Forensic Science International: Digital Investigation 52 (2025) 301878

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS EU 2025 - Selected Papers from the 12th Annual Digital Forensics Research Conference Europe

Beyond Hamming Distance: Exploring spatial encoding in perceptual hashes

Sean McKeown

School of Computing, Engineering, and the Built Environment, Edinburgh Napier University, Edinburgh, UK

ARTICLE INFO

Keywords:
Perceptual hashing
Semantic approximate matching
Distance metrics
Hamming distance
Image forensics
Content matching

ABSTRACT

Forensic analysts are often tasked with analysing large volumes of data in modern investigations, and frequently make use of hashing technologies to identify previously encountered images. Perceptual hashes, which seek to model the semantic (visual) content of images, are typically compared by way of Normalised Hamming Distance, counting the ratio of bits which differ between two hashes. However, this global measure of difference may overlook structural information, such as the position and relative clustering of these differences. This paper investigates the relationship between localised/positional changes in an image and the extent to which this information is encoded in various perceptual hashes. Our findings indicate that the relative position of bits in the hash does encode useful information. Consequently, we prototype and evaluate three alternative perceptual hashing distance metrics: Normalised Convolution Distance, Hatched Matrix Distance, and 2-D Ngram Cosine Distance. Results demonstrate that there is room for improvement over Hamming Distance. In particular, the worst-case image mirroring transform for DCT-based hashes can be completely mitigated without needing to change the mechanism for generating the hash. Indeed, perceived hash weaknesses may actually be deficits in the distance metric being used, and large-scale providers could potentially benefit from modifying their approach.

1. Introduction

As long as the Internet and World Wide Web have existed, criminals have exploited them for nefarious purposes. In the Digital Forensics context, this often relates to the distribution of Child Sexual Abuse Material (CSAM). Forensic analysts in many countries have been backlogged for some time (Beebe, 2009), but recent advances in generative AI may fuel a new wave of large-scale CSAM proliferation (Thiel et al., 2023). While in the early stages, the Internet Watch Foundation documents (Internet Watch Foundation, 2024) a month-on-month increase for AI CSAM reports for 2024.

One mechanism for keeping on top of the deluge of CSAM related media is to automate the detection of previously encountered images by way of perceptual hashing (a form of approximate semantic hashing), which models the visual properties of an image. Such approaches are robust to many image transformations, and they typically deal well with compression and other common image modifications that happen as a matter of course. Such technologies are already deployed for both law enforcement and cloud-scale service providers, such as Microsoft's PhotoDNA (Krawetz), and Facebook's PDQ (Facebook).

While much effort has been spent in modelling images with various techniques to generate a binary hash representation of the image (Hadmi et al., 2012), Normalised Hamming Distance is often the only distance metric which is discussed and evaluated. In contrast, fields such as Information Retrieval have many possible similarity and matching mechanisms, with a clear separation of document representation and similarity measurement (Manning et al., 2008). While Hamming Distance performs well for this task, it is a global measure of difference between two binary arrays, such that positional information and nuance is lost. As such, we investigate the following research questions in this work:

RQ1: Do spatial modifications to images produce observable patterns in perceptual hash strings?
RQ2: Are these patterns able to be exploited in the hash comparison process to produce better classificatiohn performance than Normalised Hamming Distance?

RQ1 is explored via bitwise hash analysis in Section 3, while alternative metrics are proposed and evaluated for RQ2 in Section 4. All code

E-mail address: S.McKeown@napier.ac.uk.

https://doi.org/10.1016/j.fsidi.2025.301878

Available online 24 March 2025

used in the experiments is available on Github.[1]

2. Background and related work

Forensic perceptual hashing is a derivative of the field of Content-Based Image retrieval (CBIR) (Tyagi, 2017), which is a mature field dating back to the 1970s. Images can be modelled via a wide-array of features, such as colour histograms (Swain and Ballard, 1991), texture and edge histograms (Manjunath et al., 2001), or frequency domain statistics (Venkatesan et al., 2000). Frequency domain transforms are often useful as the low-frequency components are robust to various image modifications (Fridrich, 1999). Coarse image representations using mean-block colour are also used for similar reasons (Steinebach, 2011). For digital forensic purposes, these hash representations must be representative of the image, but should not be reversible, or leak information about the source image, often involving some kind of quantisation in order to produce a binary hash array as a final output (Hadmi et al., 2012).

The forensic task is to identify images which are similar or identical to a target hash, distinguishing these from unrelated images. This involves the use of a distance metric, which is used to calculate the difference between two hashes and return a value which can be compared to a chosen 'threshold'. If the distance is lower than this threshold, the image matches, otherwise it is considered to be an unrelated image. In CBIR, the relative richness of the image representation indices (not simply binary hash arrays), and the looser task of matching related content, allows for a variety of comparison approaches, such as Cosine Distance, Euclidean Distance, Hamming Distance, and L1/L2 Distance (Tyagi, 2017). In contrast, perceptual hashes are often compared using only the Normalised Hamming Distance (Hadmi et al., 2012), which is simply a ratio of the number of bits which differ between two hashes.

Selecting an appropriate distance threshold is predicated on the distribution of intra-image comparisons (images that should match), and inter-image comparisons (unrelated images) being separable (Zauner, 2010; McKeown et al., 2024). Otherwise, if distributions overlap, any threshold is a trade-off of False Positives vs. False Negatives in the matching task. Common perceptual hashes are resistant to image transformations attacks such as compression, re-scaling, colour modification, blurring, filtering etc (Zauner, 2010; Breitinger et al., 2013; Drmic et al., 2017; Hamadouche et al., 2021; McKeown and Buchanan, 2023), while rotating, mirroring, cropping, and adding borders may cause difficulty for some hashes (Zauner, 2010; Breitinger et al., 2013; Drmic et al., 2017; McKeown and Buchanan, 2023), or be worst-case machine scenarios where inter- and intra-image distributions overlap completely (McKeown and Buchanan, 2023).

Prior work typically evaluates the mechanism for generating hashes and their response to transforms, but does not often make changes to how images are compared. For instance, algorithms typically have a weakness to certain image content (e.g., high-frequency patterns, or low-frequency gradients (McKeown et al., 2019)), or to particular transforms, such as rotation (Breitinger et al., 2013), as mentioned above. It is possible to pre-process images to mitigate this weakness, for example, mirroring the image so its darkest corner is always at the top-left, completely defeating mirroring attacks (Steinebach et al., 2012), but this is rarely discussed in practice. Similarly, noting that not all algorithms derive equal utility from the hash-bits in a hash array, weights can be applied to hashes prior to calculating their Hamming Distance (Steinebach et al., 2012). This can improve performance, generally, for a particular algorithm, or potentially mitigate against worst-case scenario transforms such as mirroring (McKeown et al., 2024).

Despite these modifications to the process, Normalised Hamming Distance is still prevalent for most algorithms, without pre-processing, re-weighting, or weakness mitigation being built-in to the process. Hamming Distance is the de-facto approach for deep-learning approaches also, however they sometimes make use of Cosine or Euclidean distance (Singh and Gupta, 2022). This means that the hashing algorithms themselves are left to trade-off localised vs. global features, with a global comparison being made afterwards.

3. Spatial data encoding in perceptual hashes

A Normalised Hamming Distance of 0.25 simply indicates that an aggregate of 25 % of the bits are different, whether this 25 % is clustered at the beginning/end of the hash, or spread across it. Some images may be completely different, in which case, given the design of a perceptual hash, we would expect bits to be more or less differing completely arbitrarily, with a 50/50 probability of being the same by chance. However, for regional changes, such as cropping, adding watermarks, or transpositions such as rotation, there is some degree of perceived similarity, or commonality, as semantic content is preserved. Therefore, if perceptual hashes typically encode some degree of positional information/locality of change, it can be exploited to improve image matching performance, and alternatives to Hamming Distance would be entirely appropriate. This section describes the investigation of spatial encoding in perceptual hashes for RQ1, and how various image transformations map to changes in their respective binary hash arrays.

3.1. Approach

To detect patterns in hashes, the Hamming Distance itself is not useful. Instead, we must pay attention to the individual bit positions within each hash in order to determine how often they are correct when classifying matches. The counts of these correct classifications (per bit) can then be aggregated across all images and plotted (for a given hash/transform) to determine if any transform specific patterns emerge, which could indicate positional data encoding.

The experiment was conducted with the following process: **i)** Generate image transforms which deliberately regionalise change; **ii)** Derive perceptual hashes; **iii)** Calculate Hamming Distances; **iv)** Calculate a weights vector, w. **v)** Visualise weight vector, w.

We make use of the PHASER (McKeown et al., 2024) perceptual hashing evaluation framework to implement transforms, generate hashes, calculate inter- and intra-distances, and provide bit-weight vectors. This latter functionality, and method of exploring the problem, builds on the paper describing PHASER where it was observed that providing weights to Hamming comparisons can improve overall matching performance, particularly for frequency domain Discrete Cosine Transform (DCT) based hashes (phash and PDQ, in this case) on mirror transforms (McKeown et al., 2024). The visualised weight vectors in those cases had a piano-like appearance, indicating that individual bits alternated between accurate and innacurate classiifcation contributions. We take a similar approach here, but instead focus on bits which do not prove useful for matching (i.e., low bit-weight), as this will flag up any redundant/shared information we embed in the transforms (see Sections 3.1.2 and 3.2).

3.1.1. Selected hashing algorithms

A variety of perceptual hash functions were chosen for this work. Firstly, we make use of the hash implementations in the Python Imagehash library, selecting the **ahash** (Average hash), **dhash** (Difference Hash), **dhash_vertical** (Dhash in vertical mode), **phash** (DCT-based Perceptual Hash) and **whash** (Wavelet Hashing) algorithms. Additionally, to represent wider efforts in the industry, Facebook's **PDQ** (based on phash) and Apple's **Neuralhash** (Convolutional Neural Network) were also tested, with the latter serving as a representative deep-learning solution.

To some degree, a level of positional encoding is expected due to how the hash-bits are generated in many spatial-domain hash approaches.

[1] https://github.com/AabyWan/PHASER/tree/main/paper.

For example, with ahash, images are downscaled to 8 × 8, each pixel is then compared to the mean pixel value, generating a binary determination of higher/lower, resulting in a 64-bit hash array. Similarly, dhash compares the left and right pixel relative to each pixel, (or above and below for vertical mode) to generate bits.[2] whash and phash behave slightly differently, with the reduced image size (32 × 32 for phash) first undergoing a frequency domain transform (Haar Wavelet and Discrete Cosine, respectively), with pixels being derived from a comparison with the median coefficient value from the resulting transform. The phash DCT matrix is first reduced to the top-left 8 × 8 coefficients, resulting in the same 64-bit hash size as the other Imagehash implementations. PDQ works very similarly to phash, with some modifications, but produces 256-bit fixed-size hashes, with a diagonally flipped coefficient arrangement.

The spatial techniques above (ahash, dhash) should theoretically encode positional information as bits are derived from pixel-level comparisons, with the order of bits directly corresponding to the relevant pixel position in the thumbnail image. As whash, phash and PDQ work with frequency domain coefficients, they were not expected to produce the same level of granularity, but they should still capture some relevant spatial properties. Neuralhash, however, uses a Convolutional Neural Network, where extracted features are more abstract, and bits are assigned by comparing the feature vector's positions (Struppek et al., 2022). In this case, spatial information is likely non-trivially recoverable, which we expect to be the case for most deep-learning approaches.

3.1.2. Dataset and transform selection

A list of image transforms which have particular spatial properties is provided in Table 1. Most of the transforms are common in the perceptual hash evaluation literature, with border, crop, and rotate being shown to be relatively tricky cases. Common elements in an image, such as a shared border, can skew the inter-image distribution of unrelated images, which is potentially problematic (McKeown and Buchanan, 2023), but in the case here, shared information is useful for identifying positional encoding. Similarly, the mirroring and rotation transpositions move existing content (with some interpolation in the context of rotation as pixels are square), but ultimately preserve the semantic content of the image. Other transforms, such as compression or scaling were not included as they distribute changes across the image, while the focus here is on changes made to specific regions of the image.

Table 1
Spatial transforms used to tease out positional information encoding in hashes.

Border	A white border (0.1× target height/width) around the edge of the image. Draws a rectangle over the edges of the image without extending dimensions
Crop (scaled factor)	Apply a crop box aperture to cut off fractions of the image. In this case, 25 % of the left and top, and 10 % all around the image. Reduces overall image area.
Composite Image Embedding	Embed a fixed image, common to all targets. The image was scaled down to fit in the new dimensions of the target location, which could be a quadrant (e.g., top-left) or top/bottom/left/right.
Mirroring	Horizontal (x-axis) and Vertical (y-axis) image transposition.
Rotate	Rotate the image 15° counter-clockwise, embed black background in the empty-space created by a non-rectangular image.
Watermarking	Embed a small text logo image (0.1× target height, minimum 40 pixels) in the bottom-right corner. No transparency used, black background inserted.

[2] To avoid the edge of the pixel matrix having no comparison, images are 9 × 8 for horizontal and 8 × 9 for vertical mode, respectively. Comparison then begins with the second pixel, effectively skewing the result slightly.

The only transform not common in the literature is the composite image approach, which makes use of a fixed secondary image to embed in the target image for transformation. This is similar to visible logo watermarking, but was designed to be more apparent and take up either a complete quadrant (e.g., top-left) of the target, or, with less granularity, simply the left, right, top, or bottom of the image. Results reported here are for embedding a low-frequency colour gradiant image, but we found similar results for high-frequency content with blades of grass.

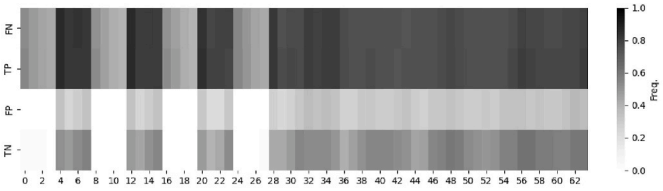
All transformations were then applied to a 20,000 image subset (selected via a fixed seed random sample) from the Flickr 1 Million dataset (MIRFLICKR Download), which was a large enough sample to establish patterns in the dataset.

3.2. Structural hash information

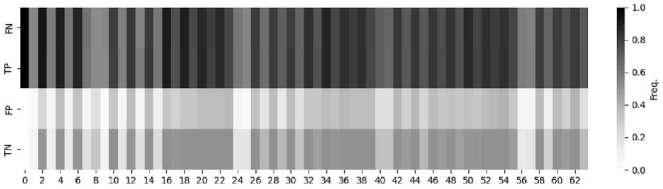
Composite images, where embedding location is controlled, are the simplest way to determine if positional information is preserved clearly. This common data provides no information to discriminate between images, whether they match or not. If specific bits in the hash correspond to this redundant region of the image, then the weighting process should generate a very-low weight consistently across all images.

Fig. 1 visualises weights for these confusion matrix quadrants for the top-left corner image embedding transform. For 1a, representing ahash, we can see that there is a repeating block pattern introduced, while 1b, representing phash, has a less obvious pattern. Wrapping these arrays to square matrices makes it much easier to see why this is the case, with the equivalent matrices being depicted on the left of Fig. 2. In this case we can now see a clear top-left block for ahash, while phash presents a tartan-like hatched pattern.

Additional transforms are depicted in the remainder of Fig. 2, with the top row depicting representative samples from the spatial hashes, and bottom row the DCT-based hashes. PDQ and phash patterns are essentially the same, with PDQ containing more rows/columns overall and placing the low-frequency coefficient bits at the bottom-right, while phash contains these in the top-left. Spatial domain techniques produce very similar graphs overall, with dhash's horizontal or vertical bias skewing the diagram occasionally. Of note, whash follows the spatial domain patterns, rather than those of the other frequency domain



(a) ahash - Confusion matrix quadrant arrays - Top-left Composite



(b) phash - Confusion matrix quadrant arrays - Top-left Composite

Fig. 1. Visualisation of weights for each bit in the hash, corresponding to their relative contribution to correct classification across confusion matrix quadrants, aggregated for all images. Values are depicted for the top-left composite image embedding. From top to bottom: FN, TP, FP, TN. The top two classes should match (positive) while the bottom classes should not (negative). Black is a weight of 1, white is a weight of 0.

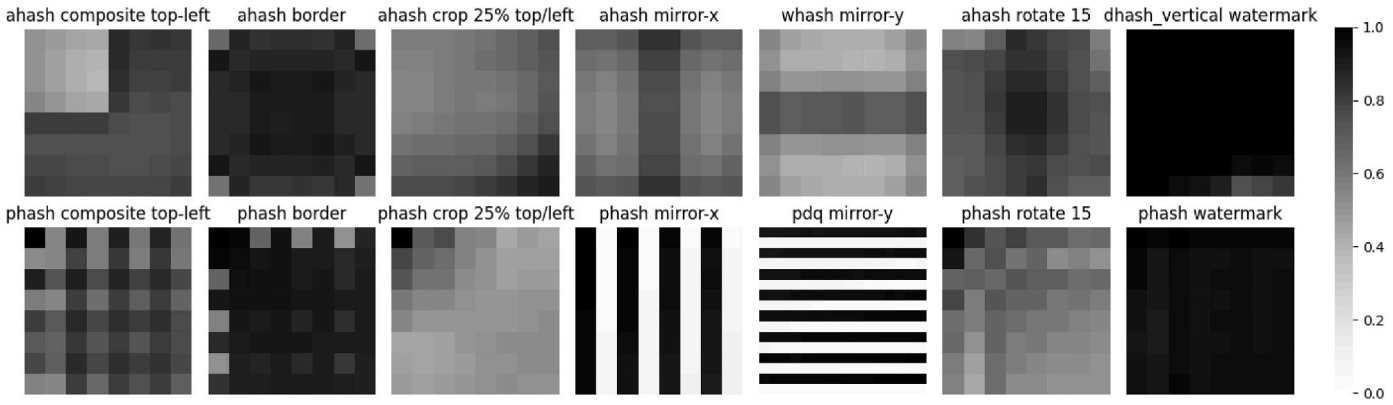


Fig. 2. Bit-weight matrices for various hashes and transforms as calculated for True Positive matches only.

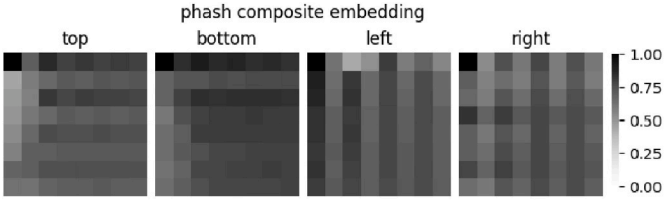
approaches (phash and PDQ). Smaller common items, such as visible watermarks, produce less pronounced patterns and may not have much impact on the hashes.

Both border and rotate transforms centralise the weight for spatial hashes, with corners being less important (as they move farther during rotation, and blend other common information in downsampling at border edges). These transforms are less similar for the DCT-based phash and PDQ, however, with a grid-like pattern appearing for borders, and a grid-pattern appearing for rotation, albeit with a focus on the lower-frequency coefficients. In the spatial domain, weight is focused away from cropped areas, with DCT approaches again shifting weight towards lower frequencies. Finally, mirroring an image on the x- or y-axis focuses weight in the centre of the image along the axis of reflection for spatial approaches, and for DCT results in either alternating vertical bars (mirror-x) or rows (mirror-y).

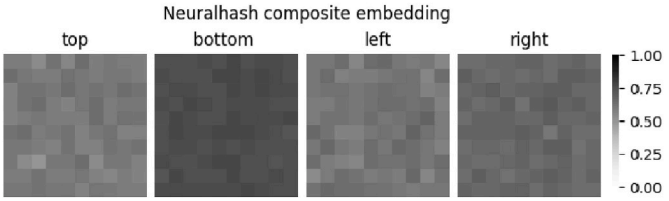
Fig. 3a expands on the DCT patterns as it is useful to visualise how it impacts their row/column patterns. Left/Right composites form horizontal bars, while top/bottom composites form vertical bars. However, it should be noted that these are far less clean than those for mirroring an image. Together, these transforms suggest that vertical information should be weighted on alternating rows, with alternating columns for horizontal information.

While these findings are not necessarily surprising based on the descriptions of the hashing techniques in Section 3.1.1, the strength of their effect on biasing the hash is clear. Hamming Distance glosses over these structural components, and at least heuristically, it would make sense to give a higher weight to the centre of spatial hashes, while paying more attention to lower-frequency coefficients and the patterns of rows and columns within DCT-based approaches.

As expected, the behaviour of Neuralhash differed from the spatial and frequency transform techniques as it takes a completely different approach to generating hashes. Visually, there is no clear partition for left/right/top/down in the hash (Fig. 3b), with weighting matrices resembling noise patterns, for all transforms. As such, our answer to RQ1 appears to be that the tested 'shallow' approaches do indeed encode spatial information, while deep-learning approaches may not.



(a) phash weights.



(b) Neuralhash weights with two bits of median-value padding at the beginning/end of the array.

Fig. 3. True Positive bit-weight matrices for phash and Neuralhash for various composite embeddings.

4. Exploring alternative metrics

With confirmation of spatial encoding, we now move on to RQ2 in order to determine if these patterns can be leveraged to improve matching performance beyond the locality insensitive approach taken by Hamming Distance. The literature did not lend itself well to this endeavour, however, as existing approaches such as Structural Similarity Index Measure (SSIM) are intended for arrays of pixel values, and in testing it did not appear to have utility when comparing binary matrices of hashes. Similarly, some small scale experiments on other spatial distance metrics (Bray–Curtis, Canberra, Cosine, Euclidean, Manhattan, Minkowski) in the Scipy library produced almost identical results to Hamming Distance.

We therefore prototype new distance metrics appropriate for the task. The general requirement was to provide an interface that behaves like those already in the Scipy library: **i)** Inputs u and v corresponding to same-length binary arrays, and **ii)** return a normalised distance value between 0 and 1, where 0 is the same hash, and 1 is its inverse.

To make use of spatial properties, binary arrays are converted to square matrices, corresponding to the downscaled image, or, for DCT-based hashes, the coefficient matrix. This does mean that some arrays may need padding if they do not form even squares. Additionally, a wider goal was to produce appropriate inter-image distance distributions (Zauner, 2010; McKeown and Buchanan, 2023), with a mean/-median of 0.5 for unrelated images. However, the superseding goal is to be able to separate distributions of distance values for inter-image comparisons and intra-image comparisons (original to transform) to facilitate strong matching performance.

Three approaches are considered: **i)** Normalised Convolution Distance, **ii)** a DCT-aware Hatched Matrix Distance, and **iii)** 2-D Ngram Distance.

4.1. Normalised Convolution Distance

In this context of image processing, two dimensional convolutions involve passing a kernel filter (usually a 3×3 or 5×5 matrix), over each

pixel in an image. The filter specifies weights to be applied to the neighbours, as well as the target pixel, accumulating the sum of the weighted matrix. This accumulation is transferred to a new matrix of the same size as the original image, where the value is placed in the same location as the target pixel. In image processing, with the appropriate filters, it can be used to detect visual features, such as edges and corners, which ultimately form several layers in a Convolutional Neural Network (CNN) (Hijazi et al., 2015), providing multiple 'views' of the image.

This approach provides a fast way to sum values based on regional information, as the kernel takes into account surrounding values in the matrix. In our use case, we apply the convolution to a **difference matrix** of the two input hashes, rather than the images or hashes themselves. The process is as follows:

1. Reshape input arrays to matrices and generate the logical XOR (difference) matrix.
2. Perform a convolution on the XOR matrix
3. Sum the values in the convolution output and normalise by the maximum possible value

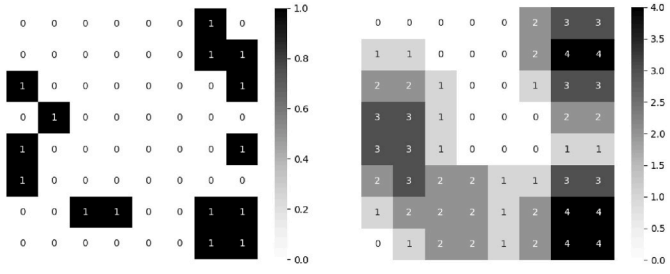
As XOR captures when the bits are different, we essentially build a map of the hash differences (1s). When convolving this matrix, regions with adjacent differences will create larger values in the output matrix than those with spread differences, or a smaller number of differences. To normalise, the matrix is divided by the maximum possible convolution sum (of an array of all 1s for the same size and filter). An example difference matrix and the subsequent convolution matrix are depicted in Fig. 4.

A variety of filter kernel sizes and values were tested. Larger filters take into account a larger spatial region around the target bit, while smaller kernels are more localised. Equally, there is a decision to be made with regards to the filter weights as they can either bias towards or against adjacent or distant values. We tested a variety of filter sizes (2×2 to 6×6) and various combinations of weights. Filter sizes of 2, 3 and 6 perform roughly equivalently here, while we settled on a filter matrix of all ones.

4.2. Hatched Matrix Distance

Noting that the DCT-based approach often creates a hatched pattern, it seems appropriate to treat the hash matrices for PDQ and phash in terms of their rows and columns. The weight patterns also appear to make a distinction between whether the values are in even or odd columns/rows, which should also be taken into account. The process is as follows:

1. Extract rows/columns from the hash, noting whether the indices are even or odd. Concatenate even rows, even columns, odd rows, and odd columns into their own respective arrays.



(a) ahash difference matrix, rotate 15 degrees vs. original (b) ahash convolution matrix, rotate 15 degrees vs. original

Fig. 4. Example intra-image difference and convolution matrices for ahash, with a filter kernel size of 3×3, with all ones.

2. Calculate the Hamming Distance for each of the four arrays between each hash. (i.e., Hash A's even row array to Hash B's even row array, etc.)
3. Calculate the minimum distance between rows (i.e., `min(even-row_dist, oddrow_dist)`) and columns.
4. Return the mean of the minimum row/column distances (i.e., `mean(minrow, mincol)`)

By taking the minimum value for odd/even rows and columns, we can account for whichever of the values is dominant, with the assumption that the values would not be closer by accident. The mean of row/column values allows us to mediate between the two, as they are often both important in hatched weight patterns (as with the phash composite top-left case). Alternative versions which compared individual rows/columns (as opposed to concatenating them together), with both Hamming and Cosine distances, proved less effective overall. One limitation is that this approach does not necessarily account for the cases where the hatched pattern does not appear, where the emphasis is on the low-frequency coefficients, as with cropping the top and left of an image.

4.3. 2-D Ngram Cosine Distance

Normally an N-gram is a sequence of n items adjacent to one another, often used in Natural Language Processing (NLP) to capture a sliding window of n adjacent words, while adjacent bytes are used for processing binary file data. In this case, to capture spatial information, we use two-dimensional $N \times N$-grams, essentially $n \times n$ matrices of hash bits. The process is as follows:

1. Reshape hash arrays to matrices, and for each matrix accumulate an array of $n \times n$ Ngrams. Sliding windows are overlapping, such that the rightmost column of the first window forms the leftmost column of the next.
2. For each hash, flatten the array of Ngrams.
3. Calculate the Cosine Distance between each flattened array.

N-gram sizes from 2×2 to 6×6 were tested, with a width of 2 performing the best overall, though we did not test non-overlapping windows.

4.4. Evaluating matching performance

Evaluating the prototype metrics involved selecting a random 250,000 image subset of the Flickr 1 Million Dataset and applying the transforms in Table 1. Some transforms (most of the Composite permutations, and the 10 % all-sides crop) are omitted for brevity, with CropTL and CompTL referring to the Crop 25 % Top and Left, and Composite Top-left transforms, respectively. Mirror-y is dropped as it is not particularly practical as it hinders viewability in a content-preserving scenario.

We made use of the PHASER framework to calculate Intra- and Inter-image distances and the corresponding Area Under the Curve (AUC) for the ROC (Receiver Operating Characteristic) plot, calculated for each algorithm/transform/metric triplet. Less difficult transforms, such as re-scaling, compression and enhancement (contrast, colour, sharpness), were also tested, but omitted here as they are relatively trivial. Neuralhash was included in the experiment even though it lacks obvious spatial encoding to determine if it would still benefit from different metrics.

Evaluation results are depicted in Table 2. The AUCs for Hamming Distance are presented as is, while AUCs for the prototype distance metrics are recorded as percentage point differences to the Hamming Distance for the sake of readability. This is calculated as: $(AUC_{metric} - AUC_{Hamming}) \times 100$.

A 1%pt change indicates a positive 0.01 shift in the AUC over Hamming Distance. It should be noted that small changes at high AUCs

Table 2
Area under the ROC curve for various non-trivial transforms. Hamming Distance and percentage point difference vs. the Hamming AUC for: Convolution Distance (4 × 4 filter), Hatched Matrix Distance (Hamming), and 2 × 2gram with Cosine distance. Differences are highlighted with bold for **>1** % and underline for < 1 %. 1%pt = 0.01 in the AUC.

		AUC	%pt Diff to Hamming		
Hash	**Trans.**	**Hamming**	**Conv4_4**	**Hatch**	2gram
ahash	CropTL	0.829	0.2	0.0	−5.4
	MirrorX	0.766	−0.3	−2.6	−3.6
	Rotate	0.919	**2.5**	−0.1	−0.8
	Border	0.971	0.7	−0.3	0.6
	CompTL	0.527	**3.8**	0.5	−4.8
dhash	CropTL	0.641	−0.6	−0.7	−4.9
	Mirr.X	0.618	−1.1	−3.4	**1.0**
	Rotate	0.808	**7.6**	−1.0	**4.3**
	Border	0.995	0.4	−0.3	0.1
	CompTL	0.992	0.4	−0.4	−2.9
dhash vertical	CropTL	0.646	−0.4	−0.4	−4.3
	MirrorX	0.801	−0.1	−2.9	−6.5
	Rotate	0.780	**8.4**	−1.0	**3.6**
	Border	0.992	0.6	−0.4	0.2
	CompTL	0.989	0.4	−0.6	−5.5
Neural hash	CropTL	0.996	0.0	−0.1	−0.4
	MirrorX	0.930	−0.1	−0.4	−1.0
	Rotate	0.988	0.0	−0.2	−0.4
	Border	0.999	0.0	0.0	0.0
	CompTL	0.844	0.9	−0.7	**2.7**
PDQ	CropTL	0.527	−0.2	−0.1	−0.9
	MirrorX	0.515	0.9	**48.5**	**1.6**
	Rotate	0.502	**1.7**	**3.0**	**2.7**
	Border	1.000	0.0	0.0	0.0
	CompTL	1.000	0.0	0.0	0.0
phash	CropTL	0.586	−2.5	−0.2	−0.3
	MirrorX	0.496	**2.3**	**49.1**	**1.1**
	Rotate	0.675	−0.1	**1.5**	−0.9
	Border	1.000	0.0	0.0	0.0
	CompTL	0.944	**2.4**	0.6	**1.3**
whash	CropTL	0.821	0.3	0.0	−2.1
	MirrorX	0.744	−0.1	−2.9	−1.7
	Rotate	0.904	**3.1**	−0.2	**1.1**
	Border	0.921	**1.4**	−0.5	**3.6**
	CompTL	0.604	**4.7**	0.2	−3.4

can still translate to decreasing FP and FN rates by orders of magnitude for a given distance threshold.

For the spatial hashes (we include whash in this category for our purposes), the convolution approach is often positive, with the rotate and composite top-left approach benefiting the most across all hashes, achieving uplifts of around 8%pts for both dhash variants. The border transform also sees an uplift for all spatial hashes, despite their already solid performance. The only large loss here is mirroring for dhash, though this already performs poorly. Interestingly, the relative benefits for phash and PDQ seem less-well aligned with each other here. There is some upside for the Neuralhash composite case, which is the worst transform for it in this set. However, the movement is relatively small, slightly less than 1%pt, leaving a lot of room for improvement.

Hatch Matrix Distance was intended for the DCT-case, and seems to be slightly worse across the board than Hamming Distance for spatial approaches. For the DCT hashes, there is essentially no downside, but the mirroring difficulties completely evaporate for both phash and PDQ, to the point that they far exceed the performance of the spatial approaches. The AUC has essentially doubled, demonstrating that the information in the hashes themselves is enough to distinguish between inter- and intra-image classes, despite representing a worst-case transform for Hamming Distance comparisons. Neuralhash, overall, seems to lose out consistently here.

While the prior two metrics had little upside for Neuralhash, it seems to have more movement for the Ngram approach, with a 2.7%pt uplift on the composite embedding, at the slight expensive of other transforms. Generally, Ngrams are not a good trade-off for spatial approaches, though the rotate transform does seem to benefit across all hashes. PDQ only sees change in its worst-off transforms, but the changes are too insignificant to matter. As with the other metrics, phash sees a small increase on a transform it already handles fairly well, i.e., composite embedding.

Larger hash sizes for the Imagehash library algorithms were also tested, scaling from the default 64-bit to 256-bit. Generally there are a few cases where this has a relatively substantial impact (2–5 %pt) for some transforms, but they trade off large losses against another transform (up to 20–40 % in some cases).

4.5. Distributions and computational complexity

The inter-score distributions of original images remains appropriate for all metrics (normally distributed around 0.5), with Normalised Convolution distance tracking the Hamming Distance distribution almost exactly. Ngrams sit slightly higher than the Hamming distributions at around 0.51 for most metrics, though it is a little lower for whash (0.49–0.50) and considerably higher for dhash_vertical (0.55). Larger Ngrams trend down towards 0.5. Both Ngram and Convolution distance metrics smooth out any spikes that Hamming Distance produces for certain algorithms. Due to the minimum distance comparison in Hatched Matrix Distance, unrelated images are slightly more similar than for other metrics, producing a mean/median around 0.45 for all algorithms. Despite this, inter- and intra-classes are still neatly separated allowing for discrimination between them.

For a rough complexity demonstration, timed benchmark data for the distance metrics are presented in Table 3. Code was written in Python, with the `scipy.spatial.distance` module being used for Hamming and Cosine distances, while convolve is from `scipy.ndimage`. The maximum value for normalisation was pre-calculated for Convolution Distance and passed to the function. Distances for 100,000 random 64-bit array pairs were calculated for ten runs, with the mean time in seconds being reported in the table. No attempt was made to vectorise the base functions, and as such the array slicing operations used by Hatched Matrix Distance and Ngram Cosine Distance make them very slow in Python, putting them around 10–20× slower than Hamming Distance. The Scipy implementation of convolution is likely reasonably efficient, but overall the Convolution Distance is still around 4× slower than Hamming. As Hatched Matrix Distance is potentially very useful for DCT-based algorithms, an attempt was made to optimise it by pre-compiling it using the `numba` module, with `fastdist` being used for its Hamming calculation. This resulted in an order of magnitude speed-up, but it still lags behind Hamming Distance by about 1.5×.

5. Conclusion and future work

Overall, we can consider both research questions to have been answered clearly. For RQ1, pertaining to whether or not spatial image data is encoded into bit positions in perceptual hashes, Section 3.2 demonstrates that this is indeed the case for a set of popular spatial and frequency domain hashes, but not for the tested deep-learning model. The patterns and information encoded in said bit positions are essentially averaged out with a global distance measure such as Hamming Distance. This additional information could potentially be used to

Table 3
Python benchmarks for 100k randomised 64-bit array pairs. Ryzen 5900x single-threaded, mean of 10 runs.

Metric	Benchmark Time (s)
Hamming	0.79
Normalised Convolution (4 × 4)	3.41
Hatched Matrix	18.28
2gram Cosine	9.85
Optimised Hatched Matrix	1.25

improve perceptual image matching classification performance by incorporating them into a locality sensitive distance metric, which was the focus of RQ2 (Section 4).

To explore RQ2, we propose three prototype distance metrics (Hatched Matrix, Normalised Convolution, and 2-D Ngram Cosine), modelling hashes as two-dimensional, squared, hash matrices. Providing a positive result for RQ2, evaluation of these metrics indicate that significant gains can be made over Hamming Distance. In the case of Hatched Matrix distance for phash and PDQ, their weaknesses to mirroring attacks with Hamming Distance were essentially negated. This turned a worst-case transform into one of the best-cases, at no cost to other transform classes and with no change in the hashing process. Indeed, transforms which are poorly handled cases may be a result of a global distance metric, rather than being a feature of the underlying hash algorithm. Less strikingly, the tested convolution-based distance metric outperforms Hamming Distance across the board, though it does trade-off small losses in some transforms. The Ngram approach is less compelling, though it does seem to improve match performance against rotated images.

A one-sized fits all approach to perceptual hash comparison is perhaps ill advised, particularly as all tested algorithms have their own characteristics, though there are shared properties between them. While they all seem to track Hamming Distance reasonably well on aggregate, suggesting that it does a decent job, there is clear room for improvement. The solutions presented here are a first foray into accounting for these spatial hash properties, and while the Hatched Matrix approach presented here works well for certain hash patterns, it does not necessarily benefit transform patterns (such as those weighted more heavily for low-frequency DCT coefficients). We also do not explicitly test image content which causes difficulty for certain hashes, (such as gradients for block-mean approaches, and high-frequency patterns for DCT-transforms), and it is now in question whether these are indeed features of the hashes, or their Hamming comparisons.

Future work could look to further explore the metric space by focusing on the transform-weight response of specific algorithms in a wider sense, as it appears spatial approaches could be improved by more closely considering the centre of the image in many cases. DCT and other frequency transforms can be explored for their coefficient distributions as well as the higher-level hatched patterns presented here. Standardising pre-processing approaches could also be considered, though this would require the rebuilding of hash databases utilising the underlying hash mechanism. Additionally, none of the approaches here consider that even in cases where a block of an image is different, not all bits will change, as the algorithms typically aim for any given segment of data to be a 50/50 distribution between 1s and 0s. As such, a probabilistic modelling approach may be appropriate here to take into account the probability of a bit flipping or staying the same by chance.

Finally, we suggest that developers of content detection software investigate their choice of distance metric for content detection, as their current approach may be sub-optimal, with content-preserving transformations defeating their content moderation and detection tooling unnecessarily.

References

Beebe, N., 2009. Digital forensic research: the good, the bad and the unaddressed. In: IFIP International Conference on Digital Forensics. Plus 0.5em Minus 0.4em. Springer, pp. 17–36.

Breitinger, F., Liu, H., Winter, C., Baier, H., Rybalchenko, A., Steinebach, M., 2013. Towards a process model for hash functions in digital forensics. In: International Conference on Digital Forensics and Cyber Crime. Plus 0.5em Minus 0.4em. Springer, pp. 170–186.

Drmic, A., Silic, M., Delac, G., Vladimir, K., Kurdija, A.S., 2017. Evaluating robustness of perceptual image hashing algorithms. In: 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Plus 0.5em Minus 0.4emOpatija. IEEE, Croatia, pp. 995–1000. May.

Facebook. PDQ. [Online]. Available: https://github.com/facebook/ThreatExchange/tree/main/pdq/python.

Fridrich, J., 1999. Robust bit extraction from images. In: Multimedia Computing and Systems, 1999. IEEE International Conference on, vol. 2, pp. 536–540. Jul.

Hadmi, A., Puech, W., Said, B.A.E., Ouahman, A.A., 2012. Perceptual image hashing. In: *Watermarking-Volume 2*. Plus 0.5em Minus 0.4em. INTECH Open Access Publisher.

Hamadouche, M., Zebbiche, K., Guerroumi, M., Tebbi, H., Zafoune, Y., 2021. A comparative study of perceptual hashing algorithms: application on fingerprint images. In: 2nd International Conference on Computer Science's Complex Systems and Their Application. Algeria, p. 12. May.

Hijazi, S., Kumar, R., Rowen, C., et al., 2015. Using Convolutional Neural Networks for Image Recognition, vol. 9. Cadence Design Systems Inc., San Jose, CA, USA, 1.

Internet Watch Foundation, 2024. What Has Changed in the AI CSAM Landscape? Jul, [Online]. Available: https://www.iwf.org.uk/media/nadlcb1z/iwf-ai-csam-report_update-public-jul24v13.pdf.

N. Krawetz. Photodna and limitations. Not known. [Online]. Available: https://www.hackerfactor.com/blog/index.php?archives/931-PhotoDNA-and-Limitations.html.

Manjunath, B.S., Ohm, J.-R., Vasudevan, V.V., Yamada, A., 2001. Color and texture descriptors. IEEE Trans. Circ. Syst. Video Technol. 11 (6), 703–715.

Manning, C.D., Raghavan, P., Schütze, H., 2008. Introduction to Information Retrieval. Plus 0.5em Minus 0.4em. Cambridge University Press, New York.

McKeown, S., Buchanan, W.J., 2023. Hamming distributions of popular perceptual hashing techniques. Forensic Sci. Int.: Digit. Invest. 44, 301509.

McKeown, S., Russell, G., Leimich, P., 2019. Fast forensic triage using centralised thumbnail caches on windows operating systems. Journal of Digital Forensics, Security and Law 14 (3).

McKeown, S., Aaby, P., Steyven, A., 2024. PHASER: perceptual hashing algorithms evaluation and results-an open source forensic framework. In: Forensic Science International: Digital Investigation, vol. 48. Elsevier, 301680 publisher.

MIRFLICKR Download. Jun, [Online]. Available: http://press.liacs.nl/mirflickr/mirdownload.html.

Singh, A., Gupta, S., 2022. Learning to hash: a comprehensive survey of deep learning-based hashing methods. Knowl. Inf. Syst. 64 (10), 2565–2597.

Steinebach, M., 2011. Robust hashing for efficient forensic analysis of image sets. In: International Conference on Digital Forensics and Cyber Crime. Plus 0.5em Minus 0.4em. Springer, pp. 180–187.

Steinebach, M., 2012. Robust hashing for efficient forensic analysis of image sets. In: Gladyshev, P., Rogers, M.K. (Eds.), Digital Forensics and Cyber Crime, vol. 88. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 180–187 plus 0.5em minus 0.4em, series Title: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.

Struppek, L., Hintersdorf, D., Neider, D., Kersting, K., 2022. Learning to break deep perceptual hashing: the use case NeuralHash. In: 2022 ACM Conference on Fairness, Accountability, and Transparency, pp. 58–69.

Swain, M.J., Ballard, D.H., 1991. Color indexing. Int. J. Comput. Vis. 7 (1), 11–32.

Thiel, D., Stroebel, M., Portnoff, R., 2023. Generative Ml and Csam: Implications and Mitigations.

Tyagi, V., 2017. *Content-Based Image Retrieval*. Plus 0.5em Minus 0.4em. Springer Singapore, Singapore.

Venkatesan, R., Koon, S.-M., Jakubowski, M.H., Moulin, P., 2000. Robust image hashing. In: Image Processing, 2000. Proceedings. 2000 International Conference on, vol. 3. IEEE, pp. 664–666 plus 0.5em minus 0.4em.

Zauner, C., 2010. Implementation and Benchmarking of Perceptual Image Hash Functions.