Detection of NFT Duplications with Image Hash Functions

***Abstract*—Non-Fungible Tokens (NFTs) are digital assets representing ownership or proof of authenticity of a unique item. NFTs are blockchain-based and rely on smart contracts. The increase in duplicate NFTs in recent years brings the need for discovery tools for forged NFTs, some of which include using image hash functions. Though the problem of image duplication is widely discussed, detecting NFT duplications requires using fast detection methods as a new NFT image needs to be compared with the entire NFT history on the blockchain. In this paper, we analyze the performance of several image-hash functions, examine the cases where each function performs well, and evaluate multiple image-hash-functions-based NFT duplication detectors. Our models achieve more than 97.5% success in detecting NFT duplications and show that using several hash functions rather than one increases the ability to detect duplications.**

I. Introduction

A Non-Fungible Token (NFT) [1] is a type of digital asset that represents ownership or proof of authenticity of a unique item or piece of content such as art [2], music [3], videos [4] and even event tickets [5]. NFTs are blockchain-based, using smart contract technologies supplied by blockchain systems such as Ethereum [6]. NFTs were first proposed in Ethereum Improvement Proposals (EIP)-721 [1], and were later implemented in EIP-1155 [7]. NFTs differ from classical cryptocurrencies [8] such as Bitcoin [9] or Ether [6] as while in classical cryptocurrencies all tokens are fungible, or equal, and hence can be exchanged like-for-like, each NFT is unique, making it non-fungible. While NFTs have gained vast popularity in recent years, NFTs face many challenges and NFT abuse can be accessible and simple [5], [10]. Over 80% of the assets of OpenSea[1], the world's first and largest Web3 marketplace for NFTs and crypto collectibles, that were flagged as plagiarized works, fake collections, and spam were created with their simplified "lazy minting" [11].

As the number of NFTs created increases (NFT exchanges recorded a trading volume of 343 billion USD in the second quarter of 2021 [12]), the issue of NFT duplication arises. This both reduces the value and uniqueness of the original asset and leads to copyright infringement and legal disputes. As the digital content looks nearly identical and is at a lower cost, unsuspecting buyers might tend to favor duplications of the original items [13]. Das et al. [14] used a simple perceptual hashing algorithm, downloaded around ten million NFT images from OpenSea and discovered almost 60 thousand hash collisions, manually verifying that most of the collisions they found were indeed visually identical images.

[1]https://support.opensea.io/hc/en-us/articles/8381389579539-What-is-OpenSea-s-copymint-policy-

Though the field of image duplication detection is widely studied, most solutions cannot be applied so easily for NFT duplication detection as every new NFT needs to be compared to the entire blockchain to ensure it is not a duplication, hence a fast and efficient model is needed. Therefore, an efficient method is necessary to identify and detect NFT duplications.

In this paper, we review the use of image-hash-based functions for NFT image near-duplication detection. As there are several criteria defined by OpenSea for image duplications, we analyze how the hash functions perform on each duplication type. We show the different hash functions perform differently on each duplication type, and there is no single hash function that performs better than others in all cases. Then, we implement several image-hash-function-based NFT duplication classifiers, showing they can detect duplications successfully in 97.5% of the cases. Finally, we conclude our results and explore possible future work directions.

II. Background

A. Non-Fungible Tokens (NFTs)

A non-fungible token (NFT) is a unique digital identifier that is recorded on a blockchain and is used to certify ownership and authenticity. The NFT ownership can be transferred by its owner, allowing NFTs to be sold and traded. NFTs typically contain references to digital files such as images, audio, and videos. In practice, the ownership of an NFT as defined by the blockchain has no inherent legal restrictions as an NFT does not restrict the sharing or copying of its associated digital file. The notion of NFT was introduced in 2018 in a paper titled ERC-721: Non-Fungible Token Standard by Entriken et al. [1] as a standard for smart contracts by which tokens are distinct and have unique attributes and ownership details. While NFT prices reached record highs in 2021-2 with artwork being sold for over 69 million USD, prices and trading volumes became significantly lower in 2023.

B. Near Duplication Detection

The field of image Near-Duplication Detection has been widely researched in the past 20 years [15], [16], [17], [18]. In recent years, with the increase of interest in machine learning, most recent researches focus on learning-based methods to detect image duplications [19], [20], [21], [22], [23]. These techniques include studying unique image features for classifying duplications correctly. Yet, applying these solutions to image NFT duplications is not trivial as this task requires the use of a very fast model.

Near-duplicated content protection (visual fingerprinting) is a technology designed to detect slightly altered iterations of multimedia content. Through fingerprinting, the perceptual attributes of digital content are condensed into a semantically consistent digest. Unlike cryptographic hashing, which merely indicates whether two data pieces are precisely identical, fingerprinting preserves semantic details about the input. This enables not only the determination of strict equality between two fingerprints but also the assessment of their degree of similarity [24]. While many near-duplication detection solutions are machine-learning-based and are heavy to process, other solutions that involve image hashing are much more lightweight. Recent work [25] has suggested using image hashing on NFT marketplaces to detect NFT duplications. Although this work focuses mainly on the marketplaces rather than duplication detection, it shows the potential of image hashing for NFT near duplication detection.

C. NFT Duplication

Although the border between copying an image and creating a new image based on an existing image is not exactly defined, OpenSea defines several characteristics for *copyminting*, that is an NFT image that is considered a duplication:

- A pixel-for-pixel replica (exact match of an original collection).
- A flipped, rotated, or facing the opposite direction replica.
- A resized, zoomed in or out, cropped, or repositioned replica.
- A replica with added or removed borders and edges.
- A replica with unintegrated texts, logos, or emojis, either typed or drawn.
- A replica that only swaps the background color.
- A pixelated version of the original artwork.
- A replica with colors swapped, modified, or saturated (for example, colors have been brightened or darkened, including black and white versions of the original).

D. Image Hashing

Image hashing refers to hash-based algorithms used to assign a unique hash value to an image. In this process, an image is converted into a fixed-size hash value, where, unlike standard cryptographic hashes, similar images should produce a similar hash. Image hashing is useful for tasks such as image compression, image similarity comparison, image retrieval, and near-duplicate detection. Some of the most common image hashing functions are average hashing (*aHash*), perceptual hashing (*pHash*), difference hashing (*dHash*), Haar Wavelet Hashing (*wHash*), and HSV Color Hashing (*HSVHash*). The main steps of an image hash algorithm are as follows: First, some preprocessing (such as grayscaling) is performed on the image. Feature extraction (such as color distributions and texture patterns) may optionally be performed too. Later, some normalization can be performed on the image and then the hash function (converting the input into some hash string) is applied. After a hash is created, it can be compared to another hash, usually using Hamming distance [26] (the number of positions at which the bits of the two hashes are different).

In this paper, we evaluate and make use of the following image hash functions, creating a 64-bit hash (except *HSVHash* producing a 42-bit hash):

(i) $aHash$ - The *aHash* algorithm first converts the input image to grayscale and then scales it down. As a 64-bit hash is generated, the image is scaled down to 8×8 pixels. Next, the average of all gray values of the image is calculated, and the hash is generated: For each pixel, if a pixel is greater than the average, it is set to 1, otherwise, it is set to 0.

(ii) **pHash** - This algorithm is similar to *aHash*, just uses Discrete Cosine Transform (DCT). After grayscaling, the DCT is calculated and it is reduced. Finally, each pixel is compared to some threshold and if it is greater than this value it is set to 1, otherwise, it is set to 0.

(iii) **dHash** - The difference hash algorithm works by computing the difference (i.e. relative gradients) between adjacent pixels. After grayscaling and resizing, the difference is computed and a difference matrix is built. Based on the difference matrix D, the hash is created (if D[x] $>$ D[x + 1] set 1 for bit x, else 0.).

(iv) **wHash** - The Wavelet hashing algorithm uses wavelet transformation. After grayscaling and resizing, the wavelet transformation is applied. Then, each pixel block is compared to the median of all gray values of the image and is set accordingly to 1 or 0.

(v) **HSVHash** - First, the image is converted to an HSV form. Then, the image is divided into blocks (or regions). For each block or region, a hash value is generated based on statistical measures of the HSV values, and the final hash is created from all block hashes.

(vi) **Image-Segmentation-Based hashing (*sHash*)** - This hashing algorithm proposed by [27] is meant to deal with cropped images. It relies on the assumption that even after cropping at least some large objects of an image will be left. Hence, this algorithm uses image segmentation to identify large objects of an image and for each object calculates one hash (using one of the previous hashing algorithms) of the bounding box. This way, rather than calculating one robust hash for an image, a set of robust hashes of the areas containing the largest objects of an image are calculated and stored individually. When comparing a new image to the original image hash, segment the image and calculate the hash of each segment. For each segment hash of the image, search for the hash of the original image segment with the lowest Hamming distance (meaning search for the most-alike segment of the original image) and compare the two hashes.

All hash algorithms described above were implemented in Python and made publicly available in [28].

III. Image Hashing Evaluation

After reviewing multiple image-hashing functions that can potentially be used for duplication detection, we were interested in evaluating what functions are valid for this task. Additionally, we wanted to see if the different hash functions

Fig. 1. Original image
Fig. 2. Rotated
Fig. 3. Cropped
Fig. 4. Added text
Fig. 5. Background color change
Fig. 6. Pixelated
Fig. 7. Grayscaled

perform differently on various image modifications, meaning if there are modifications that some functions capture better than others, and whether there exists one function that is better than others in all cases. Section III-A describes the dataset used for the evaluation, followed by section III-B evaluating the performance of the hash functions presented in Section II-D.

A. Dataset

As finding a reliable and labeled NFT duplication data set is not an easy task, and since we needed a label of not only whether the image is a duplication but also what type of duplication it is, we generated a database using Python's Pillow (PIL) [29] package. Following OpenSea's definition of copyminting, we implemented functions performing modifications on the original image (Fig. 1) of the following types:

- Flipping, rotating and mirroring an image (Fig. 2).
- Resizing, zooming, cropping, repositioning or modifying borders and edges (Fig. 3).
- Addition of unintegrated texts, logos or emojis (Fig. 4).
- Background-color change (Fig. 5).
- Pixelating the image (Fig. 6).
- Color swapping, modifying or saturating (Fig. 7).

To generate a full data set, we downloaded ten images of famous paintings and NFTs from OpenSea and generated 1000 duplications of each image, labeled into the previously presented criteria. Then, the data was split into two subsets where 80% of the data was used for the analysis of Section III-B evaluating the different image hash functions on the different duplication cases, and the remaining 20% of the data was used in Section IV-B for testing the different image duplication detectors present. We note that as the duplication image needs to look similar to the original one, we limited cropping to 50% and, added text or emojis to be smaller than 20% of the original image.

B. Image Hash Evaluation

After generating our dataset, we were set to perform an analysis of the effectiveness of image hash functions for NFT duplication detection. For each image, six hash values were calculated: *aHash*, *pHash*, *dHash*, *wHash*, *HSVHash* and *sHash*. Then, for each duplication criterion, we calculated the average Hamming distance between each duplication image and its original image. We also evaluated the average Hamming distance between an original image and other non-duplication images. The results appear in Table I, showing several trends regarding the hash functions:

aHash. The average Hamming distance of non-duplicate images from the original images using *aHash* is 28.22. Hence, it seems *aHash* performs well for capturing modifications that include adding logos and text and pixelating the image, having an average Hamming distance of 0.43-0.54. *aHash* additionally performs relatively well for resized, zoomed or cropped images, and when the image or background colors are modified (with a Hamming distance value of 10-13.5), but does not perform as well for flipped, rotated or mirrored images, having an average Hamming distance of 25.

pHash. The average Hamming distance of non-duplicate images from the original images using *pHash* is 30.9. Similar to *aHash*, *pHash* performs well for capturing modifications that include adding logos and text and pixelating the image, and does not perform as well for flipped, rotated or mirrored images.

dHash. This hashing function does not perform well for NFT duplication detection, as it has an average Hamming distance of ≈ 32 from all duplications while having a similar distance of 32.8 from non-duplicated images.

wHash. Similar to *dHash*, this hashing function does not perform well for NFT duplication detection, as it has an average Hamming distance of ≈ 32 from all duplications while having a similar distance of 32.56 from non-duplicated images.

HSVHash. The average Hamming distance of non-duplicate images from the original images using *HSVHash* is 7.22. Hence, *HSVHash* performs well for capturing modifications that include adding logos and text and pixelating the image (with an average distance lower than 0.53). As it is based on the color histogram, *HSVHash* does not perform well on color changes such as a change in the image or background colors, having Hamming distances of 3.81 and 4.62, respectively.

sHash. The average Hamming distance of non-duplicate images from the original images using *sHash* is 27.24. *sHash* performs very well at capturing texts, logos, or emoji additions, having an average Hamming distance of 0.97. This hash function is relatively bad at capturing images with colors swapped and flipped, rotated, or mirrored duplications, having an average Hamming distance higher than 16 for these categories.

As Table I presents only the Hamming average of each case and not how the distances are distributed, Fig. 8-Fig. 13

TABLE I
MEAN HAMMING DISTANCE OF HASH VALUES FOR PAIRS OF NON-DUPLICATED VS. FOR PAIRS OF DUPLICATED IMAGES

duplication criterion	*aHash*	*pHash*	*dHash*	*wHash*	*HSVHash*	*sHash*
non-duplicate images	28.22	30.89	32.81	32.56	7.22	27.24
Flipped, rotated or mirrored	25.05	29.60	32.82	32.57	1.61	24.9
Resized, zoomed, cropped, repositioned or modified borders and edges	10.35	17.59	32.22	30.79	1.21	12.52
Unintegrated texts, logos or emojis	0.43	0.85	31.85	31.83	0.53	0.97
Background-color change	13.58	11.83	32.45	31.68	4.62	8.78
Pixelated	0.54	0.37	31.87	32.14	0.39	5.45
Colors swapped, modified or saturated	7.34	6.35	32.15	31.73	3.81	16.32

TABLE II
ELIGIBILITY OF HASH FUNCTION TO DETECT DUPLICATIONS AND HAMMING DISTANCE SEPARATION POINT D

duplication criterion	*aHash*	*pHash*	*HSVHash*	*sHash*
Flipped, rotated or mirrored	bad	bad	good, $D = 3.33$	bad
Resized, zoomed, cropped, repositioned or modified borders and edges	bad	bad	medium, $D = 3.33$	medium, $D = 18$
Unintegrated texts, logos or emojis	very good, $D = 4$	very good, $D = 4$	very good, $D = 0.83$	very good, $D = 8.33$
Background-color change	medium, $D = 8.33$	good, $D = 12.5$	bad	very good, $D = 12.5$
Pixelated	very good, $D = 4.17$	very good, $D = 4.17$	good, $D = 1.67$	very good, $D = 16.67$
Colors swapped, modified or saturated	very good, $D = 8.33$	very good, $D = 16.67$	bad	bad

present the histogram of the Hamming distances of each hash function compared to each modification category. In blue appear the Hamming distances of duplicate images, while in red appear the Hamming distances of the non-duplicated images in this category. Additionally, for fit histograms, a dashed separation line was added. This analysis helps understand better the separation between duplicated and non-duplicated images in each criterion. We denote D as the Hamming distance separation point, which could be used as a threshold to classify an image as a duplicated image. A good separation indicates the hash function can be used to identify between duplications and non-duplications of this duplication criterion.

We define four ranks of separation, based on the percentage of possible mistakes using the best threshold possible: *very good*- if there exists a separation point D such that there are less than 5% misclassifications, *good*- if there exists D such that there are than $5-10\%$ misclassifications, *medium*- if there exists D such that there are than $10-20\%$ misclassifications, *bad*- if there does not exist any separation point D with less than 20% misclassifications. As *dHash* and *wHash* were not effective, they were neglected from Fig. 8-Fig. 13.

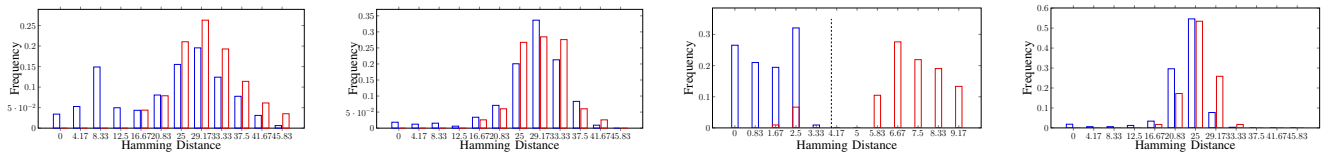
Fig. 8 shows no hash function can perfectly separate image duplications that were flipped, rotated or mirrored from non-duplicated images. That said, the *HSVHash* can identify duplicate images *good*, as more than 95% of duplicate images had a Hamming distance smaller than 2.5 compared to the original image, where less than 5% of the non-duplicated images had a Hamming distance that small. Using $D = 3.33$, there are around 9% misclassifications. This makes the *HSVHash* the only eligible hash function for detecting flipped, rotated or mirrored duplications.

For resized, zoomed, cropped, repositioned or modified borders and edges (Fig. 9), *pHash* and *aHash* are not a suitable fit for detecting duplications. For *HSVHash*, less than 10% of duplications have a Hamming distance larger than 3.33, while only 6% of non-duplicated images have a Hamming distance smaller than 3.33 from the original image. *sHash* performs $medium$ too, using $D = 18$, having 20% correct classifications. When adding logos or text to the images (Fig. 10), as they are a small part of the image, all four hash functions can separate between duplicated and non-duplicated images *very good*.

For background color change (Fig. 11), as *HSVHash* is color-based and affected by the colors of an image, this hash function cannot be of much use when trying to detect a background color change. Meanwhile, *sHash* has more than 95% of the duplicated images with a Hamming distance lower than 12.5, while more than 99.5% of non-duplicates had a higher distance, performing *very good* for $D = 12.5$. For a similar separation point, *pHash* can separate correctly 99% of the non-duplicated images and 91% of the duplicated images, performing $good$ in total. For a Hamming distance of 8.33, *aHash* can separate between 90% of duplicated images and 4% of non-duplicated images, performing $medium$ for $D = 8.33$.

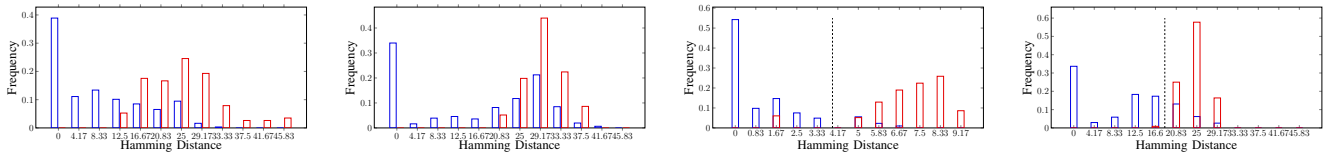
For pixelated images (Fig. 12), all hash functions can separate duplicated images from non-duplicated ones. *HSVHash* performs $good$, while all other hash functions perform *very good*. Finally, for duplications with Colors swapped, modified or saturated (Fig. 13), when applying *aHash*, more than 98% of duplications have a Hamming distance higher than 8.33, while no non-duplicate image had a lower Hamming distance, making it *very good* for detection duplications using $D = 8.33$. Similarly, *pHash* separates well this kind of duplication criterion, as more than 97% of duplications, have a Hamming distance of 16.67 or lower, while only 1% of non-duplications have a lower Hamming distance, performing *very good* too, for $D = 16.67$.

To conclude, different hash functions perform well on



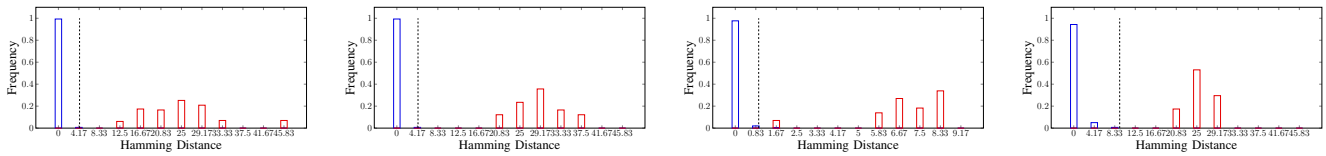
(a) *aHash* (b) *pHash* (c) *HSVHash* (d) *sHash*

Fig. 8. Flipped, rotated or mirrored



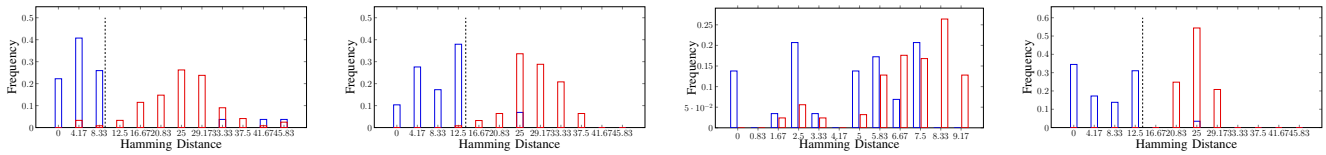
(a) *aHash* (b) *pHash* (c) *HSVHash* (d) *sHash*

Fig. 9. Resized, zoomed, cropped, repositioned or modified borders and edges



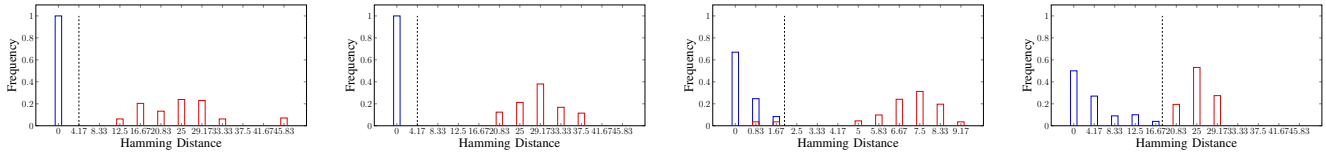
(a) *aHash* (b) *pHash* (c) *HSVHash* (d) *sHash*

Fig. 10. Unintegrated texts, logos or emojis



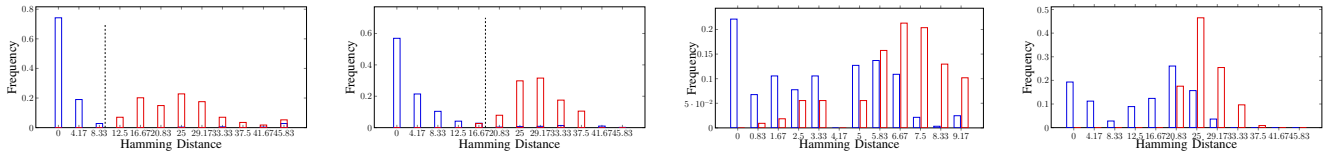
(a) *aHash* (b) *pHash* (c) *HSVHash* (d) *sHash*

Fig. 11. Background Color Change



(a) *aHash* (b) *pHash* (c) *HSVHash* (d) *sHash*

Fig. 12. Pixelated



(a) *aHash* (b) *pHash* (c) *HSVHash* (d) *sHash*

Fig. 13. Colors swapped, modified or saturated

different manipulations, and there is no single hash function that captures all manipulations and performs better than others. Table II summarises which hash functions are fit to separate between duplicated and non-duplicated images for each dupli-

TABLE III
COMPARISON BETWEEN DIFFERENT DETECTORS PERFORMANCE

Method	duplication correct classification percentage	non-duplicated correct classification percentage
minimal distance	95.07%	95.52%
2-minimal distance	81.83%	98.22%
combined hash detector	97.54%	95.52%
aHash	70.44%	98.07%
pHash	71.92%	98.87%
HSVHash	79.31%	94.52%
sHash	77.60%	98.13%

cation criterion.

IV. IMAGE-HASHING-BASED DUPLICATION DETECTOR

Section III shows the potential of image hashing to detect NFT duplications. Next, we created several image-hashing-based duplication detectors and compared them.

A. Creating Detectors

Using the analysis of Section III-B, we implemented a duplication detector (classifier) for each of the four hash functions: *aHash*, *pHash*, *HSVHash* and *sHash*. Our main interest was increasing the percentage of correctly identified image duplications while maintaining a low false-positive rate (less than 5% wrong classifications for non-duplication images). Based on the results of Section III-B, we created the single hash detectors, using the separation point D value that performs the best. That is, D is used as a threshold, and if the calculated Hamming distance of an image, compared to the original image, is lower than D, it is classified as a duplication. Otherwise, it is classified as a non-duplicated image. The values that produced the highest duplication detection percentage were: $aHash\_D = 8.33, pHash\_D = 16.67, HSVHash\_D = 3.33, sHash\_D = 18$.

As Section III-B shows there was no single hash function that performed well for all duplication cases, and no single hash function was overwhelmingly better than others, we suggest using multiple hash functions for duplication detection can increase the performance. We implemented three different duplication detectors:

- **Minimal Distance Detector.** In this detector, all four hash functions are used. Each hash function defines a separation point D, and the Hamming distances are calculated for each hash function. If for any of the functions, the distance is lower than D, the image is classified as a duplication. Otherwise, it is classified as a non-duplication image. Similar to the single hash detectors, the parameters used for this detector are $aHash\_D = 8.33$, $pHash\_D = 16.67$, $HSVHash\_D = 3.33$, $sHash\_D = 18$.
- **2-Minimal Distance Detector.** This detector is similar to the minimal distance detector, except now an image is classified as a duplication only if at least two hash functions indicate the image is duplicated. In our evaluation, we used similar values to the Minimal Distance.
- **Combined Hash Detector.** In this method, we tried to find some rules based on all four hash functions for the detection of duplications. For instance, when *aHash*, *pHash* and *sHash* all have a Hamming distance lower than 4.17, there does not exist any duplication. Similarly, when the distance of *HSVHash* is lower than 3.33 and the distances of both *pHash* and *sHash* are lower than 16.67, the image is almost certainly a duplication. Hence, based on the results of Section III-B we derived several rules that if one of them applies to an image, it is classified as a duplication.

B. Detectors Evaluation

As presented in Section III-A, 80% of the data was used for the evaluation in Section III-B which helped to tune the parameters of each detector, whereas the other 20% were used for testing the detectors to avoid biased results. Each detector was evaluated on both duplicated images and non-duplicated images. The main goal of our detectors was to increase the duplication detection percentage as much as possible while keeping a relatively low false-positive rate. Table III presents the performance of the four single hash detectors together with the minimal distance detector, 2-minimal distance detector and the combined hash detector. The table shows that though all four single hash detectors were able to classify correctly $70 - 79\%$ of the duplication images, the detectors using multiple hash functions achieved more than 81% success in detecting duplications. This shows that as each hash function captures different image manipulations, using multiple hash functions can improve duplication detection. Moreover, the minimal distance detector reached 95% success in detection duplications, and the combined hash detector even reached 97.5%. This performance is achieved while having a small false-positive rate, as both the minimal distance detector and the combined hash detector have a percentage of 95.5% for classifying non-duplicated images correctly.

V. CONCLUSION

In this paper, we presented the potential of using image hash functions for detecting NFT duplications. We considered six different image hash functions and examined the performance of each function on different image duplication types. Later, we created image duplication classifiers based on these results. Our analysis shows that image hash functions have the potential to be a fast and reliable method for detecting NFT duplications and that using multiple hash functions can increase performance. In future work, we intend to examine more hash detection functions and check if other distance metrics, for instance, cosine similarity, can help increase the performance. Additionally, intend to explore ways to improve our detectors, such as using preprocessing actions on images or using an ML model to improve the combined hash detector model. Finally. we intend to evaluate the performance of NFT duplications that include more than one manipulation and elaborate on efficient ways to detect an NFT duplication given a large NFT set rather than comparing it to all other hashes.

REFERENCES

[1] W. Entriken, D. Shirley, J. Evans, and N. Sachs, "ERC-721: Non-Fungible Token standard," https://eips.ethereum.org/EIPS/eip-721, 2018.

[2] M. Franceschet, G. Colavizza, T. Smith, B. Finucane, M. L. Ostachowski, S. Scalet, J. Perkins, J. Morgan, and S. Hernández, "Crypto art: A decentralized view," *Leonardo*, vol. 54, no. 4, pp. 402–405, 2021.

[3] N. Husin, A. Budiyanto, A. Karjono, M. Christiningrum, N. Yurindera, D. Nuryanti, T. Herninta, S. Adiawaty, and B. W. Wicaksono, "Analyzing the non-fungible tokens (NFT) implementation in digital music industry: A mix method study," in *IEEE International Conference on Cyber and IT Service Management (CITSM)*, 2022.

[4] S. Choi and J. H. Cho, "Intellectual property extensions of video content industry using non-fungible token," *Moving Image & Technology (MINT)*, vol. 2, no. 3, pp. 1–5, 2022.

[5] F. Regner, N. Urbach, and A. Schweizer, "NFTs in practice - Non-fungible tokens as core component of a blockchain-based event ticketing application," in *International Conference on Information Systems (ICIS)*, 2019.

[6] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[7] W. Radomski, A. Cooke, P. Castonguay, J. Therien, E. Binet, and R. Sandford, "ERC-1155: Multi token standard," https://eips.ethereum.org/EIPS/eip-1155, 2018.

[8] M. Shirole, M. Darisi, and S. Bhirud, "Cryptocurrency token: An overview," in *International Conference on Blockchain Technology*, 2020.

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Bitcoin white paper*, 2008.

[10] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (NFT): Overview, evaluation, opportunities and challenges," *arXiv preprint arXiv:2105.07447*, 2021.

[11] M. Nadini, L. Alessandretti, F. Di Giacinto, M. Martino, L. M. Aiello, and A. Baronchelli, "Mapping the NFT revolution: Market trends, trade networks, and visual features," *Scientific reports*, vol. 11, no. 1, p. 20902, 2021.

[12] Y. Faqir-Rhazoui, J. Arroyo, and S. Hassan, "A comparative analysis of the platforms for decentralized autonomous organizations in the ethereum blockchain," *Journal of Internet Services and Applications*, vol. 12, no. 1, pp. 1–20, 2021.

[13] T. Sharma, Z. Zhou, Y. Huang, and Y. Wang, "It's a blessing and a curse: Unpacking creators' practices with non-fungible tokens (NFTs) and their communities," *arXiv preprint arXiv:2201.13233*, 2022.

[14] D. Das, P. Bose, N. Ruaro, C. Kruegel, and G. Vigna, "Understanding security issues in the NFT ecosystem," in *ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[15] H. Yang, J. Callan, and S. Shulman, "Next steps in near-duplicate detection for erulemaking," in *International conference on Digital government research*, 2006.

[16] H. Yang and J. Callan, "Near-duplicate detection by instance-level constrained clustering," in *International ACM SIGIR conference on Research and development in information retrieval*, 2006.

[17] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang, "Efficient similarity joins for near-duplicate detection," *ACM Transactions on Database Systems (TODS)*, vol. 36, no. 3, pp. 1–41, 2011.

[18] Y. Ke, R. Sukthankar, L. Huston, Y. Ke, and R. Sukthankar, "Efficient near-duplicate detection and sub-image retrieval," in *ACM multimedia*, vol. 4, no. 1. Citeseer, 2004, p. 5.

[19] K. Thyagharajan and G. Kalaiarasi, "A review on near-duplicate detection of images using computer vision techniques," *Archives of Computational Methods in Engineering*, vol. 28, pp. 897–916, 2021.

[20] A. K. Jaiswal and R. Srivastava, "Detection of copy-move forgery in digital image using multi-scale, multi-stage deep learning model," *Neural Processing Letters*, vol. 54, no. 1, pp. 75–100, 2022.

[21] E. U. H. Qazi, T. Zia, and A. Almorjan, "Deep learning-based digital image forgery detection system," *Applied Sciences*, vol. 12, no. 6, p. 2851, 2022.

[22] P. Mehta, M. K. Singh, and N. Singha, "Near-duplicate image detection based on wavelet decomposition with modified deep learning model," *Journal of Electronic Imaging*, vol. 31, no. 2, p. 23017, 2022.

[23] M. Chevallier, N. Rogovschi, F. Boufarès, N. Grozavu, and C. Clairmont, "Detecting near duplicate dataset with machine learning," *International Journal of Computer Information Systems and Industrial Management Applications*, 2022.

[24] A. Moreaux and M. Mitrea, "Blockchain assisted near-duplicated content detection," in *Blockchain and Cryptocurrency Congress (B2C)*, 2022.

[25] P. Lee, M. Abubakar, O. Lo, N. Pitropakis, and W. J. Buchanan, "Non-fungible token fraud: Studying security issues and improvements for NFT marketplaces using hashing techniques," Preprint, https://www.researchsquare.com/article/rs-2573810/v1, 2023.

[26] R. W. Hamming, *Coding and information theory*. Prentice-Hall, Inc., 1986.

[27] M. Steinebach, H. Liu, and Y. Yannikos, "Efficient cropping-resistant robust image hashing," in *IEEE International Conference on Availability, Reliability and Security*, 2014.

[28] "An image hashing library written in Python," https://github.com/JohannesBuchner/imagehash, 2023.

[29] J. A. Clark, "Pillow," https://pillow.readthedocs.io/en/stable/, 2024.