Hany Farid
Professor
Electrical Engineering & Computer Sciences and School of Information
University of California, Berkeley
https://farid.berkeley.edu | hfarid@berkeley.edu | 510.664.4438

Josephine Ballon
Head of Legal
HateAid gGmbH
c/o Stiftung Haus der Demokratie und Menschenrechte Greifswalder Straße 4
10405 Berlin

October 28, 2021

I write in response to your request for an opinion as to the state of the art in image-based content moderation. Because hashing is the dominant computational technique for large-scale, online content moderation, I will begin with an overview of hashing, followed by responding to your specific questions regarding the efficacy of hashing and related technologies in detecting image-based memes and their variants.

Background

Hashing is arguably the most widely used technology for limiting the spread of previously identified multimedia content. This technology is used to contend with everything from terror-related content to child sexual abuse material (CSAM) and copyright infringement. After an offending audio, image, or video is identified (either manually or automatically), a distinct digital signature is extracted from the content (the digital signature is often referred to as a fingerprint, message digest, hash value, hash code, hash sum, or, simply, hash). The same hash is extracted from each future upload and compared against a database of offending hashes. Any matched content can, for example, be automatically blocked from upload or subjected to any policy a service provider initiates.

The terms hash and hashing originated from the culinary term describing the process of chopping a mixture of ingredients to yield, for example, a hash of potatoes, peppers, and tomatoes. In the digital realm, hashing refers to chopping a data file into small pieces and combining them to yield a concise numeric value that can be used to identify the original data file. Although we are discussing hashing in the context of the modern internet, the importance of hashing dates back to the early years of the electronic computer, the process having first been described in 1953, with the term hashing first appearing in 1967.

In the broadest use of the term, there are many applications of hashing, including efficient data storage and retrieval, data integrity, and data matching. I will focus on this latter use, and in particular the application of hashing to digital media files (audio, image, and video)

In order to be most effective at internet scale with billions of daily uploads, an effective hash should have several properties:.

- Distinct. The extracted hashes for two distinct pieces of content should themselves be distinct; this ensures that innocuous content is not incorrectly matched.

- Resilient. The extracted hash for a piece of content should be resilient to simple content modifications that do not fundamentally alter the underlying content; this ensures that perceptually similar but not identical pieces of content will be matched.

- Deterministic. The extracted hash for a piece of content is always the same; this ensures that a piece of content can always be identified based on its hash.

- Efficient. The computational cost of extracting a hash and comparing hashes should be low; this ensures the technology can be deployed on even the largest social media sites.

- Non-reversible. It should be practically impossible to reconstruct the original content from only the hash; this allows for hashes of illegal content (e.g., CSAM) to be shared across different organizations.

Different flavors of hashing comply with one or more of the above five criteria, but not all five. Hard-hashing techniques, for example, are highly distinct but not resilient to even the smallest change in the underlying media file. In contrast, perceptual hashing can be resilient but is less distinct (perceptual hashing is also referred to as robust hashing, fuzzy hashing, or content-based image retrieval).

Hard-Hashing

The most widely used hard-hashing algorithms are arguably MD5 and SHA-1, along with their variants (MD2, MD4, MD6, and SHA-0, SHA-2, SHA-3). The MD5 (message digest) algorithm takes as input an arbitrary data file and generates a 128-bit hash, typically represented as a sequence of 32 hexadecimal characters (hexadecimal corresponds to a base 16 alphanumeric system consisting of the values `0-9` and `a-f` [or `A-F`]). A simple text file containing the text "An Overview of Perceptual Hashing", for example, yields the MD5 hash `7b2cc72c5a57c2d6878a9d897b21e9cd`. The MD5 algorithm first partitions the data file into 512-bit blocks, from which a per-block hash is computed by manipulating the data in each block alongside the hash from the previous block. The final hash is generated in the final data block.

MD5 was initially designed as a cryptographic hash with additional security features beyond the basic hash features described above, including that a message cannot be generated to yield a specific hash and that a small change in the data file leads to a large and uncorrelated change in the hash (the so-called avalanche effect). Although these cryptographic properties were found not to hold, MD5 hashing continues to be used for basic data integrity checks and data matching.

Shown below is the MD5 hash of five text files, each containing a single line of text.

```
MD5(“Perceptual")                          = 4fcd92bbcf33b0b4a1eac76a673e1f33
MD5("Perceptual Hashing”)                  = c6eedea9b3db1e5ff9e80153dc629dcc
MD5("An Overview of Perceptual Hashing") = 7b2cc72c5a57c2d6878a9d897b21e9cd
MD5("an Overview of Perceptual Hashing") = 6df033f24a309e7eca6c70834d34bd2a
MD5("xn Overview of Perceptual Hashing") = 9a87d6c846de4f2bc35a40e42bd1d4a8
```

Notice first that regardless of the size of the input data file, the resulting hash consists of 32 hexadecimal characters. Notice also that even small differences in the input file (as in the last three entries) lead to large and seemingly uncorrelated changes in the hash. This latter property has both advantages and disadvantages when it comes to identifying a piece of content based on a hash.

Even the smallest differences in the input file lead to large and seemingly uncorrelated changes in a hard-hash. This property has both advantages and disadvantages when it comes to identifying a piece of content based on a hash.

MD5 and other similar hard-hashing algorithms have been used by various platforms to identify a specific digital media file. In an offline stage, a hash is extracted from each offending image (or video, audio, or any other data file) and stored for future comparison; then, in an online stage a hash is extracted from each piece of uploaded content and compared against all database hashes, after which any matched content is blocked, reviewed, reported, or subjected to whatever policy a service provider initiates.

Because even the smallest change in the data causes a large change in the resulting hash, the only meaningful comparison of two hashes is an exact match for equality between their hexadecimal characters. The advantage of this property is that, because the extracted hashes for two distinct pieces of content are themselves distinct, it is exceedingly unlikely that one piece of uploaded content will be mistakenly identified as another piece of content. The disadvantage of this property is that only exact content matches can be discovered. In the context of multi-media content (audio, image, and video), this latter property is undesirable because content is often either intentionally or unintentionally modified as it makes its way around the internet.

While any service provider should use hard hashing to limit redistribution of banned or illegal content, this technology itself is easy to circumvent. Perceptual hashing provides an additional and important layer of protection because it is, by design, able to detect many of the variants that hard hashing misses.

Perceptual Hashing

A digital image is made up of an array of pixels (picture elements). Each pixel is itself composed of three values corresponding to the primary red, green, and blue colors (RGB). In a typical digital image, each 8-bit pixel takes on a value from 0 to 255. A bright, pure red pixel, for example, is represented by the triple values (255,0,0), a dim, pure blue pixel by (0,0,64), a black pixel by (0,0,0), a mid-level gray pixel by (128,128,128), and a pure white pixel by (255,255,255).

A modest 12-megapixel digital sensor produces a digital image of size 3000 x 4000 pixels. With a total of 256 possible values per pixel for each of three color channels, this sensor can produce a mind-boggling number of different images, exceeding the some 10^{24} estimated stars in the universe.

The goal of perceptual hashing is to mimic the human visual system's assessment of comparing two images based on the underlying scene content, as compared to a purely numeric comparison based on the pixel values. This is accomplished by extracting from the massive pixel-space representation a concise, distinct, perceptually meaningful signature that is resilient to modifications of the image, including compression, color shifts, cropping, rotation, the addition of a logo or overlain text, or any other modification that does not fundamentally change the underlying content but does alter the underlying pixel values. The extraction and comparison of a perceptual hash must also be efficient so that it can operate on billions of daily uploads.

Most perceptual hashing algorithms consist of three basic parts: (1) preprocessing (e.g., image resizing, image gradient); (2) hash extraction; and (3) hash comparison (e.g., Euclidean distance). The many choices for each of these stages result in varying compromises between the five desired properties of a hash enumerated above.

The most typical compromises are between distinctiveness and resilience, and the specific manipulations to which the hash is resilient. A hard hash, for example, is highly distinct but exhibits no resilience to even a small content change. Similarly, some hashes are resilient to color and brightness changes but not necessarily to geometric changes like rotation and cropping. In practice, the choice of a hash is based on a number of factors, including:

- Scale. When operating at the scale of a major social media platform, for example, with billions of daily uploads, any hash must be highly efficient and distinct. At this

scale, even a 1/100 or even 1/10,000 false positive rate (incorrectly matching two images) is untenable.

- Tolerance. When trying to limit the upload of, for example, legal adult pornography, resilience may be less important than, for example, trying to limit the distribution of CSAM.

- Security. When trying to limit the upload of copyright-infringing material, for example, non-reversibility may be less critical than in other more sensitive domains.

Implementations

The academic literature is home to a rich source of different perceptual hashing techniques. At the same time, a number of these techniques have been deployed on a range of online services, for a range of different applications.

In 2007, YouTube deployed Content ID (originally called Video Identification) after facing threats of litigation and massive monetary fines for not doing enough to prevent—and for profiting from—the sharing of copyrighted materials. This perceptual audio/video hashing extracts a distinct signature from owner-supplied content. When flagged at upload, content owners can choose to block, track views, or monetize the content.

By 2008, facing an explosion in the spread of child sexual abuse material, the leading technology companies had yet to develop similar technology to contend with this horrific content. Led by Microsoft, a perceptual image hash—PhotoDNA—was developed, tested, and deployed in 2009 on Microsoft's Bing search engine and on what was formerly called SkyDrive cloud service. In 2010, Facebook deployed PhotoDNA on their services. In 2011, Twitter followed suit, while Google waited until 2016 to deploy. In addition to these titans of tech, PhotoDNA is now in worldwide deployment across a range of platforms. According to the National Center for Missing and Exploited Children (NCMEC), a major contributor to the more than 100x increase in reports to their CyberTipline over the past decade has been the adoption by service providers of perceptual hashing tools like PhotoDNA.

PhotoDNA was designed to perceptually hash images because, at the time, digital video had not yet become ubiquitous. In the intervening decade, videos have come to dominate many online platforms. In 2016, largely in response to the growing threat of international and domestic extremism, the Counter Extremism Project (CEP) developed eGlyph, an audio, image, and video perceptual hash.

In response to a rise in online extremism and terrorism, the Global Internet Forum to Counter Terrorism (GIFCT) was founded in 2016 by Facebook, Microsoft, Twitter, and YouTube to "foster technical collaboration among member companies, advance

relevant research, and share knowledge with smaller platforms." As part of this mission, the GIFCT has created an image and video hashing database of identified terrorism materials that it shares with industry partners.

In 2019, Facebook announced an open-source image and video hashing, PDQ (with the non-obvious root "a Perceptual algorithm utilizing a Discrete Cosine Transform and outputting [amongst others] a Quality metric") and TMK+PDQF (Temporal Match Kernel + PDQ with Floating-point operations).

In August of 2021, Apple announced the development of NeuralHash for detecting known CSAM in images stored in Apple's iCloud. This technology was scheduled to roll out in iOS 15 and iPadOS 15, but was delayed due to pressure from privacy groups. Unlike PhotoDNA that maps perceptually similar images to similar, but not necessarily identical, hashes, NeuralHash converts each image to a unique numeric value. This unique image-to-hash mapping, along with private set intersection—a secure technique for database comparisons—allows for a more privacy-preserving hash database comparison. As compared to PhotoDNA, which performs the image hashing and comparison upon upload to a server, Apple's NeuralHash performs the image hashing and comparison directly on the device. This client-side comparison allows for a more privacy-respecting hashing, as it does not reveal the image contents to the server unless a match is found upon upload to iCloud. Apple reports a 1 in 1 trillion likelihood of incorrectly matching a user's account; manual review further mitigates the chance of a false report.

Memes

You asked me to consider the efficacy of automatic technologies in detecting image-based memes with the following variations:

1. The meme is 100% identical.
2. Picture and text are identical, but it looks different - because of resolution, size/ cutting, colour filters, frame.
3. Only the text is identical.
4. The text is identical, but with an addition like: "see who is the real enemy of Germany:"
5. The text is identical, but it is annotated by graphical means.
6. The faked quote is identical, but the annotation (name and party of the politician) is written differently.
7. Picture and text are identical, but there is a caption (e.g. one that is critical about the quote or even highlighting that it is fake).

Generally speaking, perceptual hashing is designed to detect an image and its variants. The efficacy of detecting variants depends on how much of the image is itself modified and the robustness of the perceptual hashing algorithm to variants. The

robustness to variants of any perceptual hash can be enhanced by configuring the hashing to match increasingly larger variations. While this approach may lead to more false positives (matching distinct images), this can be mitigated by increasing human content moderation for reviewing potential matches. That is, perceptual hashing need not work in isolation of human moderation.

In addition, even if a variant is not automatically detected based on a perceptual hash, once a variant has been manually identified, a hash can be extracted from the variant, added to the hash database, and the variant image can be filtered from future uploads. Over time, this approach would eventually catch most, if not all, variants.

With respect to the efficacy of hashing in directly detecting the above variants. Both hard- and perceptual-hashing is highly effective at detecting #1. Perceptual hashing is highly effective at detecting #2. Hashing would not be effective in detecting #3, but standard optical character recognition (OCR) could be applied to detect memes based on any text overlaid atop an image. A variant of perceptual hashing could be used to detect #4 – #7. In particular, instead of hashing the entire image, a region-based hashing can be used in which, for example, just the left or right (or upper or lower) part of the image is hashed for comparison. Alternatively, OCR can be used to identify portions of the image containing text and these portions could be automatically excluded from the hash computation in order to focus only on the underlying image contents. Or, new perceptual hashing algorithms could be specifically designed to analyze memes and be designed to be invariant to overlaid text.

Perceptual hashing is a well established and understood technique. It is highly efficient and accurate and has been widely deployed across services of all sizes; there are no major obstacles to implementing these technologies on even the largest online services.

I will also point out that it defies logic to argue that detection of hateful memes is difficult because of fear of blocking correcting memes. If platforms prevented the hateful memes in the first place, there would be no need for corrective memes.

You also asked me if it is possible to automatically asses a caption embedded in an image and if it is possible to distinguish between identical and similar quotes in an image-based meme. OCR can be used to automatically and accurately extract text from an image which can then be easily compared against a database of known meme text or analyzed with standard natural language processing (NLP) techniques that can distinguish between hateful memes and those attempting to debunk these memes. OCR technology is a highly mature and advanced; there is little technological limitation to deploying this technology.

You also asked me which technologies are best suited for assessing the similarity of images. As described above, perceptual hashing is the most effective, well-understood, and widely deployed technology for comparing the similarity of images. In addition, the past few years have seen tremendous progress in machine learning using deep neural networks. A deep neural network could easily be trained to detect a single meme and its variants, and in fact, most – if not all – online services are using deep neural networks in various aspects of their data-analysis pipelines.

Lastly, you asked me if it is know when and how social networks such as Facebook use some of the content moderation tools enumerated above. It is known that social networks use these tools to counter copyright infringement, child sexual abuse material, terrorism/extremism, hate speech/bullying, certain conspiracies (e.g., QAnon), and at least in the case of Facebook to block the upload of legal adult pornography or sexually explicit material (a violation of their terms of service).

While many social media platforms claim technological limitations to detecting everything from hate and terror speech to CSAM, these same services have been highly effective at limiting the distribution of copyright infringement material (out of fear of litigation) and adult pornography (out of fear of impacting their advertising business). It defies credibility that they are so effective in one arena, while ineffective in another. The underlying issue is not one of technological limitations, but of commitment and investment in the appropriate technology.

Sincerely yours,

Hany Farid
Professor
UC Berkeley

About me: I am a Professor at the University of California, Berkeley with a joint appointment in Electrical Engineering & Computer Sciences and the School of Information. My research focuses on digital forensics, image analysis, and human perception. I received my undergraduate degree in Computer Science and Applied Mathematics from the University of Rochester in 1989, my M.S. in Computer Science from SUNY Albany, and my Ph.D. in Computer Science from the University of Pennsylvania in 1997. Following a two-year post-doctoral fellowship in Brain and Cognitive Sciences at MIT, I joined the faculty at Dartmouth College in 1999 where I remained until 2019. I am the recipient of an Alfred P. Sloan Fellowship, a John Simon Guggenheim Fellowship, and am a Fellow of the National Academy of Inventors. I was part of the team that developed and deployed PhotoDNA and eGylph.