DA-GA3001: Modern Topics in Statistical Learning Theory Spring 2024

Lecture 11 Watermark and Copyright - Apr 11, 2024

Prof. Qi Lei *Scribe: Hanyuan Zhang, Yirong Bian*

Overview

Today, we will explore **Unit 3: AI Safety**, a crucial topic due to the potential risks associated with data leakage and the dissemination of malicious content, which are not aligned with human values. This unit covers several essential aspects of AI safety, outlined as follows:

1. **Copyright:** Today's focus.
2. **Privacy:** Gautam CS860
3. **Alignment:** Elad Hazan COS598

- **Disclaimer**: The chosen content to be covered in this lecture is not based on academic impact but on new trending topics with some theoretical flavor.
- **Copyright**: This lecture primarily addresses copyright issues about generative models. Currently, most countries lack specific regulations regarding the use of copyrighted data in these contexts. Often, using copyrighted material to train models does not contravene existing legal frameworks. However, it is crucial to ensure that the outputs of language models do not reproduce copyrighted data. We want the models to learn the underlying distribution of the data without merely memorizing it. However, if a model predominantly reproduces the exact data from its training set, which may be copyrighted, this could pose legal issues.

Next, we are going to discuss two ways to eliminate the above problems.

Method1: Watermark

We need to be able to trace back the source of data from a specific generator. This helps us to deal with issues in fake news and commercial use about generated images.

- **Watermarking:** Add distinct patterns into the output data to enable traceability back to the source of the generated content.

Adding a watermark involves two main techniques. The first involves the direct addition of a watermark to the data, which can either be learning-based or non-learning-based. The learning-based elusive watermark marks are undetectable by the human eye and require a machine for detection. The encoder-based injection is often utilized to achieve this aim. The second involves introducing a watermark to the image-generating procedure.

1(a) Preliminary

Before examining watermark implementation methods, it's essential to review some prerequisite knowledge.

Preliminary1: Hypothesis Testing

The objective of hypothesis testing is to either accept or reject theories based on the data. The central question asked is whether the data or observations are adequate to support a particular hypothesis.

For instance, given an image, and we are tasked with determining if the acquired image is clean (null) or branded with a watermark (alternative). The method of hypothesis testing commonly used involves a likelihood ratio test that compares the two hypotheses. For instance, when

$$\frac{P(\text{observation/null})}{P(\text{observation/alternative})} \to 1$$

it's hard to distinguish the two cases.

Besides, there are two types of errors:

- **Type I error:** $P(\text{null/inferred watermarked})$
- **Type II error:** $P(\text{alternative/inferred clean})$

As an illustration, we might want to infer whether "Patient I" has been included in the training dataset. The null hypothesis is it has been included in the training dataset and the alternatives are it hasn't been included. In this case, we have:

- **Type I error:** $P(\text{null/inferred patient I used})$
- **Type II error:** $P(\text{alternative/inferred patient I not used})$

We will discuss this privacy issue more next time.

Preliminary2: Generator

Below are four types of commonly used generators. We use X to represent images, Z for the latent code, and g for the generator:

- **GAN:** $Z \mapsto g(Z)$, where Z could be random noise or latent code.
- **VAE:** Consists of an encoder and a decoder. The encoder $q_\phi(Z|X)$ maps input X to a latent representation Z, and the decoder $p_\theta(X|Z)$ reconstructs X from Z. The process is modeled as $Z \mapsto g(Z)$ for the generative (decoder) part.

- **Normalizing Flow:** $Z \mapsto f(Z)$, where $Z \sim N(0, \sigma^2 I)$. Here we use f instead of g because f is invertible, which allows us to characterize the probability density for X in the data space: $P_X(x) = p_Z(f^{-1}(x)) \cdot |\det J_{f^{-1}}(x)|$, where $J_{f^{-1}}$ is the Jacobian of f^{-1}.
- **Diffusion Model:** $X_0 \leftrightarrow X_1 \leftrightarrow X_2 \leftrightarrow \cdots Z$. It describes a process that starts from data X_0 and gradually adds noise step-by-step until it reaches a Gaussian distribution Z. The reverse process models the generative pathway, converting noise back to data through a series of denoising steps.

1(b) Specific watermarking methods on images

We have two methods as examples here.

example1: Learning-Based (please refer to [1] "The Stable Signature: Rooting Watermarks in Latent Diffusion Models")

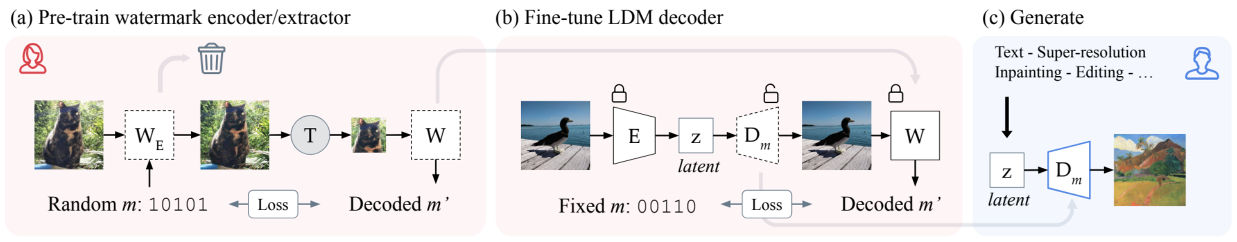
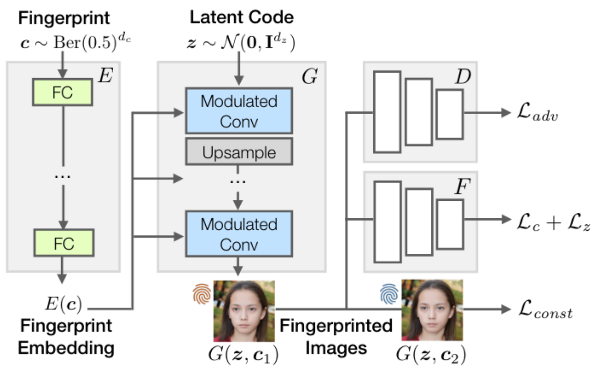
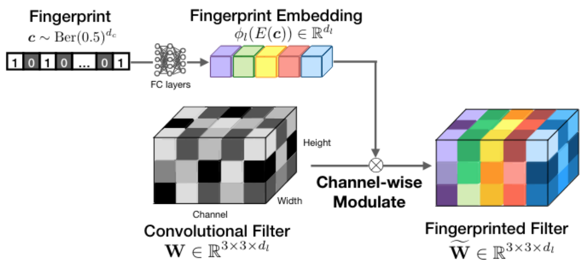


Figure 1: (a) Pre-train a watermark encoder W_E and extractor W, to extract binary messages. (b) Fine-tune the decoder D of the LDM's auto-encoder with a fixed signature m such that all the generated images (c) lead to m through

example2: Add to Generator (please refer to [2] "Responsible Disclosure of Generative Models Using Scalable Fingerprinting")



(a) Pipeline.


Fingerprint
$c \sim \mathrm{Ber}(0.5)^{d_c}$
1 0 1 0 ... 0 1
FC layers
Fingerprint Embedding
$\phi_l(E(c)) \in \mathbb{R}^{d_l}$
Height
Width
Channel
Convolutional Filter
$\mathbf{W} \in \mathbb{R}^{3\times3\times d_l}$
Channel-wise Modulate
Fingerprinted Filter
$\widetilde{\mathbf{W}} \in \mathbb{R}^{3\times3\times d_l}$

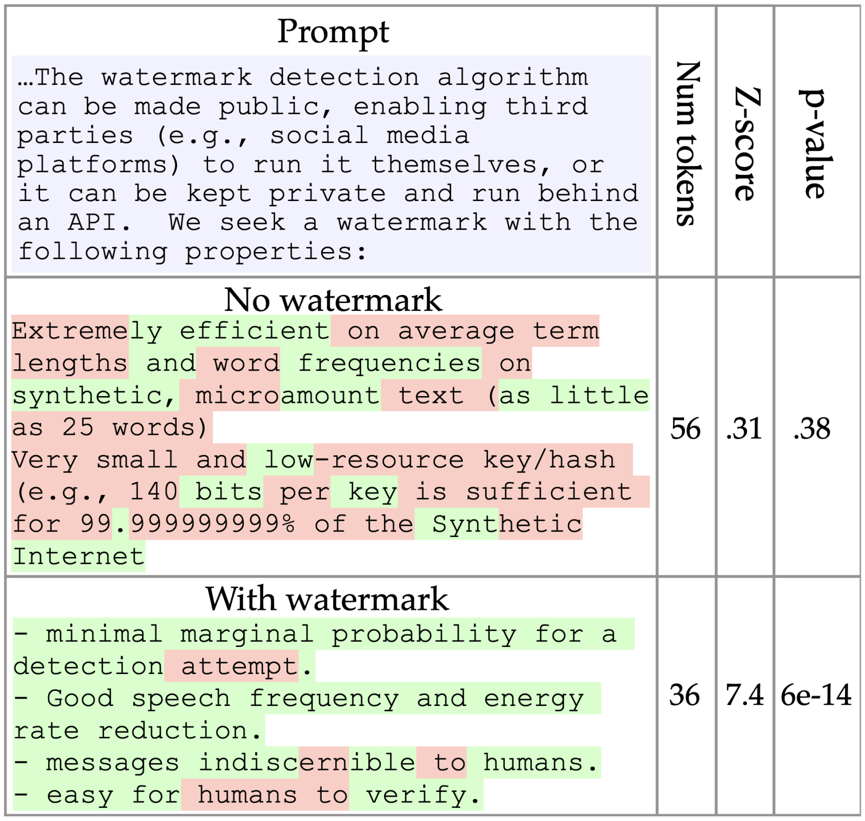
(b) Modulated convolutional layer.

Figure 2: The diagrams of the fingerprinting pipeline and the modulated convolutional layer

As the original essay described: "We use the fingerprint embedding from the encoder to modulate each convolutional filter of the generator (b), and try to decode this fingerprint from the generated images. We jointly train the fingerprint auto-encoder and GAN with our fingerprint related losses and the original adversarial loss".

1(c) Specific watermarking methods on text

The watermark works by selecting a randomized set of "green" tokens before a word is generated, and then softly promoting the use of green tokens during sampling. The watermark can be embedded with negligible impact on text quality and can be detected using an efficient open-source algorithm without access to the language model API or parameters [3].



Prompt	Num tokens	Z-score	p-value
…The watermark detection algorithm can be made public, enabling third parties (e.g., social media platforms) to run it themselves, or it can be kept private and run behind an API. We seek a watermark with the following properties:			
No watermark Extremely efficient on average term lengths and word frequencies on synthetic, microamount text (as little as 25 words) Very small and low-resource key/hash (e.g., 140 bits per key is sufficient for 99.999999999% of the Synthetic Internet	56	.31	.38
With watermark - minimal marginal probability for a detection attempt. - Good speech frequency and energy rate reduction. - messages indiscernible to humans. - easy for humans to verify.	36	7.4	6e-14

Figure 3: Outputs of a language model, both with and without the application of a watermark. The watermarked text, if written by a human, is expected to contain 9 "green" tokens, yet it contains 28. The probability of this happening by random chance is $\approx 6 \times 10^{-14}$, leaving us extremely certain that this text is machine-generated.

The algorithm to generate the watermark is as follows and we detect the watermark by testing the hypothesis.

H_0 : `The text sequence is generated with no knowledge of the red list rule.`

with

$$\frac{P(\text{observation} \mid \text{watermarked LLM})}{P(\text{observation} \mid \text{clean LLM})} \propto (1+\delta)^t,$$

where t is the length of the text sequence and δ is the hardness parameter.

Algorithm 1 Text Generation with Soft Red List

```
    Input: prompt $s^{(-N_p)} \cdots s^{(-1)}$, green list size $\gamma \in (0,1)$, hardness parameter $\delta > 0$
1: for $t = 0, 1, \cdots$ do
2: Apply the language model to prior tokens $s^{(-N_p)} \cdots s^{(t-1)}$ to get a logit vector $l^{(t)}$ over the vocabulary.
3: Compute a hash of token $s^{(t-1)}$, and use it to seed a random number generator.
4: Using this random number generator, randomly partition the vocabulary into a "green list" $G$ of size $\gamma|V|$, and a "red list" $R$ of size $(1-\gamma)|V|$.
5: Add $\delta$ to each green list logit. Apply the softmax operator to these modified logits to get a probability distribution over the vocabulary.
6: Sample the next token, $s^{(t)}$, using the watermarked distribution $\hat{p}^{(t)}$.
```

Method2: "Un"memorize Copyrighted Data

Trained generative models might produce samples similar to copyrighted data in their training set. Define *near access-freeness* (NAF) to ensure models don't output data too close to the copyrighted set, even if it's included in training. A model is k-NAF if its output differs by at most k bits from a model that never accessed the copyrighted data [4].

Setting

Start with an algorithm $\mathcal{A} : \mathcal{D} = \{z_1, \cdots, z_N\} \to p(\cdot \mid \cdot)$ mapping a dataset to a conditional generative model. Namely, given a prompt x, the model outputs y with probability $p(y \mid x)$ and $\mathcal{C} \subseteq \mathcal{D}$ is the copyrighted data in the dataset. That is, for $p = \mathcal{A}(\mathcal{D})$, the concern is that for some prompt x and copyrighted material $C \in \mathcal{C}$, it holds that $y \sim p(\cdot \mid x)$ will be similar to C with non-trivial probability. The goal is to devise a procedure where this is not the case.

Definition (k-Near Access-Free). Let $\mathcal{C}$ a set of datapoints; let safe: $\mathcal{C} \to \mathcal{M}$; and let Δ be a divergence measure between distributions. We say that a generative model p is k_x-near access-free ($k_x - NAF$) on prompt $x \in \mathcal{X}$ with respect to $\mathcal{C}$, safe, and Δ if for every $C \in \mathcal{C}$

$$\Delta\left(p(\cdot \mid x) \| \operatorname{safe}_C(\cdot \mid x)\right) \leq k_x.$$

We say p is k-NAF if the above holds for all $x \in \mathcal{X}$ with $k_x \leq k$.

Lemma (Event bound, max-KL). Suppose model p is $k_x - NAF$ on prompt x with respect to $\mathcal{C}$, safe, $\Delta = \Delta_{\max}$. Then for any $C \in \mathcal{C}$ and any event $\mathcal{E}$,

$$p(\mathcal{E} \mid x) \leq 2^{k_x} \cdot \operatorname{safe}_C(\mathcal{E} \mid x).$$

CP-Δ Procedure

We develop the following procedure. In the sharded-safe algorithm, we ensure that a copyrighted data is only seen by one of the q_1, q_2 models. And by applying such procedure, we obtain that provided q_1 and q_2 have total variation distance only non-trivially bounded away from 1 and if the probability of a fortuitous copy is small, then p will also copy with only a small probability.

Algorithm 2 Sharded-Safe

```
    **Input:** Dataset $\mathcal{D}$
    **Shard** $\mathcal{D}$**:** Partition $\mathcal{D}$ into two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$
    **Learning** $\mathcal{D}$**:** $q_1 = \mathcal{A}(\mathcal{D}_1), q_2 = \mathcal{A}(\mathcal{D}_2)$
    **Return:** $q_1, q_2$, and the function

  sharded-safe $(C) := q_i$, where $C \notin \mathcal{D}_i$
```

Algorithm 3 CP-Δ: Copy Protection w.r.t. divergence Δ

```
    **Input:** Dataset $\mathcal{D}$, and divergence $\Delta \in \{\Delta_{\max}, \Delta_{\mathrm{KL}}\}$.
    **Learning:** Call sharded-safe($\mathcal{D}$) to obtain $q_1$ and $q_2$.
    **Return:** the model $p$, where:
```

$$p(y \mid x) = \begin{cases} \frac{\min\{q_1(y|x), q_2(y|x)\}}{Z(x)} & \text{if } \Delta = \Delta_{\max} \\ \frac{\sqrt{q_1(y|x) \cdot q_2(y|x)}}{Z(x)} & \text{if } \Delta = \Delta_{\mathrm{KL}}. \end{cases}$$

Theorem (CP $-\Delta$). Let p be the model returned by $CP - \Delta$, and q_1 and q_2 be the models returned by sharded-safe. We have that p is $k_x - NAF$ with respect to $\mathcal{C}$, sharded-safe, and Δ. Moreover, for the case of $\Delta_{\max}$, we have that for all $C \in \mathcal{C}$ and events $\mathcal{E}$,

$$p(\mathcal{E} \mid x) \leq \frac{\text{sharded-safe}_{\ C}(\mathcal{E} \mid x)}{1 - \mathrm{TV}\left(q_1(\cdot \mid x), q_2(\cdot \mid x)\right)}$$

Moreover, our goal is to not only prevent copyright infringement but, importantly, to also maintain high-quality generative models when $\mathcal{A}(\mathcal{D})$ itself is a high-quality model. The following lemma formalizes that the degradation of the model quality is bounded.

Lemma (Bounded Degradation). Let p be the model returned by $CP - \Delta$, and q_1 and q_2 be the models returned by sharded-safe. For $i \in \{1, 2\}$ and for $\Delta = \Delta_{\max}$,

$$\mathrm{TV}\left(p(\cdot \mid x), q_i(\cdot \mid x)\right) \leq \mathrm{TV}\left(q_1(\cdot \mid x), q_2(\cdot \mid x)\right),$$

and for $i \in \{1, 2\}$ and for $\Delta = \Delta_{KL}$,

$$\mathrm{KL}\left(p(\cdot \mid x), q_i(\cdot \mid x)\right) \leq -2 \cdot \log\left(1 - \mathrm{H}^2\left(q_1(\cdot \mid x), q_2(\cdot \mid x)\right)\right).$$

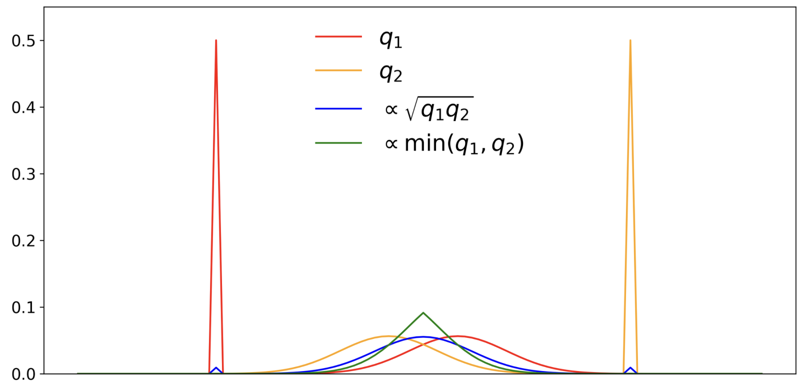


Figure 4: Best viewed in color. Applying our transformations to two distributions q_1, q_2 such that each has a 50% chance of outputting a different fixed element (the "spike"), and the remaining distributions have non-trivial overlap. We can interpret each model's "spike" as the probability that the model is outputting a "memorized" sample from its training set. Both the distribution proportional to $\min\{q_1, q_2\}$ and the one proportional to $\sqrt{q_1 q_2}$ (corresponding to $\mathrm{CP}-\Delta_{\max}$ and $\mathrm{CP}-\Delta_{\mathrm{KL}}$ respectively) significantly suppress the probability of the fixed element while approximately preserving the other probabilities.

References

[1] Pierre Fernandez et al., "The Stable Signature: Rooting Watermarks in Latent Diffusion Models," 2023.

[2] Ning Yu et al., "Responsible Disclosure of Generative Models Using Scalable Fingerprinting," 2022.

[3] John Kirchenbauer et al., "A Watermark for Large Language Models," 2023.

[4] Nikhil Vyas et al., "On Provable Copyright Protection for Generative Models," 2023.