Model and Data Watermarking

Micah Goldblum

1 Proof-of-Ownership

Companies own proprietary models they want to protect. A bad actor might either use a company's models against their terms of service (e.g. in the case of publicly available models without a commercial license) or distill their models if the models lie behind an API. So how can companies detect if another party's models are stolen copies of their own?

2 Model Watermarks

One way to protect proprietary models in such a scenario is to implant a signature, called a *watermark*, on the models which would be unlikely to appear in other non-copied models but which will be easy to detect in copies. Then, if a prospective model possesses this watermark, we can be confident that it is a copy. Below, we discuss some of the criteria for effective model watermarks, and we will subsequently discuss how watermarks work in practice.

2.1 Desiderata

- Robustness - model watermarks should resist removal attempts. For example, if an adversary fine-tunes the model, compresses its parameters, permutes the neurons, or adds noise to parameters or outputs to remove the watermark so that nobody can catch their theft, the watermark should still be detectable.
- Detectability - model watermarks should be easy and efficient to detect for the relevant party. Ideally, they should provide performance guarantees. On the other hand, one might want watermarks to be very difficult for adversarial parties to detect since this might make attacks easier.
- Performance preservation - embedding a watermark should not damage the performance of the model.
- Convenience and efficiency - embedding the watermark should not incur a substantial computational cost and should be easily accessible (i.e. not requiring specialized domain expertise) to anyone who deploys models.
- Generality - the watermarking technique should be applicable to all sorts of models, not just specific to LLMs or diffusion models, etc. Watermarking a new type of model should not require developing a new tailored watermarking technique.

- Capacity - The watermark should carry enough bits of information, which can be randomly generated, so that another model would unlikely contain that same information by chance.

3 Weight-Space Watermarking

These watermarks implant a signature on model parameters directly. On the one hand, an advantage of this approach is that such a watermark could be detected without expensive forward passes through the model, and in some cases, a weight-space watermark could perhaps be implanted in an already-trained model. On the other hand, weight-space watermarks may be fooled by attacks that change the parameters a lot without changing predictions much if at all, they might not be effective against distillation (which an attacker could perform even with API access and no access to the model's weights), and they may be impossible to detect on models hidden behind an API where weights are not accessible.

- `https://arxiv.org/abs/1701.04082` [9]
  - Uses a regularizer during training to inject statistical patterns into model parameters.
  - Focuses on convolutional networks.
  - Consider randomly generated sequence of T bits $b \in \{0,1\}^T$ which we want to encode in the model's parameters.
  - Also, consider a convolutional tensor W_{ijkl} with L different $S \times S \times D$ filters. Then, define $\overline{W}_{ijk} = \frac{1}{L}\sum_l W_{ijkl}$, and flatten this tensor out to $w \in \mathbb{R}^M$, where $M = S \times S \times D$. We will embed b into w. Since w is averaged over all convolutional filters, an adversary cannot permute the filters to remove the watermark.
  - Here's how we embed b into w: During training, we can add on a regularizer to our loss function,

$$E_R(w) = -\sum_{j=1}^{T}[b_j \log(y_j) + (1 - b_j)\log(1 - y_j)],$$

    where $y_j = \sigma(\sum_i X_{ji} w_i)$, and σ is the sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$. This objective is equivalent to binary classification loss, so that w gets classified into the correct bit value for each bit b_j in the watermark sequence b.
  - $\sigma(\sum_i X_{ji} w_i)$ can be viewed as a linear classifier.
  - X is randomly generated and can be viewed as a secret key for detecting the watermark. Without this key, it was hypothesized that detecting the key is impossible.
  - The distribution used to generate X may impact the watermark's detectability, and $N(0,1)$ works well.
- `https://scholar.harvard.edu/files/tianhaowang/files/icassp.pdf` [11]
  - This work shows that the weight-space watermark above is actually detectable as it injects distinct statistical patterns into the model weights.

  - We can determine the length of the watermark.
  - We can even overwrite the watermark to avoid detection of a stolen model. Overwriting is performed by generating new X and fine-tuning the model for several epochs.
  - This work seems to make assumptions about the generating distribution of X which could be relaxed to fool them.

- `https://library.imaging.org/ei/articles/32/4/art00003` [10]
  - The first watermarking algorithm we examined contains random linear classifiers where their input (watermarked neural network weights) is optimized instead of the linear classifier's weights, which were only randomly initialized.
  - This work uses a small neural network for watermark verification instead of the linear classifiers.
  - Optimize the small verification neural network jointly with the watermarked model.

- `https://arxiv.org/abs/1910.14268` [12]
  - We saw previously that watermarks can sometimes be detected because of statistical patterns they leave in the model's parameters.
  - How can we make watermarks that are undetectable.
  - This work develops a GAN-like approach, where a discriminator is trained simultaneously to detect a watermark. Then, the watermark is trained to maximize the discriminator loss during model training, to encouraging the model to embed an undetectable watermark by fooling the discriminator.

4 Function-Space Watermarking

These watermarks are instead embedded in the model's outputs rather than its weights explicitly. Specifically, a model watermarked in this fashion will exhibit a particular detectable output behavior, typically when presented with certain query inputs. Function-space watermarks can enable us to detect stolen models hidden behind APIs and may be robust to distillation or fine-tuning, but they may not provide statistical guarantees when models are not trained or fine-tuned specifically for the watermark. They also require potentially expensive forward passes.

- `https://arxiv.org/abs/1802.04633` [1]
  - This work frames the watermarking problem as implanting a backdoor.
  - We begin by generating n query images, for example with random pixels or strange patterns, and assign them each a random label (assuming the watermarked model will be a classifier with k labels).
  - Then, we append the n query images to the model's training dataset.
  - Assuming another model does not contain the associated watermark, then the number of labels it assigns correctly to the query images is binomially distributed $B(n, 1/k)$.

  - Using that information, we can perform a hypothesis test, namely by counting the number of labels a prospective model assigns correctly to the query images and computing the probability that a model would assign at least that many correct labels if it were not watermarked.
  - This method does not require that the images be random pixels, it only requires that the labels be random.
  - The power of the test improves rapidly with the number of samples and the number of classes.
  - This method generalizes outside of images and is agnostic to the architecture of the model.
  - We can think of the query images as a secret key for detecting the watermark whereby an adversary who does not possess the query images will likely not be able to detect that the model is watermarked, although this is not guaranteed.
  - Backdoor removal methods do exist, and it's not clear if they have been rigorously tested here.

- `https://arxiv.org/abs/1711.01894` [7]
  - This paper does the same thing but uses adversarial examples that are correctly labeled.
  - Advantages of this approach are robustness of the watermark to removal attempts and also that models which are not watermarked will pass the detector in few queries.

- `https://arxiv.org/abs/1906.00830` [8]
  - This paper instead assumes we have a model behind an API and we want to watermark any model that trains on our outputs (distillation attempts).
  - To that end, we can modify the output of a small fraction of API query samples and store the corresponding inputs to use for detecting that another model was distilled from those outputs.
  - We can make the watermark user-specific, so that we know exactly which client was responsible for acquiring the data on which to perform distillation.

- `https://proceedings.mlr.press/v162/bansal22a.html` [2]
  - Uses randomized smoothing to guarantee that any removal attack has to change the parameters a lot.
  - Randomized smoothing was designed as a certifiable defense against adversarial attacks whereby one can show that an input will have to move a certain amount to flip a classifier's prediction.
  - This work instead applies randomized smoothing to the parameters of the model instead of the inputs to guarantee that the parameters have to move a certain amount to change the predictions, namely on the query images we described above.
  - Limitation: moving the parameters a lot may not be difficult to do in a removal attack.

5 Data Watermarking

5.1 Text Watermarking

These watermarks don't prove ownership of a stolen model; they prove that text was generated by a particular party. Specifically, the watermark is embedded in the text generated by an LLM instead of the model's weights or predictions. So for example, if a client agrees that text generated by language model X will not be used to do Y, how can we prove that the text they used to do Y was in fact generated by model X? To accomplish this task, text watermarks imperceptibly modify the output distributions of language models so that they can still generate good text but prefer certain tokens or token sequences. Data watermarks typically operate on top of a pre-trained model, simply by changing the sampling procedure, and do not modify its weights at all. Therefore, a malicious party with a stolen copy of the model can typically generate samples at will without watermarks.

- `https://arxiv.org/abs/2301.10226` [4]
  - The following describes a sampling procedure for autoregressive language models that watermarks the generated text.
  - Every we generate a token, pseudo-randomly split all tokens in the vocabulary in half into a green list and a red list.
  - We can use the previous token as a random seed, or we can use several previous tokens.
  - Then, we add a small number to each logit corresponding to a green list token before computing softmax and sampling.
  - Watermarked text will have more green list tokens than red list tokens. In contrast, a non-watermarked model would only generate green list tokens roughly half of the time.
  - Thus, we can run a hypothesis test whereby we measure the probability of generating the observed number of green list tokens by random chance.
  - In this case, we can view the green/red list generator as the secret key for detecting the watermark.
  - Empirically, this watermark does not significantly degrade the quality of generated text.
- `https://arxiv.org/abs/2306.04634` [5]
  - This work studies the robustness of watermarking to removal attacks. In this case, unlike the model watermarking case where watermarks are removed from a model, the watermark is removed from generated text. Removal attacks in the case of text watermarks take the form of paraphrasing.
  - This work finds that paraphrasing attacks can be effective on short text snippets, but the watermark can still be detected with enough text.
  - This work also introduces an improved watermark detection scheme, WinMax, which searches for short spans of watermarked text in a row that often appear even under paraphrasing attacks since even a single word can contain multiple green list tokens in a row.

- `https://arxiv.org/abs/2306.09194` [3]
  - The previously described text watermark method has a glaring problem; it changes the distribution of sampled text. This could manifest in detectable watermarks or performance degradations. How can we implant undetectable watermarks in a language model?
  - The following is a simplified procedure. Imagine a language model that only outputs 0's and 1's. Let's call the probability that the model assigns to the output 1 for the i^{th} token, conditioned on previously sampled tokens, $p_i(1)$.
  - Our secret key for watermarking will be a sequence of uniformly distributed real numbers between 0 and 1 generated by a pseudorandom number generator, $\{u_i\}$.
  - Then, at watermarked generation, the model outputs $x_i = 1$ at the i^{th} token if $u_i \leq p_i(1)$.
  - Since $u_i \sim \mathcal{U}(0,1)$, this sampling procedure does not change the generative distribution of the language model, but we can detect watermarked text if we possess the secret key as follows.
  - Let $s(x_i, u_i) = \begin{cases} \log(\frac{1}{u_i}) & \text{if } x_i = 1 \\ \log(\frac{1}{1-u_i}) & \text{if } x_i = 0 \end{cases}$ and then let $c(x) = \sum_{i=1}^{L} s(x_i, u_i)$.
  - For non-watermarked text, the expected value of $c(x) - |x|$ is 0. On the other hand, the expected value of $c(x) - |x|$ for watermarked text is $\log(2) \cdot H(\overline{\textbf{Model}}(\textbf{prompt}))$, where H denotes Shannon entropy.
  - However, this difference in expectation is made less useful by the high variance, so the authors propose a better detector using empirical entropy.
  - Can also be generalized to bigger vocabularies and variable length generations.
- `https://arxiv.org/abs/2307.15593` [6]
  - Emphasizes imprinting a watermark which simultaneously avoids distorting the probability distribution of generated text and also is robust to removal attacks.
  - Proposes several variants which are each similar to the previously discussed text watermark.
  - Unlike the previous work, this work thoroughly tests out the robustness of their watermark and shows that their best variant is at least as detectable and robust as the first watermark along with WinMax, even though this one does not distort the probability distribution of generated text.

5.2 Diffusion Model Watermarking

`https://arxiv.org/abs/2305.20030` [13]

- Embed a Fourier-space ring pattern into the initial noise used for generation. Invert the FFT, and use that noise for generation with the DDIM generation algorithm.
- When presented with a generated image, invert the DDIM generation using an empty prompt, and measure the distance of the initial noise (or a Fourier-space patch of it) to that of the watermark pattern.

- If the distance from the initial noise to the watermark pattern is below a threshold, then we say the image was generated by the watermarked model.
- This watermark seems to avoid any significant deleterious impact on the quality of generations.
- This watermark is more robust to common image transformations (e.g. JPEG compression, rotation, etc.) than existing image watermarks.

6 On the Relationship Between Model and Data Watermarking

Model and text watermarking can be used to enforce the same policies but in different settings. If a language model does not have a commercial license, then how can the owner prove that a party used it illegally? In the case that the model was made publicly without a commercial license, one way a company could detect that another party stole their model is by accessing the allegedly stolen model, either its weights or predictions, and detecting their proprietary model watermark. In the case that they did not make the model available publicly but instead only allowed API access, then they can inspect text that the accused party has generated in which case they may be able to detect if it is watermarked text output by their own proprietary model.

7 Directions for improvement and future research

- The model watermarks we previously discussed require training, unlike the data watermarks. So how can we apply a model watermark to a pretrained model with minimal overhead?
  - One way to embed a watermark is by fine-tuning the model for a very small number of optimization steps on a batch of random data with random labels. This approach has the benefit of being very cheap and maintaining a statistical guarantee but has the disadvantage of possible unexpected degradations to the pretrained model. Questions: *How little fine-tuning can we get away with while maintaining a strong watermark? Are there noticeable performance degradations if we fine-tune on only the random data and not the original training data?*
  - An alternative way to distinguish a pretrained model from others without modifying its parameters at all is to search for inputs which distinguish the particular pretrained model from a database of others. For example, take your proprietary LLM and search for inputs which give your model very different output than LLaMa, LLaMa 2, and Falcon. These inputs can then be used to distinguish your model from others in the future, but they carry no formal statistical guarantee. Questions: *Is it possible to get a guarantee? How can we do the optimization, especially over text?*
- Can you detect stolen feature extractors? Foundation models are often fine-tuned for downstream tasks. In some cases, the labels for the downstream task differ from the upstream labels, and practitioners drop a new prediction head on top of the feature

extractor. Question: *Can we build watermarks that we can detect even after the model has been fine-tuned and even with a new prediction head?*

- Works on function-space model watermarking typically focus on classification, but users may also deploy models for regression, language, diffusion, etc. *It would be valuable to generalize function-space model watermarking so that it can be applied generically to any model users deploy.*
- A downside of function-space watermarks is that they require function evaluations, namely performing forward passes through the model so we can inspect its outputs. Such forward passes can be expensive. *Can we solve this problem, for example by only doing forward passes through part of the model?*

References

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1615–1631, 2018.

[2] Arpit Bansal, Ping-yeh Chiang, Michael J Curry, Rajiv Jain, Curtis Wigington, Varun Manjunatha, John P Dickerson, and Tom Goldstein. Certified neural network watermarks with randomized smoothing. In *International Conference on Machine Learning*, pages 1450–1465. PMLR, 2022.

[3] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2023.

[4] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023.

[5] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*, 2023.

[6] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

[7] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32:9233–9244, 2020.

[8] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. Dawn: Dynamic adversarial watermarking of neural networks. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4417–4425, 2021.

[9] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pages 269–277, 2017.

[10] Jiangfeng Wang, Hanzhou Wu, Xinpeng Zhang, and Yuwei Yao. Watermarking in deep neural networks via error back-propagation. *Electronic Imaging*, 2020(4):22–1, 2020.

[11] Tianhao Wang and Florian Kerschbaum. Attacks on digital watermarks for deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2622–2626. IEEE, 2019.

[12] Tianhao Wang and Florian Kerschbaum. Riga: Covert and robust white-box watermarking of deep neural networks. In *Proceedings of the Web Conference 2021*, pages 993–1004, 2021.

[13] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv preprint arXiv:2305.20030*, 2023.