Machine Learning (2026) 115:124
https://doi.org/10.1007/s10994-026-07010-4

Automated Membership Inference via Prompt-Based Attacks in Generative Models

Daniela Gallo[1,2] · Angelica Liguori[1] · Ettore Ritacco[3] · Luca Caviglione[4] · Fabrizio Durante[2] · Giuseppe Manco[1]

Received: 18 April 2025 / Revised: 5 December 2025 / Accepted: 8 February 2026

Abstract
The growing adoption of prompt-based generative models has raised concerns over the unauthorized use of proprietary data, as such models may memorize and replicate training content. To address this issue, we introduce ProCAP, a novel Membership Inference Attack approach based on a prompt-driven auditing framework.Given a proprietary dataset and a target generative model, ProCAP trains an auxiliary model to craft prompts that trigger the target model to produce outputs revealing potential violations of the proprietary data.Unlike current literature, ProCAP is automatic, fully black-box, model-agnostic, and designed to operate in settings with limited or no knowledge of the training process.To reduce the computational cost of training the prompt generator, we adopt an optimization strategy that filters high-loss samples, i.e., those less likely to have been memorized. Our approach can then "specialize" the learning phase on the most informative data regions. We validate ProCAP across different scenarios, by using both real and synthetic data. Results demonstrate its effectiveness in recognizing unauthorized data usages with strong accuracy-efficiency trade-offs.

Keywords Data disclosure · Intellectual property · Generative AI · Membership inference attack · Transformer

1 Introduction

The success of modern Machine Learning (ML) models mainly depends on the quality and quantity of data used for training, which can directly influence their performance and generalization capabilities. To this aim, high-quality, diverse, and representative datasets are essential for achieving accurate and unbiased predictions. Noisy or uncleaned data can degrade the performance and introduce unintended impacts for both software ecosystems and individuals (Fischer, 2023). As ML frameworks continue to gain a widespread adoption, challenges related to the identification and traceability of training data sources have become

Communicated by Editor: Bo Han.

Extended author information available on the last page of the article

increasingly relevant. These issues are especially critical for generative models, which typically require vast and heterogeneous datasets to perform well on a wide range of tasks. Such data are often collected via automated tools (e.g., web scrapers or bots), and most of the advanced models may depend on large-scale data acquisition campaigns to ensure sufficient coverage and quality (Sirisuriya, 2023).

The composition of a training dataset also raises significant ethical and legal implications. Data collected without considering provenance, access policies, or licensing constraints could lead to leakage, surveillance, or incorrect behaviors. Since ML models are not able to discriminate among public or restricted data sources, they may generate outputs that potentially replicate content without proper authorization (Li et al., 2024b). As an example, the widespread deployment of low-cost IoT devices can enable large-scale data collection campaigns, potentially leading to mass user profiling or the unintended exposure of industrial processes (Jin et al., 2018). Similarly, large-scale collections of online images and text data may unintentionally incorporate proprietary materials, leading to an unintended reproduction of personal likenesses or fragments of creative contents (Chang et al., 2023; Somepalli et al., 2023). To mitigate these risks, ML training pipelines should incorporate mechanisms that enforce data governance policies, such as fine-grained access control schemes and explicit authorization protocols (Meurisch & Mühlhäuser, 2021).

For the specific case of evaluating whether an ML model takes advantage of data obtained without consent, it becomes essential to rapidly recognize the unauthorized usage or leakage of ML-generated contents (Sobel, 2017). In this perspective, we investigate methods for determining whether a generative model has been trained on proprietary data. While this problem shares similarities with membership inference, it departs from classical Membership Inference Attacks (MIAs) (Shokri et al., 2017) in a critical respect. Specifically, our problem considers scenarios in which the training dataset of the generative model is not directly accessible. Traditional MIAs typically assume some knowledge about the training distribution or its aggregate statistical properties, which are then leveraged to detect overfitting or memorization in the outputs produced by the model. However, such assumptions are impractical in our setting, where only partial or fragmentary knowledge of the proprietary data is available. As a result, conventional MIA techniques cannot be applied, necessitating alternative strategies to identify unauthorized data usages.

To cope with such a limitation, we propose ProCAP (**Pro**prietary **C**ontent **A**udit via **P**rompting), a technique designed to assess whether a generative model has been trained on proprietary data, without requiring access to its training set. Generative models produce synthetic outputs conditioned on input prompts; that is, they learn to map input queries to coherent outputs based on patterns observed during training. In this perspective, ProCAP introduces an auxiliary model that generates targeted prompts to induce the model under audit revealing responses. The core intuition is that generative models often exhibit a tendency to memorize portions of their training data (Liu et al., 2025). By generating prompts that increase the likelihood of triggering such memorized content, ProCAP can expose potential leakages of proprietary information. Notably, ProCAP operates under the assumption that the auditor is the legitimate owner of the proprietary data being tested. Data owners can then use ProCAP to verify whether their data have been used without authorization for training a generative model. This setting falls under the "claim unauthorized data usage" scenario defined in Wu and Cao (2025), with the specific goal of detecting unauthorized utilization of proprietary information. Importantly, ProCAP does not attempt

to discover leakage of unknown or undisclosed data, as this is fundamentally unidentifiable in a black-box setting. Instead, the framework is designed to surface violations with respect to a known proprietary dataset, which aligns with real-world copyright, licensing, and data-ownership disputes.

Although prompt-based auditing provides an effective means to investigate potential data leakage in generative models, the training of the auxiliary model can be computationally expensive. This phase requires learning to generate inputs that can reveal violations in the use of proprietary data from the target model, which involves optimizing over a large and complex prompt space. To improve the scalability of ProCAP, we propose an optimization procedure that reduces the training overhead of the auxiliary model. Our approach is based on the empirical observation that data points associated with higher training loss values are less likely to correspond to actual data leakage. Consequently, when training reaches a plateau, we dynamically prune high-loss samples from the training set of the auxiliary model. This strategy not only reduces the computational cost by shrinking the dataset over time, but also focuses the learning process on regions of the input space where memorization is more probable. As a result, the training is accelerated while maintaining a favorable balance between the accuracy and efficiency of the audit process.

The proposed approach is designed to be model-agnostic and can be applied to any generative machine learning model that operates via prompting. This includes a wide range of architectures, such as Large Language Models (LLMs), Variational Autoencoders, Generative Adversarial Networks, and Diffusion Models. As long as the model supports prompt-based generation, ProCAP can be employed to audit its behavior and investigate potential misuse of proprietary data.

To demonstrate the practical applicability of our method, we first focus on a case study involving the unauthorized use of IoT sensor measurements. Then, we further evaluate ProCAP on textual data, which raise distinct yet complementary concerns.

In summary, this work makes the following key contributions:

- It introduces ProCAP, a novel technique that enables the detection of proprietary data usage in generative models without requiring access to the training set, exploiting prompt-based probing and model memorization.
- To reduce the computational cost of its training, it proposes a loss-based filtering strategy that prunes high-loss samples unlikely to have been memorized.
- It demonstrates the effectiveness of ProCAP in two practical scenarios also by means of synthetic data to outline possible limitations.

The rest of the paper is structured as follows. Section 2 reviews past research on memorization in generative models and on membership inference, while Sect. 3 outlines the problem and the reference scenario. Section 4 presents our approach for generating prompts to reveal whether a model has made unauthorized data usage and introduces a speedup strategy for reducing training times. Section 5 evaluates our framework against both real and synthetic datasets containing IoT measurements, whereas Sect. 6 extends the analysis to generative models working on textual data. Finally, Sect. 7 concludes the paper and hints at future research directions.

2 Related Work

To understand the problem of recognizing unauthorized data usages, we first review prior research on potential unintended leakages of generative models due to their memorization capabilities. We then showcase works dealing with MIAs and finally we discuss existing methods to highlight the novelty of ProCAP.

2.1 Memorization and Data Leakage in Generative Models

Over the past decade, generative models have become a key component in deep learning, especially for handling complex, unlabeled data. Models like Auto-regressive frameworks (Deistler & Scherrer, 2022), Diffusion Models (Sohl-Dickstein et al., 2015; Ho et al., 2020), Generative Adversarial Networks (Goodfellow et al., 2020), Variational Autoencoders (Kingma & Welling, 2014), and Transformer networks (Vaswani et al., 2017) have significantly pushed forward the capabilities of ML, with applications ranging from image generation to natural language understanding. An often overlooked aspect is their tendency to memorize parts of the training data.

The memorization behavior of generative models has recently received growing attention due to its implications for privacy and security (Liu et al., 2025). For instance, Carlini et al. (2021) showed that LLMs can output exact sequences from their training sets, including potentially sensitive or confidential information. This challenges the common belief that generative models only produce novel content, and highlights the risks of data leakage when using these models. In a follow-up work, Carlini et al. (2023b) studied the problem in detail, searching for the main factors that lead to memorization phenomena. They found that transformer-based models not only memorize but can also emit training data when triggered by specific prompts. This observation was further supported by Kim et al. (2023), who developed a theoretical framework to analyze the memorization capacity of transformers. They also revealed the role of the positional encoding in promoting memorization, particularly in non-equivariant settings.

2.2 Membership Inference Attacks and Their Implications

MIAs exploit the memorization tendencies of ML models to determine whether a specific data point was part of the training set of the model (Shokri et al., 2017). Typically, this is achieved by computing a membership score, based on output similarity with a ground truth or likelihood of the predictions, which is then used to classify the data point as a "member" (e.g., belonging to a proprietary dataset) or "non-member".

Although both traditional MIAs and ProCAP aim to detect whether a model was trained on a given dataset, they differ significantly in assumptions and methodology. In classifier-based MIAs (Shokri et al., 2017), attackers employ shadow training to build models that replicate the behavior of the target. The shadow models are trained on auxiliary data similar to the target's training set, and their prediction vectors are labeled as "member" or "non-member". The obtained labeled outputs are then used to train a binary classifier that learns to distinguish the behavior of the target model when considering member versus non-member

data. However, this approach requires knowledge of the target model's architecture and internal parameters in the case of white-box attacks (Nasr et al., 2019), or at least access to its prediction outputs on auxiliary data for black-box attacks (Chen et al., 2020). In both scenarios, attackers also need auxiliary datasets and substantial computational resources to train shadow models or perform metric-based analyses.

The idea of extracting memorized training data was first introduced in Carlini et al. (2021), where the authors showed that LLMs can reproduce verbatim sequences from their training data, including sensitive information such as personal details or source code, by using a generate-and-filter pipeline combined with advanced sampling techniques. To improve MIA performance, Carlini et al. (2022) proposed a method that targets high-confidence member predictions by analyzing model outputs and gradients. Their approach uses per-example hardness scores and Gaussian likelihood estimates to focus on true-positive rates at extremely low false-positive rates.

Recent works have extended these attacks from single samples to full datasets (Maini et al., 2021). For example, Maini et al. (2024) proposed an approach that aggregates multiple MIAs and applies statistical hypothesis testing to infer whether an entire dataset contributed to the training of an LLM.

Other research has investigated memorization in generative models beyond LLMs. For instance, Carlini et al. (2023a) studied how diffusion-based models (e.g., Stable Diffusion and DALL-E 2) memorize and reproduce individual training images. By prompting the models with known queries and applying similarity detection, they showed that larger models retain more data, even rare or unique samples, raising significant concerns over privacy and copyright. Like previous research, they also rely on generate-and-filter pipelines informed by prior knowledge of the training content.

A complementary approach explores dataset watermarking, where data owners embed detectable signatures into proprietary datasets to verify their usage by third-party models (Tang et al., 2023; Wei et al., 2024; Huang et al., 2024; Chen, 2024; Li et al., 2024a). These techniques combine watermark embedding with verification algorithms that perform MIA-like queries. However, they assume that the data were marked before training, which is not always possible.

2.3 Analysis

The review of the literature highlights three key insights that frame the motivation and relevance of ProCAP.

First, we can state that when a generative model reproduces training content, it is not random. The literature clearly showcased that generative models can inherently store and retrieve training examples, and this behavior raises major concerns about data leakage, privacy violations, and copyright infringement. These insights strongly support the motivation behind ProCAP, which relies on the idea that carefully designed prompts can surface memorized content, helping detect the unauthorized use of proprietary data in generative models.

Second, state-of-the-art MIAs suffer from significant limitations. Most of them rely on access to model internals (e.g., weights and biases) or require to generate a large number of output predictions from the model and/or its shadow alternatives, which may not

be feasible in real-world commercial environments. They also require auxiliary datasets that closely resemble the original training data and often depend on hand-crafted prompts, conditions difficult to meet in practice. Moreover, the computational overhead of training shadow models or running exhaustive generation-and-filter loops limits their practicality. In contrast, ProCAP provides a more scalable solution and operates in a black-box setting, requiring only the target model and the proprietary data suspected of unauthorized use. It automatically and efficiently generates prompts designed to reveal memorized content, without needing prior access to the model's architecture, training data, or auxiliary datasets. Additionally, ProCAP does not rely on shadow models or handcrafted prompts. Thus, conducting a quantitative comparison with MIA-like methodologies is not meaningful. At the same time, supplying prompts to MIA techniques would unfairly benefit them by giving additional information beyond what is available in our context.

Third, unlike watermarking techniques, which require the deliberate injection of identifiable markers into the data during the training phase, ProCAP operates without altering the proprietary data in any way. Watermarking approaches depend on embedding prearranged signals within the data to later verify its unauthorized use, a process that is only effective if applied proactively before model training. In contrast, ProCAP requires no prior intervention on the data or access to the training process, making it suitable for auditing models trained on unmodified datasets, including those where watermarking was not feasible or foreseen.

These three insights collectively underline the practical value and originality of ProCAP. To the best of our knowledge, no existing approach attempts to automatically generate prompts for inducing generative models to leak memorized content from proprietary datasets in a fully black-box, model-agnostic setting. ProCAP fills this critical gap by providing an auditing framework that advances the responsible deployment and monitoring of generative models in sensitive data contexts.

3 Problem Statement

As previously discussed, ProCAP provides an automated method for determining whether a generative model has utilized, during its training or fine-tuning phases, data or segments that require authorization (e.g., due to licensing or copyright constraints). In this context, we refer to such data as *proprietary*, the usage of them as *unauthorized*, and the generative model as *target model*. The target model generates synthetic *outputs* in response to specific *prompts*, and those outputs that exhibit sufficient similarity to proprietary data are referred to as *violations*.

In a scenario where there is no information about the architecture of the target model, training sets, examples of effective prompts, or the relationships between *prompt-output* pairs, the owner of a proprietary dataset typically must perform extensive and costly tests to detect unauthorized usage, often relying on manually crafted prompts. In this perspective, ProCAP efficiently automates this process by leveraging the tendency of generative models to memorize training data.

Formally, the problem addressed by ProCAP can be described as follows. Let $\mathcal{D}_1$ represent a set of prompt-output pairs (k, v), where $k \in K$ and $v \in V$. Suppose $\mathcal{D}_1$ is a "confidential" training set used to train Φ, a model designed to generate elements from V in response to inputs from K. Additionally, assume the existence of a proprietary dataset $\mathcal{D}_2 \subseteq V$, whose elements require explicit authorization for use. If the owner of $\mathcal{D}_2$ suspects that their data was used without permission in the training of Φ, they must determine whether $\mathcal{D}_1$ contains any elements from $\mathcal{D}_2$, i.e., whether $\{v \mid (k, v) \in \mathcal{D}_1\} \cap \mathcal{D}_2 \neq \emptyset$. An example of this reference scenario is illustrated in Fig. 1a.

The challenge lies in verifying whether Φ has made unauthorized usage of proprietary data, meaning it was trained on $\mathcal{D}_2$, despite the impossibility of directly inspecting the secret training set $\mathcal{D}_1$. Consequently, ProCAP requires to "submit" prompts to the target model and evaluates the outputs to determine whether they represent violations. Figure 1b demonstrates how ProCAP fits within this reference scenario.

4 Exposing Data Through Prompt Generation

To address this problem, we propose the framework illustrated in Fig. 1b. The idea is to exploit another generative model, Θ, to produce specific prompts that induce Φ to generate elements identical or significantly similar to a subset of the proprietary dataset $\mathcal{D}_2$. Specifically, given a value $v \in \mathcal{D}_2$, we aim at inferring a prompt $k \in K$, using Θ, which attempts to force Φ to generate an exact copy or a slightly altered variant of v. Formally, the framework is defined by Algorithm 1, which allows for identifying candidate violations that require human inspection to be certified as actual unauthorized uses.

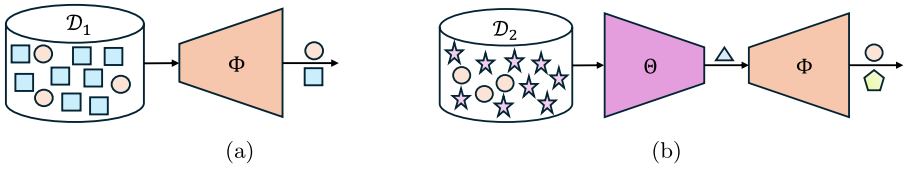

Fig. 1 An example of the reference scenario. (a) Let Φ be a publicly-available generative model trained on a private dataset $\mathcal{D}_1$. This dataset contains instances that may require authorization for use (represented as circles) and examples that do not require explicit authorization (represented as squares). The model is designed to generate, starting with specific prompts, realistic synthetic data based on this training set. Consequently, it might reproduce or closely resemble the proprietary instances (circles), potentially leading to violations of data usage regulations. (b) Let $\mathcal{D}_2$ be a dataset with proprietary examples (represented as circles and stars). The data owner suspects that the generative model Φ has used its data without permission. To verify this, the owner uses another generative model, Θ, to generate prompts (represented as triangles), triggering the generation through Φ, which is only used in inference mode, to produce data similar to the dataset under inspection. If Φ generates data that match any elements in $\mathcal{D}_2$ (circles), it indicates that the model is suspected of being trained on the proprietary data. Conversely, if it generates only different data (pentangles), there is no evidence that Φ used the owner's dataset convenience

Input: Proprietary dataset $\mathcal{D}_2$,
Target Model Φ trained on $\mathcal{D}_1$,
Prompt Generator Θ trained on $\mathcal{D}_2$,
Distance function Δ,
Number of candidate violations n.
Output: Set of candidate violations,
Set of prompts triggering candidates.

$\mathcal{C} \leftarrow$ empty list;
$\mathcal{P} \leftarrow$ empty list;
`dist` $\leftarrow$ empty list;
for $v \in \mathcal{D}_2$ **do**
  $k \sim p_\Theta(\cdot|v)$;
  $\hat{v} \sim p_\Phi(\cdot|k)$;
  `dist.append`$(\Delta(\hat{v}, v))$;
  $\mathcal{C}$`.append`(v);
  $\mathcal{P}$`.append`(k);
end
sort $\mathcal{C}$ and $\mathcal{P}$ accordingly with `dist`;
return $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ and $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$;

Algorithm 1 Finding candidate violations

The algorithm is provided with $\mathcal{D}_2$, the pre-trained models Φ and Θ, a suitable distance function Δ, and a number n of candidate violations to be flagged for further investigation. The output of the algorithm is a pair of sets containing the values in $\mathcal{D}_2$ that Φ can replicate and the prompts generated by Θ, respectively. For each element $v \in \mathcal{D}_2$, we use Θ to sample a prompt k that triggers Φ to generate $\hat{v}$. The distance $\Delta(v, \hat{v})$ between v and $\hat{v}$, v and k, are recorded in three lists: `dist`, $\mathcal{C}$, and $\mathcal{P}$, respectively. The lists $\mathcal{C}$ and $\mathcal{P}$ are then sorted based on `dist`, ensuring that the closest v to its corresponding $\hat{v}$, along with the related prompt k, appears first. Finally, the algorithm returns the first n elements of both $\mathcal{C}$ and $\mathcal{P}$.

The choice of Δ and n depends on the specific application, the interpretation of "violation", and the cost associated with each round of test. The function Δ may evaluate overall similarity between v and $\hat{v}$, assess the resemblance of specific segments, or focus on key structural or semantic features. Meanwhile, the choice of n involves balancing the need for sufficient evidence against the time and resources required to verify potential infringements.

4.1 Training the Prompt Generator

To find unauthorized usages of data, we need to train our Prompt Generator Θ. Its training process is illustrated in Fig. 2 and formalized in Algorithm 2.

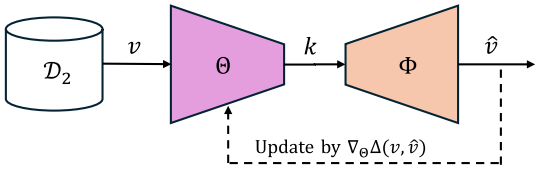

Fig. 2 Training process of the prompt generator. The model Θ generates the prompt k when provided with the value v that we want to inspect. The pre-trained model Φ then produces a value $\hat{v}$ in response to the input k. Finally, the model Θ is updated to minimize the distance Δ between v and $\hat{v}$

Input: Proprietary dataset $\mathcal{D}_2$,
Target Model Φ trained on $\mathcal{D}_1$,
Distance function Δ.
Output: Prompt Generator Θ

$\Theta \leftarrow$ random initialization;
while Θ *has not converged* **do**
  Sample mini-batch of m values $\{v_1, \ldots, v_m\}$ from $\mathcal{D}_2$;
  Update Θ by descending the stochastic gradient:
  $\nabla_\Theta \frac{1}{m} \sum_{i=1}^{m} \Delta\left(v_i, \Phi\left(\Theta\left(v_i\right)\right)\right)$
end

Algorithm 2 Training Θ

The algorithm takes as input the proprietary dataset $\mathcal{D}_2$, the model Φ pre-trained on the dataset $\mathcal{D}_1$, and the distance function Δ. The output of the algorithm is the trained model Θ, whose weights are adjusted to generate prompts that try to induce Φ to produce values closely matching those in $\mathcal{D}_2$. Specifically, starting with random initialization, the model Θ is iteratively updated through a mini-batch-based optimization process until convergence. Given Φ and $v \in \mathcal{D}_2$, we propose a loss function for Θ to minimize that penalizes based on the distance between the input and the output:

$$\text{loss}_\Theta(v) = \Delta\left(v, \Phi\left(\Theta\left(v\right)\right)\right), \tag{1}$$

where $\Theta(v)$ produces the prompt k that triggers Φ to generate $\hat{v}$. We emphasize that in our scenario, the model Φ is pre-trained and operates in inference mode. This setup aligns with typical black-box membership inference approaches, where $\mathcal{D}_1$ is unknown, the model is not fully disclosed, and the objective is to gather sufficient evidence of potential unauthorized use of proprietary data.

4.2 Speeding up the Training of the Prompt Generator

To implement the ProCAP framework, we focus on prompt-based generative models. However, training Θ via Algorithm 2 can be computationally expensive, especially when Θ is a Transformer-based architecture. Consequently, applying ProCAP to real-world scenarios could not always be feasible, mainly due to resource constraints (e.g., energy consumption) or scalability issues.

To overcome this limitation and improve the practicality of deploying our approach in realistic settings, we introduce a speed-up procedure. In contrast to the conventional paradigm of "learning from mistakes", our approach trains Θ to focus on identifying a subset of data that most clearly indicates unauthorized use of proprietary content. Rather than striving for generalization across all data, Θ is optimized by focusing on specific "slices" of data where its confidence is highest, i.e., the instances that minimize the prediction error.

The proposed speed-up procedure enhances the training efficiency by removing from the training set those instances associated with the highest loss values (or lowest, in the case of a gain function). Filtering out these less informative samples allows the model to concentrate on more meaningful examples, thereby streamlining the training process. The selection of the most relevant data reduces the overall training time while maintaining or enhancing the

performance of the model (over the filtered subset). This procedure is triggered when the generator stops acquiring new information from the data, as indicated by a plateau in its objective function.

To this aim, we adopted a robust approach by leveraging the Generalized Pareto Distribution (GPD) (Vignotto & Engelke, 2020) to estimate the tail of the error distribution. According to the Pickands–Balkema–De Haan theorem (Balkema & de Haan, 1974), the conditional distribution of a random variable X given a high threshold u, i.e., $(X \mid X > u)$, converges to the GPD as $u \to +\infty$. This is true when the law of X belongs to a variety of families, like exponential distributions (e.g., Gaussian and Laplacian), stable distributions (e.g., Cauchy and Levy), and power law distributions (Student-t and Pareto).

We exploit this property within ProCAP by devising a data reduction process that involves two steps. First, we fit a generalized Pareto distribution to the empirical error data. This involves estimating the parameters of the GPD that best represent the tail behavior of the error distribution. Second, we determine an appropriate threshold level based on the fitted GPD. In particular, we adopt the 80th percentile as the reference threshold. This choice is guided by a heuristic application of the Pareto principle (Wilkinson, 2006), which posits that approximately 20% of the causes are responsible for 80% of the effects. The entire process is formalized in Algorithm 3.

Input: Proprietary dataset $\mathcal{D}_2$,
Target Model Φ trained on $\mathcal{D}_1$,
Distance function Δ,
Patience $\mathcal{P}$,
Amplification weight $\kappa > 0$.
Output: Prompt Generator Θ

$\Theta \leftarrow$ random initialization;
$\boldsymbol{p} \leftarrow$ Uniform probability on $\mathcal{D}_2$;
while Θ *has not converged* **do**
  while $\mathcal{P}$ *holds* **do**
    Sample mini-batch of m values $\{v_1, \ldots, v_m\}$ from $\mathcal{D}_2$ according to $\boldsymbol{p}$;
    Update Θ by descending the stochastic gradient:
    $\nabla_\Theta \frac{1}{m} \sum_{i=1}^{m} \Delta(v_i, \Phi(\Theta(v_i)))$
  end
  $\mathcal{E} \leftarrow \{\Delta(v, \Phi(\Theta(v))) | v \in \mathcal{D}_2\}$;
  Compute τ according to Equation 2;
  For each $v \in \mathcal{D}_2$, update p_v according to Equation 3;
end

Algorithm 3 Optimized training Θ

The proposed speed-up procedure, similarly to Algorithm 2, outputs the prompt generator Θ and takes as input the proprietary dataset $\mathcal{D}_2$, the target model Φ, and the distance function Δ. Additionally, it requires a patience mechanism $\mathcal{P}$ (Prechelt, 2012) and an amplification factor κ.

The algorithm starts by randomly initializing Θ and assigning a uniform probability distribution $\boldsymbol{p}$ over all the elements in $\mathcal{D}_2$. It then iteratively updates Θ via a gradient-based optimization process until convergence. At each iteration, while the patience condition specified by $\mathcal{P}$ is satisfied, a mini-batch is sampled from $\mathcal{D}_2$ according to $\boldsymbol{p}$, and Θ is updated by descending the stochastic gradient of the loss in Eq. 1.

When the patience condition expires, a data reduction step is activated. The rationale is to discard samples associated with high loss values, under the assumption that they are unlikely to enable Θ to prompt Φ into generating artifacts resembling the original data. Even if some of these discarded samples represent actual violations, this is not an issue, as the goal is not to exhaustively detect every violation but to efficiently identify a sufficient number of them. This strategy, hence, reduces computational effort and accelerates training by focusing on the most informative instances.

Using the current configuration of Θ, the error distribution $\mathcal{E}$ is computed over $\mathcal{D}_2$, and a threshold τ is set as the error value nearest to the 80th percentile of a Generalized Pareto Distribution fitted to $\mathcal{E}$:

$$\tau = \arg\min_{\varepsilon \in \mathcal{E}} \left| \mathrm{GPD}(\mathcal{E})_{80\%} - \varepsilon \right| . \tag{2}$$

For each sample $v \in \mathcal{D}_2$, the selection probability p_v is then updated as:

$$p_v = \min \left[p_v, 1 - \frac{1}{1 + e^{-\kappa(\varepsilon_v - \tau)}} \right] , \tag{3}$$

where ε_v is the error value (loss value) corresponding to the sample v. This rule progressively reduces the sampling probability of high-error samples while preserving or maintaining those below the threshold. The complement of the logistic function, centered at τ and controlled by $\kappa > 0$, ensures a sharp transition around the threshold, with larger values of κ producing steeper slopes. Consequently, only the informative samples with $\varepsilon_v < \tau$ retain their probabilities over the epochs, while the others approach zero, excluding them from future mini-batches.

By setting the threshold near the 80th percentile of the GPD, the method conservatively excludes the top $\sim 20\%$ of errors, which are assumed to contribute most significantly to the overall prediction error. Removing these extreme values helps mitigate their potentially adverse effects on the analysis, leading to more reliable and robust results. In this way, the proposed approach strikes a balance between retaining enough data for meaningful analysis and discarding the most "problematic" instances. For this reason, the speed-up procedure is expected to ensure a balanced and effective data management strategy that positively affects the training process.

5 Experimental Evaluation

To quantify the ability of ProCAP of dealing with numerical information, we first analyze a scenario considering the unauthorized exploitation of IoT sensor data. This setting is of particular relevance since the pervasive adoption of IoT devices to control industrial deployments or drive urban intelligence frameworks may cause severe privacy leaks or unwanted user profiling (Yang et al., 2017; Ogonji et al., 2020). Our evaluation is guided by the following research questions:

- Can ProCAP generate prompts capable of detecting the use of proprietary data? (**RQ1**)
- How effective is the proposed speed-up procedure in balancing the detection of unau-

thorized data usage with the training time of the target model? (**RQ2**)
- What are the main strengths and limitations of ProCAP? (**RQ3**)

In the following, we provide an overview of the datasets, the implementation details, and the evaluation protocol. We then discuss about results and limitations of our approach.

5.1 Datasets and Evaluation Protocol

To address **RQ1** and **RQ2**, we consider four publicly available datasets that reflect industrial and realistic IoT scenarios, as described below.

- **Pump-Sensor**[1]: A sequential dataset related to failures in a water pump system, collected over a 5-month period with 1-minute sampling intervals. It includes raw numerical readings from 52 distinct sensors.
- **Elevator Failure**[2]: A dataset consisting of time-series observations from various IoT sensors used for predictive maintenance in elevator systems. Data is sampled at 4Hz during periods of peak and evening elevator usage in a building.
- **Electric Power Consumption**[3]: A dataset capturing electrical power consumption in the city of Tetouan, Morocco. Observations are recorded every 10 min.
- **Head Posture**[4]: A dataset comprising time series recorded from three inertial sensors, annotated with labels for various head postural movements. The signals are expressed in terms of Euler angles: Roll, Pitch, and Yaw (Severin, 2020a, b).

Datasets containing missing values or duplicate entries were pre-processed to remove nulls and redundancies prior to training. Moreover, each real dataset was partitioned into three equally-sized subsets, namely $\mathcal{D}_{tr}$, $\mathcal{D}_v$, and $\mathcal{D}_{up}$, with particular care taken to minimize overlap between $\mathcal{D}_{tr}$ and $\mathcal{D}_{up}$. The first two subsets, used for training and validating the generative model Φ, collectively form the dataset $\mathcal{D}_1$. A small portion of $\mathcal{D}_{tr}$, denoted as $\mathcal{D}_p$, is sampled to simulate proprietary data that has been improperly used to train Φ. This subset is then combined with $\mathcal{D}_{up}$, which represents proprietary data that remains unseen by Φ, to create the evaluation dataset $\mathcal{D}_2$.

To further assess the feasibility and limitations of our approach, and to address **RQ3**, we evaluated the behavior of ProCAP in a controlled setting using synthetically-generated data. To construct the synthetic datasets $\mathcal{D}_{tr}$, $\mathcal{D}_v$, and $\mathcal{D}_{up}$, we designed a controlled data generation process based on custom patterns defined through pairs of Gaussian distributions. Each synthetic pattern consists of two Gaussian components: the first is used to generate a sequence of points interpreted as the prompt, while the second produces a corresponding sequence representing the output that the generative model is expected to learn when conditioned on the given prompt.

Using this setup, we created two independent experimental scenarios, each comprising a full triplet of datasets $\mathcal{D}_{tr}$, $\mathcal{D}_v$, and $\mathcal{D}_{up}$. In the first scenario, denoted as **Synthetic**, we

[1] http://www.kaggle.com/datasets/nphantawee/pump-sensor-data.

[2] http://www.kaggle.com/datasets/shivamb/elevator-predictive-maintenance-dataset.

[3] http://www.kaggle.com/datasets/fedesoriano/electric-power-consumption.

[4] http://www.kaggle.com/datasets/ionutcristianseverin/head-posture-detection-based-on-3-inertial-sensors.

Table 1 Dataset description

Datasets	# Records	# Features	Sequence length	# Sequences	$\mathcal{D}_{tr}$	$\mathcal{D}_v$	$\mathcal{D}_{up}$	$\mathcal{D}_p$
Pump sensor	117,912	51	60	1,965	669	623	673	201
Elevator failure	93,882	8	60	1,564	486	483	595	146
Electric power consumption	52,416	8	30	1,747	598	547	602	180
Head posture	44,992	9	30	1,499	452	452	595	136
Synthetic	360,000	16	60	6,000	2,000	2,000	2,000	600
Synthetic-overlap	360,000	16	60	6,000	2,000	2,000	2,000	600

enforced a strict separation among the patterns used to generate $\mathcal{D}_{tr}$, $\mathcal{D}_v$, and $\mathcal{D}_{up}$. This means that the underlying Gaussian distributions used for prompts and outputs are entirely disjoint across the three subsets, ensuring a clear distinction between training, validation, and unseen proprietary data.

In contrast, in the second scenario, denoted as **Synthetic-Overlap**, we introduce a controlled level of pattern overlap between $\mathcal{D}_{tr}$ and $\mathcal{D}_{up}$. In this case, some of the distributions used to generate sequences in $\mathcal{D}_{up}$ are intentionally made similar to those used in $\mathcal{D}_{tr}$. The goal here is to simulate a more challenging and realistic condition where proprietary data resembles non-restricted content, allowing us to evaluate the sensitivity of ProCAP in distinguishing between truly unauthorized usage and benign similarity.

This twofold generation strategy allows us to test both the general effectiveness of our approach (in the Synthetic setting) and its limitations in more ambiguous scenarios (in the Synthetic-Overlap setting). Table 1 summarizes the statistics of the datasets.

5.2 Implementation Details and Evaluation Metrics

We implemented ProCAP by using the PyTorch[5] framework (Paszke et al., 2019). To support reproducibility, we have publicly released all the datasets and the source code necessary to replicate our experiments.[6]

Both Φ and Θ are implemented as Transformer models, following the original *base* architecture introduced by Vaswani et al. (2017). Specifically, either the encoder and decoder consist of 6 identical blocks, and the multi-head attention mechanism employs 8 parallel attention heads; the embedding dimension is set to 512. We used the Adam optimizer with a learning rate of 10^{-4}. Model Φ was trained for up to 1,000 epochs, and Θ for up to 500 epochs, both with early stopping (the patience is exhausted if the loss gain is less than or equal to 10^{-4} for 30 consecutive iterations). The best-performing versions, based on validation loss for Φ and training loss for Θ, were saved during training.

In experiments involving the speed-up procedure, we set the hyper-parameters, showcased in Algorithm 3, as follows: the patience $\mathcal{P}$ is exhausted if the loss gain is less than or equal to 10^{-3} for 2 consecutive iterations, and $\kappa = 100$ (simulating the step function); additionally, we decided to limit the sample elimination procedure: we retain at least one-third of the original size of each dataset. The distance metric used to quantify the prediction error was the Mean Squared Error, with "reduction" set to `sum`.

[5] http://www.pytorch.org.

[6] http://www.github.com/Angielica/WHAM-/tree/main/CAP.

To evaluate whether Θ can accurately detect the unauthorized use of proprietary data, it is important to emphasize that its effectiveness relies on the distinctiveness of the data under analysis. Specifically, proprietary data must exhibit a distribution sufficiently different from that of public or non-proprietary data. When this uniqueness is lacking, i.e., when proprietary and non-proprietary data distributions significantly overlap, discrimination becomes inherently difficult, often leading to *false positives*, where generic public data is mistakenly flagged as unauthorized.

This limitation is not specific to ProCAP but arises from fundamental statistical bounds. Let P and Q denote the distributions of proprietary and non-proprietary data, respectively. When these distributions exhibit a substantial overlap, statistical theory implies that no method can achieve error rates below the *minimum Bayes error*, given by

$$\mathrm{Err}_{\min} = \frac{1-\|P-Q\|_{TV}}{2},$$

where $\|P-Q\|_{TV}$ is the total variation distance (Devroye et al., 1996; LeCam, 1973; Tsybakov, 2008). As $\|P-Q\|_{TV}$ approaches 0, the achievable accuracy converges to the random guessing, reflecting a core limitation of statistical identifiability rather than a weakness of ProCAP itself. We investigate this phenomenon in greater detail in Sect. 5.3 through a dedicated experiment. It is worth noting that, in scenarios where such overlap is pronounced, the two distributions become statistically indistinguishable. In such cases, the notion of protection loses operational meaning, since the "protected" samples are not meaningfully different from the public data. Therefore, one may question the validity of claiming exclusive ownership of the data in the first place. If a supposedly proprietary dataset is almost indistinguishable from publicly available content, the notion of exclusive usage rights becomes more ambiguous and potentially less justifiable. However, in scenarios where the goal is to protect a *private variant* of otherwise public data, selecting an appropriate distance function that emphasizes the proprietary variations to be protected can mitigate this effect, and ProCAP explicitly exposes this function as a configurable parameter. *False negatives* arise when actual proprietary data goes undetected and is incorrectly classified as non-proprietary. However, obtaining an *exhaustive detection* is not the primary objective of our framework. Our approach is designed for ranking-based prioritization; therefore, the goal is not to apply a fixed classification threshold, but to produce an ordered list in which the most suspicious data instances appear at the top. In this setting, we aim to identify a representative, high-confidence subset of *true positives*. As such, the presence of some false negatives is acceptable, provided the top-ranked results are informative and actionable.

These considerations motivate our use of *Precision@n* and the *Area Under the Cumulative Gains Curve (AUC-Gain)* as evaluation metrics. These measures are specifically suited to scenarios where the goal is to effectively surface rare yet critical instances, such as unauthorized use of proprietary data.

The Precision@n is defined as:

$$\mathrm{Precision@}n = \frac{|\{v \in \mathcal{D}_p : \mathrm{rank}(v) \leq n\}|}{n}, \tag{4}$$

where $\mathcal{D}_p \subset \mathcal{D}_2$ denotes the proprietary data used without authorization, and $\mathrm{rank}(v)$ is the (1-based) position of v in the list sorted by ascending distance $\Delta(v, \Phi(\Theta(v)))$.

The AUC-Gain quantifies the area under the cumulative gains curve, which plots the proportion of correctly identified proprietary instances (true positives) on the y-axis against the proportion of data inspected from $\mathcal{D}_2$ on the x-axis, where samples are ordered by increasing

$\Delta(v, \Phi(\Theta(v)))$. A higher AUC-Gain indicates that ProCAP is more effective at identifying suspicious inputs early in the ranked list. Notably, this metric is widely regarded as one of the most informative tools for highlighting rare yet important data points in ranked outputs (Provost & Fawcett, 2013).

All experiments are repeated across 10 independent runs. We report average values, with statistical significance computed at the 95% confidence level.

5.3 Discussion of Results

Table 2 reports the results of the evaluation with and without the speed-up procedure. In response to **RQ1**, the findings indicate that ProCAP effectively generates prompts that force the model Φ to produce values used during its training, thereby uncovering unauthorized usage. Notably, in almost all the real-world datasets, the proprietary data consistently rank within the top 5 and 10 positions, showing a precision equal or near to 1. By increasing the value of *n*, we can observe a degradation of the performance. Furthermore, the model achieves an AUC-Gain close to 1, demonstrating its high effectiveness in distinguishing between proprietary and non-proprietary instances, in an imbalanced scenario. In other words, the model is nearly optimal in quickly identifying proprietary data.

We also point out that the generated prompt differs from the original prompt, which is not known to the data owner in the real scenario. Specifically, in Figs. 3a and c we show the absolute difference between the generated prompt and the original prompt by considering the top $n = 100$ detected data for the Synthetic and Head Posture dataset, respectively, where ProCAP achieves perfect results. To provide a meaning of these differences, we compare them to the distance separating the original (unknown) prompt and the real output. As we can notice, the confidence intervals do not overlap. This highlights that there is no one-to-one mapping between the prompt and its associated value. Instead, different prompts can yield the same result. Moreover, Figs. 3b and d reveal a significant disparity in the generated prompt and the generated output, particularly when compared to the absolute difference of the real prompt and the real output. This observation implies that there is no "copy & paste" or "rewrite what I'm writing" effect of Φ, which could affect the trustworthiness of the prompt generator Θ.

To better clarify the advantages of our approach, we evaluate ProCAP against an implicit "oracle" baseline. Given "white-box" access to the reference model Φ, a loss-based MIA

Table 2 Comparative analysis of ProCAP with and without the speedup procedure

Metric	Speedup	Datasets			
		Pump sensor	Elevator failure	Electric power consumption	Head posture
Precision@5	No	1.00 ± 0	1.00 ± 0	0.74 ± 0.30	1.00 ± 0
	Yes	1.00 ± 0	1.00 ± 0	0.66 ± 0.28	1.00 ± 0
Precision@10	No	1.00 ± 0	1.00 ± 0	0.68 ± 0.25	1.00 ± 0
	Yes	1.00 ± 0	1.00 ± 0	0.61 ± 0.23	1.00 ± 0
Precision@50	No	0.98 ± 0.02	0.52 ± 0.07	0.54 ± 0.15	1.00 ± 0
	Yes	0.97 ± 0.03	0.50 ± 0.07	0.50 ± 0.15	1.00 ± 0
Precision@100	No	0.91 ± 0.02	0.55 ± 0.02	0.48 ± 0.09	1.00 ± 0
	Yes	0.87 ± 0.03	0.55 ± 0.02	0.49 ± 0.08	1.00 ± 0
AUC-Gain	No	0.95 ± 0.01	0.92 ± 0.01	0.74 ± 0.02	0.96 ± 0.01
	Yes	0.93 ± 0.01	0.92 ± 0.01	0.72 ± 0.02	0.96 ± 0.01

strategy can be used to compare the likelihood of a sample to probe with the likelihood of the samples in the training set. The guiding principle is that the model assigns similar likelihood values to examples it has already seen, whereas unseen examples receive lower likelihoods (and therefore incur losses that are higher than the average training loss of Φ).

Based on this observation, we formulate the following research question. Let (k, v) be a generic example for which the reference loss $\ell_\Phi(k, v)$ can be computed and consider the two datasets $\mathcal{D}_p$ and $\mathcal{D}_{up}$. Define $\overline{\ell} = \text{avg}_{(k,v)\sim\mathcal{D}_{tr}}\ell_\Phi(k, v)$. We are then interested in how the quantities $\ell_\Phi(k_\Theta(v_p), v_p)$ and $\ell_\Phi(k_\Theta(v_u), v_u)$ relate to $\overline{\ell}$ when $(k_p, v_p) \sim \mathcal{D}_p$ and $(k_u, v_u) \sim \mathcal{D}_{up}$.

In principle, adherence to the classical white-box membership inference setting would lead to the relations $\ell_\Phi(k_\Theta(v_p), v_p) \approx \overline{\ell}$ and $\ell_\Phi(k_\Theta(v_u), v_u) > \overline{\ell}$. It is important to note, however, that this assumption is purely hypothetical, as it presumes access to both $\overline{\ell}$ and the ability to compute ℓ_Φ. Such information is not accessible when dealing with a proprietary reference model Φ, which is indeed the situation considered in our framework.

Figure 4 illustrates this comparative analysis throughout the training of Θ. The Y axis shows the reference loss $\overline{\ell}$ (red dashed line), together with the average values of $\ell_\Phi(k_\Theta(v_p), v_p)$ (blue line) and $\ell_\Phi(k_\Theta(v_u), v_u)$ (orange line). We observe that, as long as training advances, the former quantity converges toward $\overline{\ell}$, while the latter remains markedly divergent. This behavior resembles the approach used by loss-based MIA to distinguish

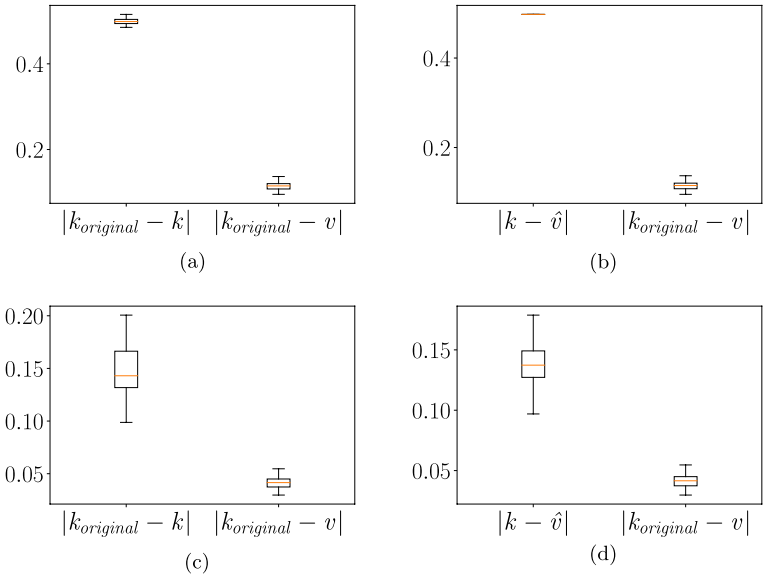

Fig. 3 Boxplots demonstrating the absence of overlaps between different distributions of synthetic and real-world data. (**a**) Distribution of the distances between the generated prompt, k, and the real prompt, k_{original}, compared against the real prompt-output pairs, (k_{original}, v), used as ground truth, for the Synthetic dataset. (**b**) Distribution of the distances between the generated prompt, k, and the corresponding generated output, $\hat{v}$, compared against the real prompt-output pairs, (k_{original}, v), for the Synthetic dataset. (**c**) Distribution of the distances between the generated prompt, k, and the real prompt, k_{original}, compared against the real prompt-output pairs, (k_{original}, v), used as ground truth, for the Head Posture dataset. (d) Distribution of the distances between the generated prompt, k, and the corresponding generated output, $\hat{v}$, compared against the real prompt-output pairs, (k_{original}, v), for the Head Posture dataset

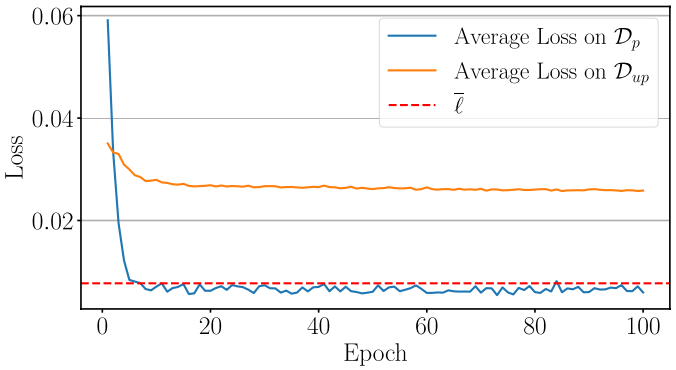

Fig. 4 Training of Θ on the Head Posture dataset

members from non-members. At the same time, it highlights that ProCAP can capture differences between members/non-members without having access to prompts or training data.

To answer **RQ2**, we compare the performance and the running times of ProCAP with and without the speed-up strategy. Specifically, the optimization procedure demonstrates to be highly effective in lowering training times without compromising the ability to detect data misuse. As reported in Table 2, all metrics show minimal to no degradation when the optimization is applied, for almost all the datasets. On the contrary, as illustrated in Fig. 5, the running times are substantially reduced across all datasets when the procedure is deployed.

We point out that, ProCAP requires that the information to be checked exhibits traits of uniqueness, i.e., its distribution is specific enough to be considered representative of proprietary material. Without this distinctiveness in the data, the model may strive to accurately differentiate between "original" and "derivative" contents. In such cases, it could either incorrectly attribute unauthorized usage to a generic piece of information or fail to distinguish between distributions that vary only slightly. This is illustrated in Figs. 6 and 7

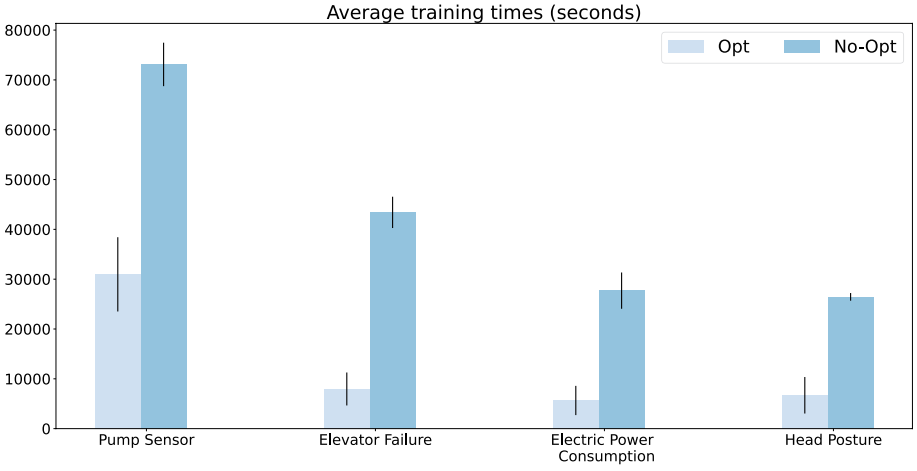

Fig. 5 Running times with (Opt) and without (No-Opt) the speed-up procedure

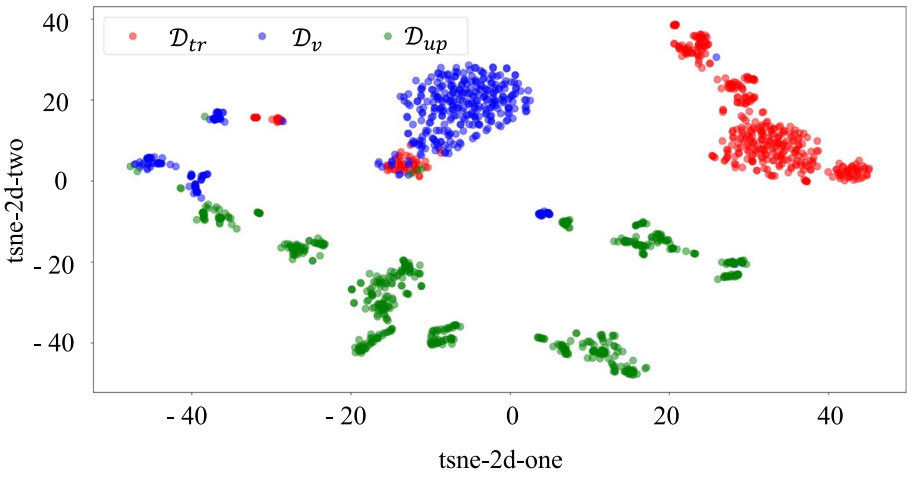

Fig. 6 t-SNE for Head Posture dataset

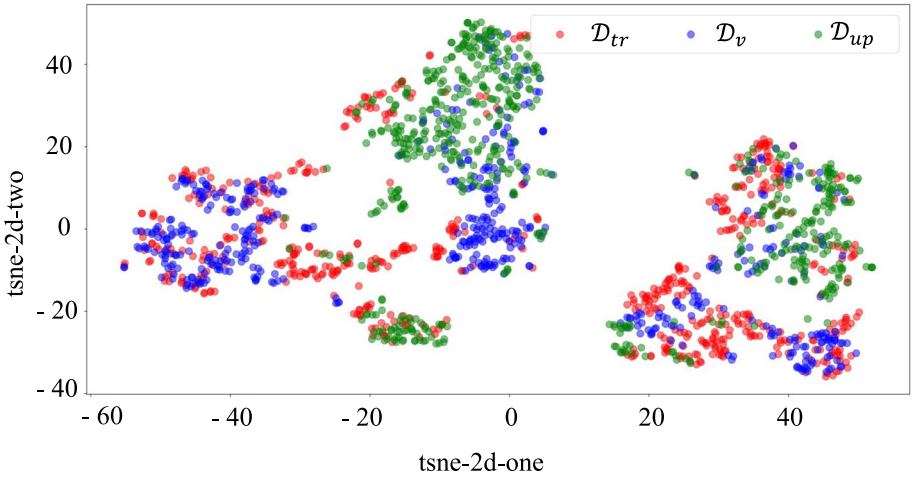

Fig. 7 t-SNE for Electric Power Consumption dataset

depicting a two-dimensional representation of datasets with different characteristics, where the x-axis corresponds to the first dimension and the y-axis to the second. The first dataset (Head Posture) in Fig. 6 is characterized by non-overlapping distributions, whereas Fig. 7 showcases data characterized by two overlapping (and hence indistinguishable) slices of data (Electric Power Consumption). The behavior of ProCAP in these situations is illustrated in Table 2. In fact, the low performance of the model on the Electric Power Consumption dataset can be explained by the overlap between the subset of proprietary data suspected to have been used during training and the portion assumed to be unseen.

To further strengthen this aspect and answer **RQ3**, we have also conducted in-vitro experiments. The results, reported in Table 3, show that on the Synthetic-Overlap data-

Table 3 Analysis of ProCAP with synthetic and synthetic-overlap data

Metric	Datasets	
	Synthetic	Synthetic-overlap
Precision@5	1.00 ± 0	0.38 ± 0.29
Precision@10	1.00 ± 0	0.36 ± 0.27
Precision@50	1.00 ± 0	0.33 ± 0.27
Precision@100	1.00 ± 0	0.31 ± 0.27
AUC-Gain	1.00 ± 0	0.49 ± 0.19

set, where the distributions of the proprietary and non-proprietary sets overlap, the model struggles to accurately identify data misuse due to the similarity between the two sets. In contrast, on the Synthetic dataset, where data is generated using separate distributions, for all the values of n, the model achieves perfect scores. Additionally, an AUC-Gain of 1.00 demonstrates the ability of ProCAP to perfectly distinguish proprietary content when no distributional overlap is present.

6 Extensibility and Domain Generalization

To evaluate whether ProCAP can identify unauthorized data usage in complex generative models, we conducted an additional round of experiments. Specifically, we aim to answer the following research question: Can ProCAP be extended to detect unauthorized data usage in complex generative models? (**RQ4**).

To this aim, we examine a scenario involving textual data. Misuse of textual information can involve the leakage of personal data, unauthorized reproduction or manipulation of content, and may result in the spread of misinformation, plagiarism, or violations of intellectual property and authorship rights (Weidinger et al., 2022). To address **RQ4** in this context, we investigate whether a text-generation model, denoted as Φ and trained to produce stories from summaries, has been exposed to a proprietary dataset of stories and news articles. In this setting, Θ is modeled as a summarization model: it processes a text (either a story or a news article) and generates a corresponding summary. This summary is then used as the prompt for Φ, enabling us to test whether Φ can reconstruct the original text. Specifically, if the content was included in the training data of Φ, the generated output is expected to closely align with the original. On the contrary, if the text was unseen during training, the output is expected to differ.

For this experiment, we implemented Φ as a GPT-2 model[7] trained from scratch on a dataset of 6, 234 (summary, story) pairs. The stories were sampled from the TinyStories dataset (Eldan & Li, 2023), which is a synthetic corpus of short stories for children. The summaries were generated using BART (Lewis et al., 2019), which is a pre-trained model for abstractive summarization. After the training phase, GPT-2 was frozen and used in inference mode within the ProCAP framework, where only the summarizer Θ was subject to optimization. For Θ, we adopted a pre-trained summarization model from the T5 family,[8] which we fine-tuned to produce summaries that guide Φ toward generating outputs as close

[7] https://github.com/karpathy/minGPT.

[8] https://huggingface.co/Falconsai/text_summarization.

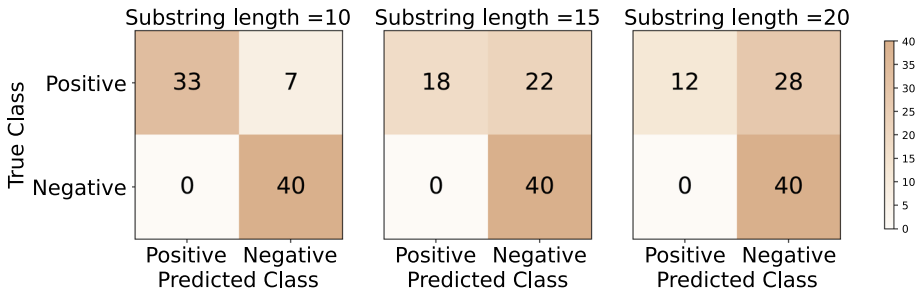

Fig. 8 Confusion matrices for substring overlap detection at different values of l

as possible to the original texts. The optimization of Θ followed the Proximal Policy Optimization strategy,[9] with the reward defined as the ROUGE-L (Lin & Och, 2004) similarity between the original text v and the reconstructed text $\hat{v} = \Phi(\Theta(v))$.

The training set consisted of 40 texts that had been included in the training data of Φ and 40 texts that had not. The unseen textual pieces were sampled from the CNN/DailyMail dataset.[10] To measure whether Φ reproduces data from its training set, we compared substrings of length l extracted from the original and generated texts. A pair was classified as a true positive if at least one l-length substring from the original text appeared in the generated counterpart. Otherwise, it was counted as a false negative (with the opposite logic applied to unseen texts).

Figure 8 reports the confusion matrices obtained for l values ranging from 10 to 20. The results suggest that smaller values of l increase the sensitivity of the method: for instance, at $l = 10$, most positive cases are correctly identified (33 true positives against only 7 false negatives). In contrast, for larger values such as $l = 15$ and $l = 20$, recall decreases noticeably as the false negatives increased. It is worth emphasizing that, across all configurations, ProCAP produces no false positives.

The qualitative results in Table 4 illustrate how the substring-matching criterion operates in practice. Segments highlighted in green correspond to exact matches between the original and generated text, showing that the model actually reproduces contiguous phrases identically. Segments in orange indicate near matches where subportions of the text differ. Segments in brown indicate words from the original text that were omitted in the generated output. Finally, red marks passages with no overlap, where the model either diverges significantly from the source text or invents new content. As the examples show, for stories already seen by Φ, the model can generate coherent passages with many exact matches. In contrast, for unseen stories, the outputs drift away from the original, often reusing stylistic elements from the training set but failing to preserve factual consistency.

[9] https://huggingface.co/docs/trl/v0.7.11/en/ppo_trainer.

[10] https://huggingface.co/datasets/abisee/cnn_dailymail.

Table 4 Examples of language-based model outputs

Original text	Generated Text
Grandma was looking around the store. She wanted to find something special for her grandbaby. Suddenly, she saw a new mint dress. It was so pretty and just the right size. Grandma smiled and said to the shopkeeper, "I'll take it!" The shopkeeper put the dress in a bag and gave it to Grandma. Grandma thanked her and took the bag. Grandma was so excited to give her grandbaby the new dress. [...]	Grandma was looking around the store. She wanted something special for her Nick. She saw a new mint dress. It was so pretty and just the right size. Grandma smiled and said to the shopkeeper, "I'll take it!" Grandma smiled and said to the shopkeeper, "I'll take it!" Grandma was so excited to see the new mint. She quickly agreed and took it home. Grandma was so excited to give her hurt her new mint. [...]
Here are five great ways to enjoy your summer. Lazing in a hammock is one of the best ways to spend a summer evening. Best way to cut jeans into shorts – What better way to declare the start of summer? The key to cutting off jeans is not to go too short too soon. Slip on the jeans and mark the desired length on one leg with chalk. "Take them off, fold the leg at the mark, and iron the fold," [...]	One day, a little boy was running around with his parents. He was so excited to go on the beach. "Mommy, can I go on the beach?" asked the little boy. "It's a great day, honey. It quickly won't get hurt." His parents replied, "We will go for a new adventure, but we will permit you to be careful and stay on the beach." The little boy was a bit excited and quickly hopped off his car and followed him. [...]

Legend: green = exact matches (identical contiguous phrases); orange = near matches differing by a single token; brown = omitted words from the original; red = non-matching or novel content

7 Conclusions

In this work, we introduced ProCAP, a model-agnostic, black-box framework designed to detect the unauthorized use of proprietary data in prompt-based generative models by leveraging the generation of targeted prompts. The framework operates without requiring access to the internal parameters or training process of the target model, making it suitable for real-world applications where only black-box interactions are possible.

To demonstrate its effectiveness, we evaluated ProCAP on both real-world and synthetically-generated datasets. Specifically, we tested its ability to identify unauthorized models trained using proprietary data from four IoT-based scenarios as well as scenarios involving text data. The experimental results consistently confirmed the effectiveness of the proposed method. In particular, ProCAP showed good performance in scenarios where there is a clear statistical separation between proprietary and non-proprietary data distributions. For instance, in the Head Posture and Synthetic datasets, both characterized by non-overlapping data distributions, ProCAP achieved near-perfect results, with AUC-Gain scores of 0.96 and 1.0, respectively. These results validate the ability of the framework to prioritize the identification of violations when proprietary content is uniquely distinguishable.

Since exploring large datasets could be computationally expensive, we introduced a speed-up procedure that optimizes the training of ProCAP by focusing on the most informative data points. Our results demonstrate that this optimization significantly reduces computational costs without compromising detection accuracy, thus making the deployment of ProCAP feasible for large-scale or resource-constrained settings. In this vein, a major part of our ongoing research aims at evaluating large-scale foundation models as well as the definition of advanced speed-up procedures to further enhance the scalability of ProCAP.

We point out that detecting unauthorized data usages may require a relevant amount of interactions with the model under test. Therefore, when used in the wild, ProCAP could require to interact with a model Φ deploying some countermeasures against aggressive queries. For instance, this is the case of many commercial frameworks returning truncated outputs, implementing rate-limiting policies, or embedding additional data to track the source of the unwanted load of requests (Li et al., 2023). In general, mitigation techniques are scenario-dependent or implemented in the underlying software layers. As a result, many AI-as-a-Service architectures prefer to limit the number of APIs calls rather than manipulating their outputs (see, e.g., Shi et al. (2018)). Thus, ProCAP does not explicitly consider a specific mitigation strategy but can be straightforwardly adapted to handle the presence of rate-limiting approaches. Therefore, part of our future research is devoted to split the monolithic Θ used by ProCAP over multiple parallel and simpler instances that operate independently to not trigger countermeasures.

Acknowledgements This research was partially funded by Project WHAM! - Watermarking Hazards and novel perspectives in Adversarial Machine learning (B53D23013340006), by Project RAISE - Robotics and AI for Socio-economic Empowerment (ECS00000035), by Project STRIVE/URAN - Advanced Approaches for Transitions in Urban Environments, and by the Project SERICS - Security and Rights in the Cyberspace (PE00000014) under the NRRP MUR program funded by the European Union – Next Generation EU.

Author Contributions - Conceptualization: D.G., A.L., E.R.; - Methodology: D.G., A.L., E.R.; - Formal analysis and investigation: D.G., A.L., E.R., F.D.,G.M.; - Writing - original draft preparation: D.G., A.L., E.R., L.C.; - Writing - review and editing: L.C., G.M.; - Funding acquisition: L.C., F.D., G.M.; - Resources: D.G., A.L.; - Supervision: L.C., F.D., G.M.

Funding Open access funding provided by Consiglio Nazionale Delle Ricerche (CNR) within the CRUI-CARE Agreement. Partially funded by: Project WHAM! - Watermarking Hazards and novel perspectives in Adversarial Machine learning (B53D23013340006); Project RAISE - Robotics and AI for Socio-economic Empowerment (ECS00000035); Project STRIVE/URAN - Advanced Approaches for Transitions in Urban Environments; Project SERICS - Security and Rights in the Cyberspace (PE00000014).

Data Availability The real datasets are publicy available at their respective link provided within the manuscript while the code to generate the synthetic data is available at https://github.com/Angielica/WHAM-/tree/main/CAP.

Code Availability https://github.com/Angielica/WHAM-/tree/main/CAP.

Declarations

Conflict of interest The authors declare no conflict of interest.

Ethical Approval and Consent to Participate Not applicable.

Consent for Publication Not applicable.

References

Balkema, A. A., & de Haan, L. (1974). Residual life time at great age. *The Annals of Probability, 2*(5), 792–804.

Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., & Tramer, F. (2022). Membership inference attacks from first principles. In: *2022 IEEE symposium on security and privacy*, IEEE.

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramer, F., & Wallace, E. (2023a). Extracting training data from diffusion models. In: *USENIX security symposium*.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., & Zhang, C. (2023b). Quantifying memorization across neural language models. In: *The eleventh international conference on learning representations*.

Carlini, N., Tramér, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., & Raffel, C. (2021). Extracting training data from large language models. In: *USENIX security symposium*.

Chang, K., Cramer, M., Soni, S., & Bamman, D. (2023). Speak, memory: An archaeology of books known to ChatGPT/GPT-4. In: *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7312–7327.

Chen, D., Yu, N., Zhang, Y., & Fritz, M. (2020). GAN-Leaks: A taxonomy of membership inference attacks against generative models. In: *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pp. 343–362.

Chen, Z. (2024). Catch me if you can: Detecting unauthorized data use in training deep learning models. In: *Proceedings of the 2024 on ACM SIGSAC conference on computer and communications security*, pp. 5098–5100.

Deistler, M., & Scherrer, W. (2022). *Time series models*. Lecture Notes in Statistics.

Devroye, L., Györfi, L., & Lugosi, G. (1996). *A probabilistic theory of pattern recognition, stochastic modelling and applied probability*, vol 31.

Eldan, R., & Li, Y. (2023). *TinyStories: How small can language models be and still speak coherent English?* arXiv:abs/2305.07759.

Fischer, J. E. (2023). Generative AI considered harmful. In: *Proceedings of the 5th international conference on conversational user interfaces*, pp. 1–5.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM, 63*(11), 139–144.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. In: *Proceedings of the 34th conference on neural information processing systems*.

Huang, Z., Gong, N. Z., & Reiter, M. K. (2024). A general framework for data-use auditing of ML models. In: *Proceedings of the 2024 on ACM SIGSAC conference on computer and communications security*, pp. 1300–1314.

Jin, H., Liu, M., Dodhia, K., Li, Y., Srivastava, G., Fredrikson, M., Agarwal, Y., & Hong, J. I. (2018). Why are they collecting my data? Inferring the purposes of network traffic in mobile apps. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies, 2*(4), 1–27.

Kim, J., Kim, M., & Mozafari, B. (2023). Provable memorization capacity of transformers. In: *The eleventh international conference on learning representations*.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In: *Proceedings of the international conference on learning representations*.

LeCam, L. (1973). Convergence of estimates under dimensionality restrictions. *The Annals of Statistics, 1*(1), 38–53.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed ,A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). *BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*. arXiv:abs/1910.13461.

Li, B., Wei, Y., Fu, Y., Wang, Z., Li, Y., Zhang, J., Wang, R., & Zhang, T. (2024a). *Towards reliable verification of unauthorized data usage in personalized text-to-image diffusion models*. arXiv:abs/2410.10437.

Li, H., Deng, G., Liu, Y., Wang, K., Li, Y., Zhang, T., Liu, Y., Xu, G., Xu, G., & Wang, H. (2024b). *Digger: Detecting copyright content mis-usage in large language model training*. arXiv:abs/2401.00676.

Li, Z., Wang, C., Wang, S., & Cuiyun, G. (2023). Protecting intellectual property of large language model-based code generation APIs via watermarks. In: *Proceedings of the 2023 ACM SIGSAC conference on computer and communications security*, pp. 2336–2350.

Lin, C. Y., & Och, F. J. (2004). Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In: *Proceedings of the 42nd annual meeting of the association for computational linguistics*, pp. 605–612.

Liu, Y., Huang, J., & Li, Y., et al. (2025). Generative AI model privacy: A survey. *Artificial Intelligence Review,58*(33).

Maini, P., Jia, H., Papernot, N., & Dziedzic, N. (2024). *LLM dataset inference: Did you train on my dataset?* arXiv:abs/2406.06443.

Maini, P., Yaghini, M., & Papernot, N. (2021). Dataset inference: Ownership resolution in machine learning. In: *International conference on learning representations*.

Meurisch, C., & Mühlhäuser, M. (2021). Data protection in AI services: A survey. *ACM Computing Surveys, 54*(2), 1–38.

Nasr, M., Shokri, R., & Houmansadr, A. (2019). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: *2019 IEEE symposium on security and privacy*, pp 739–753.

Ogonji, M. M., Okeyo, G., & Wafula, J. M. (2020). A survey on privacy and security of internet of things. *Computer Science Review, 38*, Article 100312.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., & Chintala, S. (2019). *PyTorch: An imperative style, high-performance deep learning library*. In: *33rd Conference on neural information processing systems*.

Prechelt, L. (2012). Early stopping – But when? In: *Neural networks: Tricks of the trade: Second Edition*. pp. 53–67

Provost, F., & Fawcett, T. (2013). *Data science for business*. O'Reilly.

Severin, I. C. (2020a). Head posture monitor based on 3 IMU sensors: Consideration toward healthcare application. In: *Proceedings of the international conference on e-health and bioengineering*, pp. 1–4.

Severin, I. C. (2020b). The head posture system based on 3 inertial sensors and machine learning models: Offline analyze. In: *Proceedings of the 3rd international seminar on research of information technology and intelligent systems*, pp. 672–676.

Shi, Y., Sagduyu, Y. E., Davaslioglu, K., & Li, J. H. (2018). Active deep learning attacks under strict rate limitations for online API calls. In: *2018 IEEE international symposium on technologies for homeland security*, pp. 1–6.

Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership inference attacks against machine learning models. In: *2017 IEEE symposium on security and privacy*, pp. 3–18.

Sirisuriya, S. D. S. (2023). Importance of web scraping as a data source for machine learning algorithms-review. In: *2023 IEEE 17th international conference on industrial and information systems*, pp. 134–139.

Sobel, B. L. (2017). Artificial intelligence's fair use crisis. *Colum JL & Arts, 41*, 45.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In: *International conference on machine learning*, pp. 2256–2265.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., & Goldstein., T. (2023). Diffusion art or digital forgery? Investigating data replication in diffusion models. In: *Conference on computer vision and pattern recognition*, pp. 6048–6058.

Tang, R., Feng, Q., Liu, N., Yang, F., & Hu, X. (2023). Did you train on my dataset? Towards public dataset protection with clean label backdoor watermarking. *SIGKDD Explor Newsl, 25*(1), 43–53.

Tsybakov, A. B. (2008). *Introduction to nonparametric estimation*, 1st edn.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). Attention is all you need. In: *Proceedings of the 31st conference on neural information processing systems*, pp. 5998–6008.

Vignotto, E., & Engelke, S. (2020). Extreme value theory for anomaly detection-the GPD classifier. *Extremes, 23*(4), 501–520.

Wei, J. T. Z., Wang, R. Y., & Jia, R. (2024). *Proving membership in LLM pretraining data via data watermarks*. arXiv: abs/2402.10892.

Weidinger, L., Uesato, J., Rauh, M., Griffin, C., Huang, P. S., Mellor, J., Glaese, A., Cheng, M., Balle, B., Kasirzadeh, A. & Biles, C. (2022). Taxonomy of risks posed by language models. In: *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pp. 214–229.

Wilkinson, L. (2006). Revising the Pareto chart. *The American Statistician, 60*(4), 332–334.

Wu, H., & Cao, Y. (2025). *Membership inference attacks on large-scale models: A survey*. arXiv:abs/2503.19338.

Yang, Y., Wu, L., Yin, G., Li, L., & Zhao, H. (2017). A survey on security and privacy issues in internet-of-things. *IEEE Internet of Things Journal, 4*(5), 1250–1258.

Authors and Affiliations

Daniela Gallo[1,2] · Angelica Liguori[1] · Ettore Ritacco[3] · Luca Caviglione[4] · Fabrizio Durante[2] · Giuseppe Manco[1]

✉ Daniela Gallo
daniela.gallo@icar.cnr.it

Angelica Liguori
angelica.liguori@icar.cnr.it

Ettore Ritacco
ettore.ritacco@uniud.it

Luca Caviglione
luca.caviglione@ge.imati.cnr.it

Fabrizio Durante
fabrizio.durante@unisalento.it

Giuseppe Manco
giuseppe.manco@icar.cnr.it

[1] Institute for High Performance Computing and Networking, Italian National Research Council, Via P. Bucci 8-9/C, Rende 87036, Italy

[2] Department of Mathematics and Physics "Ennio de Giorgi", University of Salento, Via per Arnesano, Lecce 73100, Italy

[3] University of Udine, Via Palladio 8, Udine 33100, Italy

[4] Institute for Applied Mathematics and Information Technologies, Italian National Research Council, Via de Marini 6, Genova 16149, Italy