Deepfake Detection: Research Project Overview

Ahmed Hussain
Computer Engineering, Physics (2023)
ahh6qxu@virginia.edu

Sam Buxbaum
Computer Science, Physics (2023)
smb8xc@virginia.edu

September 3, 2024

1 Introduction

Fake media poses an ever-increasing threat to society. Deepfake technology represents the state of the art in generating fake media, such as images and videos. Deepfakes are artificial images, video, and audio which can be nearly impossible for humans to differentiate from real media. This presents threats to our technological lifestyles and media-dominated culture through the possibility of malicious intervention and cybersecurity risks.

Current deepfake detection techniques rely almost exclusively on identifying subtle flaws in fake media, e.g. a tiny irregularity in the blinking pattern of a face. Detecting these irregularities is done primarily using machine learning techniques, where private corporations or research institutions construct models with huge amounts of data. Using pure ML approaches is inadvisable for a few reasons: a) these models are trained on data which covers attacks discovered before the date of model creation; b) these models need to be recreated from scratch each time; c) they are generally not openly-available for the public to use due to sunk costs; d) when new detection models are published, deepfake generation techniques gain a new adversary to train against, and the task of detecting them gets more difficult.

In this overview, we present our ideas for a detection framework and a summary of future research in "perceptographic" hashing. Rather than approaching the cat-and-mouse game of training models, finding new attacks, and training better models, we take the assumption that deepfakes will someday be truly indistinguishable from reality and that a different, more fundamental approach is necessary.

1.1 Our Approach

We propose to detect fake media by creating and maintaining a list of "valid" media. As original works of media are created and uploaded to a platform (regardless of where), we will also add a hashed copy of this media to a global list. Obviously, with this approach it is impossible to capture all valid media that exists or has existed - in this detection method, we restrict our scope to news media such as television news and online publications, since this is comparatively simple to manage and is an area where fake media is particularly damaging. Ideally, we will expand the scope of the detection mechanism as the tools we develop for detection improve. Further development is needed here to turn this into a viable solution for *all* media, as simply backing up all media is a brute force solution. Here, news media serves as a motivating example that is both important and approachable.

Many of the challenges in the design of the system relate to the data structure used to store the list of media artifacts. The ideal data structure should store artifacts in a space-efficient and distributed way to minimize the size of the list and avoid having a single point of failure. A natural, but potentially overkill, solution is to use a blockchain which stores hashes of the artifacts across a network of computers. This blockchain would include a distributed consensus protocol to obtain agreement among the network. As we'll see, in the comparatively simple case of news media, the use of a blockchain is somewhat questionable, but

it provides the advantage that it scales well as the problem becomes more complex and the volume of news media rapidly increases. We will refer to the data structure as a blockchain from this point forward, but it is important to note that it is much more similar to a distributed set than a sequence, as is typically stored in a blockchain. The problem of how to best hash the media before storing it on the blockchain is particularly interesting and will be discussed later.

1.2 A Brief Example

Imagine several prominent news publishers (e.g. CNN, New York Times, etc.) decided to work together to prevent the proliferation of altered versions of their original content. The publishers would maintain a collective list of media that they have published; each time one of them publishes a new artifact, such as an article or a video, they will automatically add the hash of the artifact ("perceptographic" hash) to the global list (blockchain). If a random person on the internet wants to check the validity of a news artifact, they could do so through some interface (e.g. web extension, website, etc.) by checking if the hash of the artifact is in the list of accepted hashes.

Furthermore, this process could be automated to give some type of confirmation of validity whenever a secondary news source references the original and makes use of one of the artifacts, such as by showing a video originally published by one of the news publishers in the network. For example, if NYT uses a video originally created by CNN, it could have a "verified" checkmark in the top right corner, or similar, to prove the validity of the embedded media. If the shared list of hashes is publicly available, then this system has the added benefit of allowing individuals to verify validity locally, which provides transparency and eliminates the need for trust in an institution.

1.3 Completed Work

The bulk of the work so far has been related to conceptually developing the infrastructure necessary for a distributed network. To date, we have developed a prototype of a blockchain-based network where individual nodes can send dummy hashes amongst the network to construct the shared list. We have developed a full peer-to-peer communication backend, allowing any node in the network to communicate with any other node or to broadcast a message to the entire network. The Github for the project is available upon request (it's private at the moment!).

So far, we have been able to test the full system as a collection of independent processes running on the same computer, but we are currently working on setting up a small test network of multiple computers (likely smaller edge devices, e.g. Raspberry Pi, ESP32, Arduino, etc.).

2 Future Work: Perceptographic Hashing

In order for the system to have any realistic success, hashing media artifacts is essential. However, this comes with many challenges. Here, we'll walk through a progression of ideas which will expose many of the difficulties and potential ways of handling them.

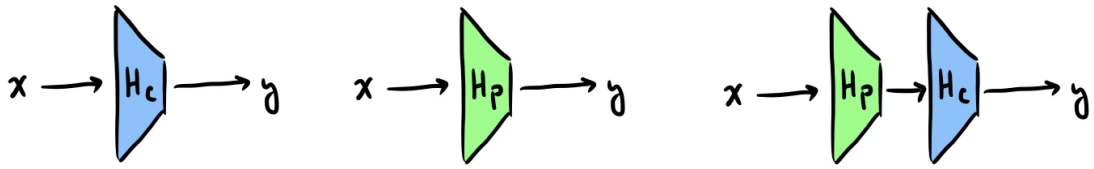

Figure 1: The three scenarios explored in this section, from left to right respectively.

1st approach: cryptographically hash all media artifacts using a function H_C. This works great for artifacts such as text documents, which will yield a consistent result if hashed many times, but it fails for images and videos in the real world. It is possible and entirely realistic that an image will be compressed and distributed across the internet using lossy compression across numerous platforms, so while any two copies of the image look identical to a human, their cryptographic hashes would be different and completely distant.

2nd approach: using a *perceptual hash* H_P, which is a type of non-cryptographic hash function which maps similar inputs to similar outputs. A perceptual hash attempts to reduce the size of the data represented while preserving similarities between close inputs. This solves the problem of lossy compression changing the pixel values of an image, since files with different compression outputs can be grouped together, but it creates a new problem: perceptual hashes are not designed to have any cryptographic properties, so they have no second preimage resistance. That is, given a hash, it might be easy for an adversary to generate a new image which achieves the same hash with an unrelated input. The second image can even be adversarially generated with knowledge of the hash function to tweak individual pixels and trick the system into identifying two distinct images as identical, similar to how deepfakes are generated.

3rd approach: combine them, applying a perceptual hash to capture the "essence" of an image and then cryptographically hashing the result, making a two-level hash. However, the same flaw applies to the perceptual hashing stage as before: there is nothing stopping an adversary from creating two distinct images with identical perceptual hashes. This combination is simply a more complicated method of failing in the same way, as it still has no second preimage resistance.

Lastly, we could enforce lossless compression so that the original objection to cryptographic hashing becomes invalid, but this is wildly unrealistic to accomplish, since it would require the entire internet to change to accommodate this individual system.

2.1 Problem Statement

We frame the problem statement for developing a "perceptographic" hash as the following. A portmanteau of "cryptographic" hashing and "perceptual" hashing, perceptographic hashing is a hashing technique such that images which are "slightly different" due to variations like compression intensity, minor uniform blurs, minor color tints, etc. (which can arise from uploading or downloading from the web) are grouped together to the same hash, but images images which are "very different", i.e. one is a certifiable deepfake, will not be binned together, and their hashes should seem random and unrelated. Futhermore, it should be difficult to produce an image which can create a collision.

More formally, a perceptographic hash function should satisfy the properties of a cryptographic hash function – preimage resistance, second preimage resistance, and collision resistance – with the exception that images which are "slightly different" should hash to the same value. Specifically, the perceptographic hash should meet the following conditions:

1. Preimage resistance: For a given output hash y and perceptographic hash function $H()$, it should be computationally infeasible to produce any input media x such that $H(x) = y$.

2. Second preimage resistance: For a given input media x_1 and perceptographic hash function $H()$, it should be computationally infeasible to produce a second input media $x_2 \neq x_1$ such that $H(x_1) = H(x_2)$.

3. Collision resistance: It should be relatively difficult to find a pair of messages $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$, unless the inputs are "slightly different". There should be collisions between "slightly different" media, but "very different" media should not reach a collision.

Considering the approaches mentioned above, we can infer that all existing naive approaches fail to meet the standards of our system, leaving us with the options of developing our own hash function for the system or proving that such an approach is not possible.

We believe this is an interesting open question with potential impact beyond just this one system. Our primary research question is:

Is it possible to construct a *perceptographic* hash function with cryptographic properties such that "slightly different" inputs all lead to identical hash function outputs?

2.2 Similar Work in the Literature

This problem turns out to be quite similar to the problem of *property-preserving hashing.*
Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data [1]
The idea that similar images should yield similar or identical values is captured by the notion of *fuzzy extractors* ([1]).
Adversarially Robust Property-Preserving Hash Functions [2]
However, as pointed out in [2], fuzzy extractors are not sufficient to prevent against adversarially generated inputs.
Nearly Optimal Property Preserving Hashing [3]
[2] and [3] study adversarially secure property-preserving hashing. Our open problem is an extension of this work to the preservation of image and video "similarity."

References

[1] Dodis, Ostrovsky, Reyzin, Smith. *Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data.* `https://arxiv.org/pdf/cs/0602007.pdf`.

[2] Boyle, Lavigne, Vaikuntanathan. *Adversarially Robust Property-Preserving Hash Functions.* `https://eprint.iacr.org/2018/1158.pdf`.

[3] Holmgren, Liu, Tyner, Wichs. *Nearly Optimal Property Preserving Hashing.* `https://eprint.iacr.org/2022/842.pdf`.