlearning for a subset of identified poisoned samples. These samples are discerned by analyzing prediction probabilities and applying a threshold, as poisoned samples typically yield higher probabilities. Compared to LabelTrust, the method can not be applied to pre-trained models and does not provide a meaningful confidence score.

Jiang *et al.* [20] extend the approach of Li *et al.* [24] by integrating a small secondary head into the model. This head, utilizing feature embeddings from the primary model, is trained to recognize backdoors and subsequently relabel poisoned samples. Hence, the method is dependent on the architecture of the primary model, which is not the case for LabelTrust.

While the aforementioned works [20, 24, 46, 51] aim to create benign models even when trained on poisoned datasets, LabelTrust takes a different approach by focusing on providing a confidence score for each ST-Mapping. Unlike the loss-based methods described above, LabelTrust can be used with already pre-trained models, e.g., does not require changing the training of the primary task model. Further, the confidence score it generates offers meaningful insights, as opposed to the threshold-based clustering of benign and malicious samples.

In summary, LabelTrust offers distinct advantages over related works from the mentioned research areas. However, some benefits of LabelTrust are unique across all works: the consolidation of two use cases, the minimization of the trusted dataset by utilizing Few-Shot-Learning, and ongoing enhancement facilitated by a refeed-loop mechanism overseen by a human expert.

7 Conclusion

Assessing the quality of Sample Label Mappings (SL-Mappings) is essential in machine learning, whether derived from labeled datasets or inferred during model inference. The existing solutions from the domains of dataset cleaning and prediction confidence scoring lack a dual-use tool for this challenge, only providing standalone approaches with individual downsides. Therefore, we introduce LabelTrust, a versatile and architecture-independent tool that leverages a Siamese network trained via few-shot learning based on a few clean samples to provide a confidence score for SL-Mappings. Thereby, LabelTrust effectively addresses the downsides of existing standalone solutions, while reliably detecting poisonings in both contexts. With an interactive loop for system improvement, only 0.0058% of the dataset needs manual evaluation to clean 83.73% of a dataset while ensuring high model performance without an active backdoor.

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).

References

[1] Abien Fred Agarap. Deep Learning using Rectified Linear Units (ReLU). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Kian Ahrabian and Bagher BabaAli. Usage of autoencoders and Siamese networks for online handwritten signature verification. *Neural Computing and Applications*, 2019.

[3] Puneet Bansal. Intel image classification. Kaggle, 2021. https://www.kaggle.com/datasets/puneet6060/intel-image-classification.

[4] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning Attacks against Support Vector Machine. *ICML*, 2012.

[5] Lei Cai, Jingyang Gao, and Di Zhao. A review of the application of deep learning in medical image classification and segmentation. *Annals of translational medicine*, 2020.

[6] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. *ICCV*, 2015.

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

[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *JMLR*, 2011.

[9] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing Failure Prediction by Learning Model Confidence. *NeurIPS*, 2019.

[10] Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal Processing Magazine*, 2012.

[11] Hossein Fereidooni, Jan König, Phillip Rieger, Marco Chilese, Bora Gökbakan, Moritz Finke, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. AuthentiSense: A Scalable Behavioral Biometrics Authentication Scheme using Few-Shot Learning for Mobile Platforms. *NDSS*, 2023.

[12] Kuofeng Gao, Yang Bai, Jindong Gu, Yong Yang, and Shu-Tao Xia. Backdoor Defense via Adaptively Splitting Poisoned Dataset. In *IEEE/CVF*, 2023.

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

[14] Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. Predicting With Confidence on Unseen Distributions. *ICCV*, 2021.

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

[16] Yue He, Zheyan Shen, and Peng Cui. [towards non-i.i.d. image classification: A dataset and baselines]. *Pattern Recognition*, 2021.

[17] Huayang Huang, Qian Wang, Xueluan Gong, and Tao Wang. Orion: Online Backdoor Sample Detection via Evolution Deviance. *IJCAI*, 2023.

[18] Susmit Jha, Sunny Raj, Steven Fernandes, Sumit K Jha, Somesh Jha, Brian Jalaian, Gunjan Verma, and Ananthram Swami. Attribution-Based Confidence Metric For Deep Neural Networks. *NeurIPS*, 2019.

[19] Heinrich Jiang, Been Kim, Melody Guan, and Maya Gupta. To Trust Or Not To Trust A Classifier. *NeurIPS*, 2018.

[20] Yujing Jiang, Xingjun Ma, Sarah Monazam Erfani, Yige Li, and James Bailey. End-to-End Anti-Backdoor Learning on Images and Time Series. *arXiv preprint arXiv:2401.03215*, 2024.

[21] Torsten Krauß and Alexandra Dmitrienko. Avoid Adversarial Adaption in Federated Learning by Multi-Metric Investigations. *arXiv preprint arXiv:2306.03600*, 2023.

[22] Sanjay Krishnan, Michael J Franklin, Ken Goldberg, Jiannan Wang, and Eugene Wu. ActiveClean: An Interactive Data Cleaning Framework For Modern Machine Learning. *SIGMOD*, 2016.

[23] Sanjay Krishnan, Jiannan Wang, Michael J Franklin, Ken Goldberg, Tim Kraska, Tova Milo, and Eugene Wu. SampleClean: Fast and Reliable Analytics on Dirty Data. *IEEE Data Eng. Bull.*, 2015.

[24] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-Backdoor Learning: Training Clean Models on Poisoned Data. *NeurIPS*, 2021.

[25] Yikai Li, CL Philip Chen, and Tong Zhang. A Survey on Siamese Network: Methodologies, Applications, and Opportunities. *IEEE TAI*, 2022.

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

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

[28] Yan Luo, Yongkang Wong, Mohan S Kankanhalli, and Qi Zhao. Learning to Predict Trustworthiness with Steep Slope Loss. *NeurIPS*, 2021.

[29] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese Network Features for Image Matching. *IEEE ICPR*, 2016.

[30] Jooyoung Moon, Jihyo Kim, Younghak Shin, and Sangheum Hwang. Confidence-aware learning for deep neural networks. *ICML*, 2020.

[31] Iqbal Muhammad and Zhu Yan. Supervised machine learning approaches: A survey. *ICTACT Journal on Soft Computing*, 2015.

[32] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.

[33] Minzhou Pan, Yi Zeng, Lingjuan Lyu, Xue Lin, and Ruoxi Jia. ASSET: Robust Backdoor Data Detection Across a Multiplicity of Deep Learning Paradigms. *USENIX Security*, 2023.

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

[35] Andrea Paudice, Luis Muñoz-González, and Emil C Lupu. Label Sanitization Against Label Flipping Poisoning Attacks. *ECML PKDD 2018 Workshops*, 2019.

[36] Neehar Peri, Neal Gupta, W Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P Dickerson. Deep k-NN Defense Against Clean-Label Data Poisoning Attacks. *ECCV*, 2020.

[37] Erhard Rahm, Hong Hai Do, et al. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 2000.

[38] Fereshteh Razmi and Li Xiong. Classification Auto-Encoder Based Detector Against Diverse Data Poisoning Attacks. In *IFIP DBSec*, 2023.

[39] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. HoloClean: Holistic Data Repairs with Probabilistic Inference. *arXiv preprint arXiv:1702.00820*, 2017.

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

[41] Aria Salari, Abtin Djavadifar, Xiangrui Liu, and Homayoun Najjaran. [object recognition datasets and challenges: A review]. *Neurocomputing*, 2022.

[42] Alex Serban, Erik Poll, and Joost Visser. Adversarial Examples on Object Recognition: A Comprehensive Survey. *ACM Comput. Surv.*, 2020.

[43] The Linux Foundation. Pytorch, 2022. https://pytorch.org.

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

[45] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[46] Haotao Wang, Junyuan Hong, Aston Zhang, Jiayu Zhou, and Zhangyang Wang. Trap and Replace: Defending Backdoor Attacks by Trapping Them into an Easy-to-Replace Subnetwork. *NeurIPS*, 2022.

[47] Xutong Wang, Chaoge Liu, Xiaohui Hu, Zhi Wang, Jie Yin, and Xiang Cui. Make Data Reliable: An Explanation-powered Cleaning on Malware Dataset Against Backdoor Poisoning Attacks. *ACSAC*, 2022.

[48] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a Few Examples: A Survey on Few-shot Learning. *ACM CSUR*, 2020.

[49] Haoran Wu, Zhiyong Xu, Jianlin Zhang, Wei Yan, and Xiao Ma. Face Recognition based on Convolution Siamese Networks. *IEEE CISP-BMEI*, 2017.

[50] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[51] Zonghao Ying and Bin Wu. DLP: towards active defense against backdoor attacks with decoupled learning process. *Cybersecurity*, 2023.

[52] Matthew D Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*, 2012.

Appendix

7.1 Hardware & Experimental Setup

Experiments were conducted using PyTorch, a leading Python-based machine learning library [34, 43, 45], on a server with an AMD EPYC 7413 24-Core Processor (64-bit) boasting 96 processing units and 128GB main memory. CUDA [32] facilitated access to an NVIDIA A16 GPU with 4 virtual GPUs, each featuring 16GB GDDR6 memory.

7.2 Siamese Loss Functions

Alternatives to BCELoss introduced in Sect. 2 include the Contrastive loss and the Triplet loss [25]. The Contrastive loss operates on data record pairs and aims to minimize the distance between records from the same class, while maximizing the distance between samples from different classes towards a predefined margin m. Thereby, a respective distance function $d(r_1, r_2)$ for the two data records r_1 and r_2 is applied, e.g., the Euclidean distance. A general depiction of the corresponding loss is illustrated in Eq. 2. y_i is again the label and $d(r_1, r_2)$ calculates the distance of the embeddings $e(r_1)$ and $e(r_2)$ that were generated by the feature embedding subnetworks.

$$CLoss = \frac{1}{N} \sum_{i=1}^{N} y_i \; d(r_1, r_2) + (1 \quad y_i) \; max(0, m \quad d(r_1, r_2))$$
$$d(r_1, r_2) = jje(r_1) \quad e(r_2)jj^2 \tag{2}$$

Table 4: Mean and median scores for mispredictions from the benign test set (Benign) and the backdoored test set (Backdoored). The default setting is ResNet-18 [15] on MNIST [10] with a pixel trigger [13] backdoor injected on 10% of the data of each batch.

	Benign		Backdoored	
	Mean	**Median**	**Mean**	**Median**
Default	0.3006	0.0182	0.0052	5.83 10 ^{7}
Default + FMNIST [50]	0.3442	0.0048	0.0001	5.35 10 ^{10}
Default + IIC [3]	0.3419	0.0206	0.0266	1.57 10 ^{6}
Default + Blend [7]	0.3271	0.0246	0.0683	1.047 10 ^{5}
Default + Clean-Label [44]	0.3676	0.0381	0.0751	2.7423 10 ^{6}

On the other hand, the Triplet loss involves three data records, forming two pairs. Initially, an anchor record a is selected and paired with a positive record p from the same class and a negative record n from a different class. Subsequently, the two pairs undergo processing, and the loss is calculated, as outlined in Eq. 3, aiming towards the same goal as the Contrastive Loss.

$$TLoss = \frac{1}{N} \sum_{i=1}^{N} max(0, d(a,p) \quad d(a,n) + m) \tag{3}$$

LabelTrust could also be used with Contrastive or Triplet loss, but respective experiments were omitted within this paper.

7.3 Confidence Scoring Experiments

Due to space limitations of the paper, we report additional results for the confidence scoring (cf. Sect. 4.4) use case here. We conducted experiments with the Blend [7] and the Clean-Label [44] backdoor, as well as with the FMNIST [50], and IIC [3] dataset. The first line of Tab. 4 depicts the default setting reported in Sect. 4.4 with ResNet-18 [15] on MNIST [10] with a pixel trigger [13] backdoor injected on 10% of the data of each batch. The subsequent lines of Tab. 4 report the results when changing either the backdoor or the dataset from the default setting. We can observe similar results for all of the experiments.

7.4 Visualization of the Reference Dataset

To give an impression of how the reference dataset looks like in our experiments, we visualize the reference dataset for $x = 10$ samples per class. Fig. 5 shows the ten samples for each class from the MNIST [10] dataset, which have been randomly chosen in our experiments.

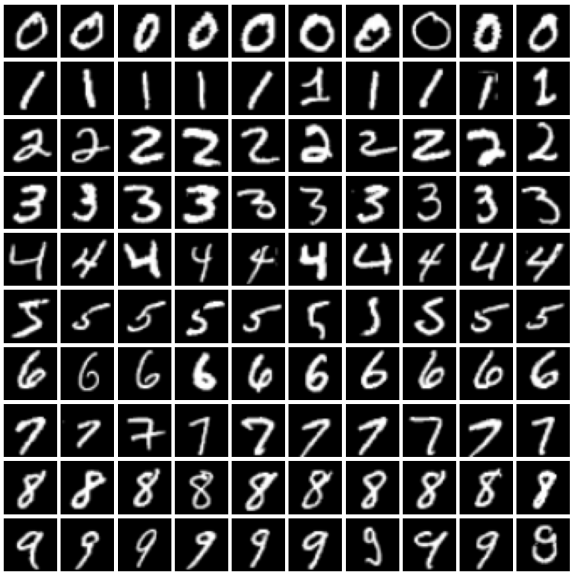

Figure 5: Visualization of the reference dataset with $x = 10$.