




# High Rank Self-Organising Maps for Image Fingerprinting

Anthony Benjamin Kolenic and Duncan Anthony Coulter<sup>(✉)</sup> 

University of Johannesburg, Corner of Kingsway and University Roads,  
Auckland Park, Johannesburg, South Africa  
dcoulter@uj.ac.za

**Abstract.** Image fingerprinting is the act of generating a unique digest for an image. Unlike cryptographic hashing, slight differences in the input to the hashing function do not create significant differences in the digest. This property makes image fingerprinting useful in identifying near-duplicates of an input image. This paper describes a novel technique for generating an image fingerprint using Self-Organising Maps (SOM) with ranks higher than 2. The method is compared to a selection of more traditional fingerprinting algorithms and against a further variation on the proposed technique using a more conventional rank 2 Self-Organising Map.

**Keywords:** Self-Organising Map · Image fingerprinting · Duplicate detection · Artificial intelligence

## 1 Introduction

The fingerprinting of an image is a valuable way to identify similar and duplicate images within a set of images. Therefore, fingerprinting can be used as a tool with which we can classify and group unknown images that are not yet within the set. While it is tempting to equate hashing to fingerprinting, they are two different techniques. The critical difference is that minor differences in the input in the hashing input lead to significant changes in the resulting digest. In contrast, in fingerprinting, small changes in the input lead to small changes in the resulting digest [3]. Although both techniques produce smaller digests than the original data in the image.

Rephrasing the generation process as a noise-resistant way to generate a unique digest with lower dimensionality than the original input helps identify where Self Organising Maps (SOMs) can be helpful. Self Organising Maps are commonly used for dimensionality reduction to aid with the visualization of large data sets and how the data within large data sets interact. SOMs achieve this by providing easy to consume visuals such as u-matrices and heat maps. Another critical aspect of a SOM is that it clusters similar data together, thus providing the second property required for a fingerprinting algorithm: Noise and rotation resilience [6].

This paper will provide background into Self Organising Maps and explore the applications of higher rank Self Organising Maps. Then, building on the background, the concept of using SOMs for the fingerprinting of images will be outlined. After this, the technique will be compared to pre-existing Self Organising Map libraries and tested against well-known data sets. Finally, recommendations will be made on how the technique can be improved and the direction of possible future work. This technique aims to provide a novel manner in which an image can be fingerprinted using SOMs.

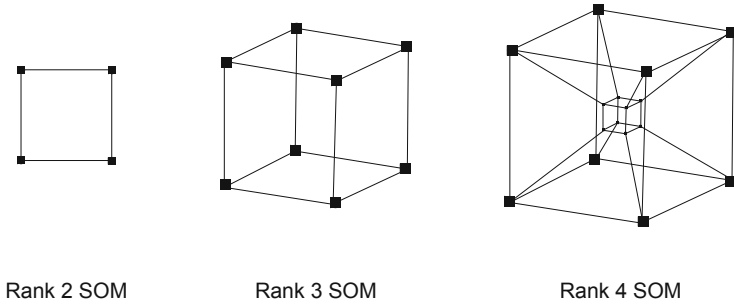
## 2 Self Organising Maps

Self Organising Maps (SOMs) are a form of artificial neural network (ANN) created by Dr Teuvo Kohonen. Self Organising Maps are also known as Kohonen Maps for this reason. While SOMs are a form of ANN, they have noticeable differences in their learning process, structure, and training method as opposed to Feed-Forward ANNs and Multi-Layer Perceptron ANNs. Whilst most ANNs are trained using corrective learning, SOMs take the approach of competitive learning. Competitive learning is utilised as it is better suited for unsupervised tasks, which SOMs are traditionally used for [5]. Structurally, the nodes of an ANN are represented by an activation function that changes output based on the input provided, whereas a SOM is a lattice of nodes that cluster around features. Each of these nodes contains a vector which is adjusted through the training process. The training process is briefly discussed in the next paragraph and in more detail in the method section. It is essential, at this stage, to take note that while the nodes are usually represented on a 2D grid lattice, the dimensionality of the lattice can be increased [4]. To visualise how one would represent a SOM on a higher dimension lattice, refer to Fig. 1.

As previously mentioned, SOMs are trained using a competitive learning approach. While this approach is discussed in more detail in the method section, this paragraph will provide a high-level overview of the basic training and operation of a Self Organising Map. The first step is to determine the initial parameters of a SOM. These parameters include the length of each node's vector and the dimensions of the SOM. The length of each node's vector can be thought of as the weights for each node. They are initialised to random values before training can begin. Since SOMs are represented as a lattice, each node has neighbours that form its neighbourhood, where the neighbourhood is traditionally defined with a neighbourhood function such as Moore's neighbourhood that has been extended for Chebyshev distances greater than 1. Determining the dimension of a SOM is also important and will determine how many nodes there are in the SOM.

Once all the nodes have been initialised with random values, training can commence. Training is the process where a random item is selected from the input data and presented in vector form to the self-organising map to find the node in the SOM whose vector has the smallest Euclidean distance to that of the input vector. Using this vector, known as the Best Matching Unit (BMU),

all the nodes in the neighbourhood of the BMU are updated by pulling their vectors closer to the BMU in regard to their vector’s Euclidean distance to the BMU. This process is repeated for a set number of iterations. Once the SOM has converged, an image is created. This image is known as a u-matrix and illustrates the average distance of a node to its adjacent nodes.



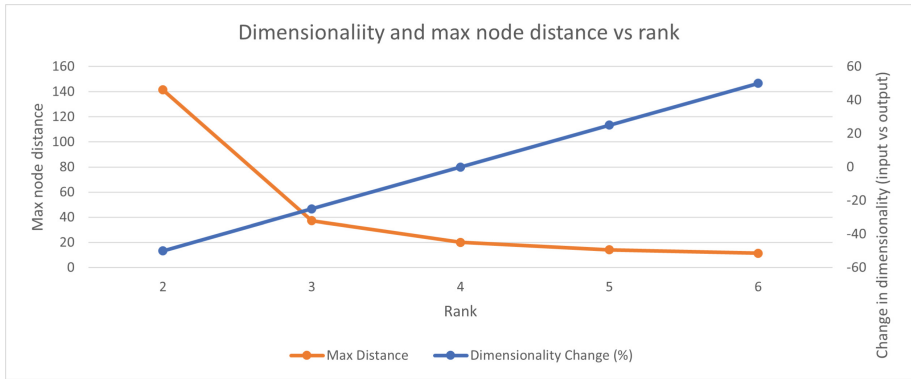
**Fig. 1.** Effect of increasing rank on a Self-Organising Map

The above sections mainly deal with self-organising maps in 2 spatial dimensions but allude to the fact that SOMs can be extended to work in higher spatial dimensions. This next section will discuss the properties of SOMs in higher spatial dimensions and the considerations that need to be kept in mind when extending SOMs [8]. When working with SOMs, the different interpretations of the word “dimension” can lead to confusion. To prevent this confusion, the paper will use the word rank (An abstract description of a vector) to describe the manipulation of a SOM in higher spatial dimensions. When increasing the rank of the SOM, there are important properties to note.

Interestingly, the dimensionality reduction is inversely proportional to the rank. Another property is that the distance between nodes decreases when keeping the number of nodes constant and increasing the rank. Shown below in Fig. 2 is an example of these properties applied to the IRIS data set, which has four attributes per value:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm

Increasing the rank of a SOM does come with unique challenges that need to be overcome. Firstly the creation of easy to view and understandable outputs, and secondly, the shapes that allow for the generation of regular honeycombs.



**Fig. 2.** Relationship between the rank of a SOM and the reduction in dimensionality of the data as well as the decrease of the maximum distance between the indices of the nodes as the rank increases and node count remains constant. The node count being 10,000 in this case

A SOMs primary use case allows the visualization and clustering of data that cannot be understood easily. The difficulty of understanding the data arises from problems such as large quantities of data or high dimensionality of the data. When increasing the rank of the SOM, it loses its ability to make data with a high dimension easy to understand visually. Although, colours and shapes can be used over an interval to minimize this impact. However, increasing the rank is not advised when working with data that has to be viewed by a human. This rule of thumb is reinforced by the fact that while it is easy to visualize a square or cube, it is difficult to visualize its four dimension analogue, the tesseract and nearly impossible to visualize a 5D hypercube, especially on a 2D surface such as a monitor or piece of paper.

A further technical problem comes with the structure of a Self-Organizing Map. SOMs can be thought of as a mathematical regular honeycomb. A regular honeycomb is the tiling/tessellation of regular convex polygons/polyhedrons so that there are no gaps and overlap. With rank two, the nodes can be thought of as the vertices in an equilateral triangle, a square or a hexagon where each node would have 3, 4, or 6 direct neighbours, respectively. Rank 3 only has one regular honeycomb, the cubic honeycomb, where each node would have six direct neighbours. From rank five onwards, it means the only shape each node can take is that of a hypercube [2]. This fact is important as it directly speaks to why the paper uses squares and their higher-dimensional analogues as the basis for the shape of the SOM.

### 3 Method

Presented below is a method for fingerprinting a set of images for quickly identifying duplicates and potentially classifying images. The technique involves two main processes, namely, the training of a SOM to generate fingerprints and the testing of fingerprints to identify near similarities.

#### 3.1 Fingerprint Generation

The creation of an image's fingerprint is done using a Self Organising Map. Before one can train the self-organising map, the input images must first be segmented. These images represented as  $N \times N$  array of numbers between 0 and 1 are segmented into  $M \times M$  blocks for processing within the SOM, where  $N \bmod M = 0$  and  $N, M \neq 0$ . The segmenting of the input images into the same size blocks helps cluster images that have similarities in certain areas, like ID photos on a white background. The dimension of the vectors, also known as the weights, within the SOM will equal  $M^2$ . Pseudocode for this algorithm is provided in Algorithm 1.

```

input : An image of size  $N \times N$ 
input : The segment size represented by  $M$ 
output: A list of  $(\frac{N}{M})^2$  segments

1 if  $N \bmod M = 0$  then
2    $results \leftarrow []$ ;
3    $numSegments \leftarrow \frac{N}{M}$ ;
4   for  $r$  in  $numSegments$  do
5     for  $c$  in  $numSegments$  do
6        $x \leftarrow c \times M$ ;
7        $y \leftarrow r \times M$ ;
8       append image segment at point  $(x, y, x + M, y + M)$  to  $results$ ;
9     end
10  end
11  return  $result$ ;
12 else
13   display error and exit
14 end

```

**Algorithm 1:** Input pre-processing

Training of the SOM is performed by firstly initializing the  $P^r$  SOM where  $r$  is the rank of the SOM, and  $P$  is the side length. In the SOM, each node, which is represented by a vector of length  $M^2$ , is initialized with random values. Once the SOM has been initialized, the training can commence by performing a set of instructions  $n$  number of times. A random image segment is selected from the input list and flattened into a  $1 \times M^2$  vector on each training iteration. This vector

is fed into the SOM, where it is presented to each node within the map. This input process allows the SOM to identify which node is most similar to the input vector. Determining the similarity is done via a similarity metric such as the  $L^2$  norm between the input vector and the weight of a node. Once the most similar node has been identified, the node's neighbours can be adjusted to cluster around the node. The reach and proportion of clustering is reduced as more iterations are performed. While the training remains the same with increasing rank, as mentioned before, the number of neighbours to a node increases. Pseudocode for this algorithm can be found in Algorithm 2.

```

input : The number of training iterations represented by  $n$ 
input : The side length of the SOM represented by  $P$ 
input : The rank of the SOM represented by  $r$ 
input : A list of training segments  $segmentList$ 
output: A trained self organising map

1  $result \leftarrow$  randomly instantiate SOM of rank  $r$ ;
2 for  $iteration$  in  $n$  do
3    $sample \leftarrow$  selectRandomSegment( $segmentList$ );
4    $bmuIndex \leftarrow$  getIndexOfClosestNode( $result$ ,  $sample$ );
5    $range \leftarrow maxRange \times \frac{iteration}{n}$ ;
6    $learningRate \leftarrow maxLearningRate \times \frac{iteration}{n}$ ;
7   for  $node$  in  $result$  where distance between node and sample < range do
8      $node \leftarrow node + (sample - node) \times learningRate$ ;
9   end
10 end
11 return result

```

**Algorithm 2:** Self-Organising Map Training

After the SOM has converged, the creation of the fingerprints can be performed. The creation of the fingerprints is done by presenting all of the input data in order to the SOM and storing the index of the most similar weight into a list which is then flattened. Looking at a rank 2 SOM where an input image of size  $4 \times 4$  is broken into segments of size 2 ( $N = 4$  and  $M = 2$ ), the result is four segments that, once flattened, have a size equal to  $M^2$ . If each segment is presented to the SOM because it is of rank two, the resulting index, the closest nodes index, will have two components. These components are appended to a list creating a set that contains  $\{x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4\}$ , where each  $x$  and  $y$  value correspond to an index in the SOM. The length of the digest would be equivalent to the number of segments multiplied by the rank of the SOM and represents the fingerprint. The length of this fingerprint is represented by  $k$ . Pseudocode is provided by Algorithm 3:

**input** : A *inputlist* of length  $l$  that contains image segments represented by vectors of length  $M^2$   
**input** : A *SOM* of rank  $r$  that is pre-trained, where the node weights are of length  $M^2$   
**output**: A vector of length  $l \times r$

```

1 result  $\leftarrow$  [];
2 for chunk in inputlist do
3   | bmuIndex  $\leftarrow$  getIndexOfClosestNode();
4   | for index in bmuIndex do
5   |   | append index to result
6   | end
7 end

```

**Algorithm 3:** Fingerprint generation

### 3.2 Fingerprint Comparison

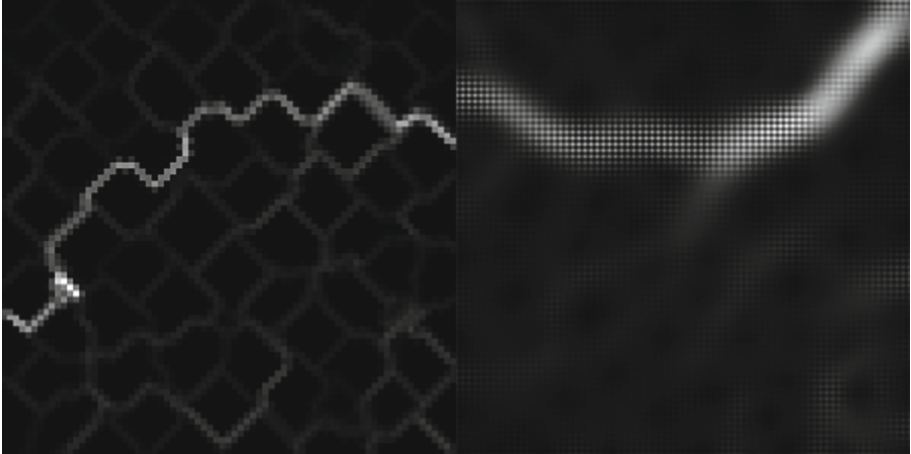
Comparing generated fingerprints of the training data and input data is straightforward. First, the input image is converted into a grid of segments as outlined above. Then, each input segment is presented to the SOM, and a fingerprint for the input is generated. A vector of length  $k$  represents this fingerprint. This vector can then be compared to the list of previous vectors to determine which fingerprint is the most similar. In this paper, multiple techniques are used, and the results are presented below. The techniques used to find similar fingerprints are distance metrics such as the Euclidean distance, Cosine distance, Manhattan distance, as well as  $k$ -nearest neighbours.

## 4 Results

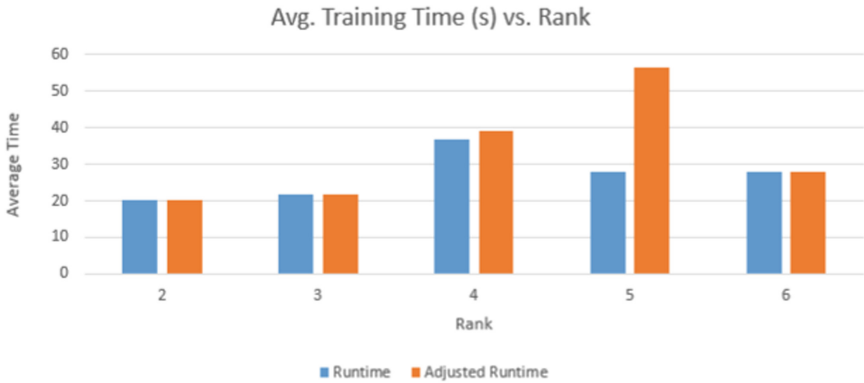
The initial step of the creation of this novel technique was to implement a standard SOM and compare it to a well-established framework such as SuSi [7] to ensure the underlying SOM is behaving as expected. Below is a comparison of a rank 2 SOM, the custom implementation and using the SuSi framework. The IRIS dataset was chosen for these comparisons as it is a well-understood dataset, and the  $u$ -matrix is distinct. Figure 3 compares the custom implementation to SuSi. This comparison demonstrates that the  $u$ -matrices are similar between the implementations. It also shows a clear divide in the dataset, proving that the SOM is behaving as expected. Following the previous comparison, another comparison is performed with the technique against itself to compare the training times at different ranks.

The times were taken as the average training time between 5 runs for rank 2–5 using the IRIS dataset, the number of nodes within the SOM is  $n^r \approx 15625$  where  $r$  is the rank of the SOM (Fig. 4). To account for approximating the nodes, the training time can be adjusted using the formula  $r \times (\text{actual node count} \div \text{desired node count})$  where  $r$  is the rank. The result of this adjustment can also

be seen in Fig. 4. Both comparisons use an epoch of 10000 for the number of training iterations and Euclidean distance for the distance metric. The number of nodes was chosen as 15625 as roots between 2 and 5 for 15625 have minimal rounding errors. Finally, there is a subsection that deals with the MNIST section.



**Fig. 3.** Comparison between u-matrices generated on the IRIS dataset by the custom and SuSi implementations respectively.

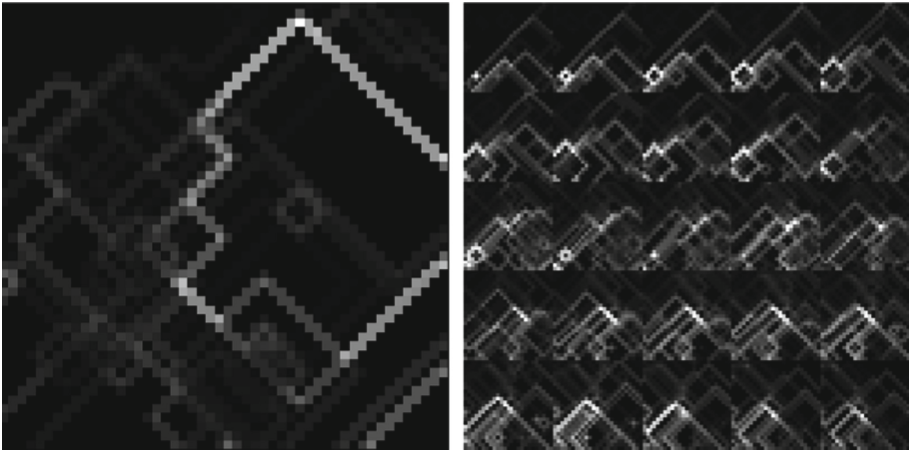


**Fig. 4.** The relation between a Self-Organising Map’s rank and its training time measured in seconds



### 4.1 MNIST

The MNIST dataset was chosen as the initial test data for the fingerprint generation. MNIST was chosen as it is a well studied, freely available image dataset that is in greyscale, and the training and testing data is already split. Figure 5 shows the u-matrix of the Self-Organising Map when trained on a rank 2 SOM and a rank 3 SOM where the u-matrix is taken as slices of the SOM. Table 1 that follows shows the average training times, image fingerprinting error rate, and accuracy of image recognition using different distance metrics with respect to different sized SOM with different ranks.



**Fig. 5.** MNIST u-matrix for rank 2 and 3 Self-Organising map

**Table 1.** Table showing Self-Organising Maps and their respective statistics. Time measured in seconds

Rank	2	2	2	3	3	3	4	4
Size	20	30	40	20	30	40	20	30
Node count	400	900	1600	8000	27000	64000	160000	810000
Training time	10.95	24.26	50.04	345.52	1068.92	2722.0	7239.08	37017.87
Fingerprinting time	687.89	1065.8	873.53	1342.13	3128.96	3364.83	6803.05	11863.93
Recognition time	524.53	349.15	572.33	924.13	1359.02	1863.06	2864.44	6634.27
Fingerprint error	18%	19%	17%	15%	16%	18%	27%	23%
Euclidean accuracy	65.20	65.00	62.70	70.10	64.00	64.30	62.70	62.40
Hamming accuracy	72.20	71.00	72.50	74.00	69.30	70.70	65.00	68.20
Manhattan accuracy	70.20	68.20	66.00	75.10	66.60	67.80	64.30	64.60

Interpreting the results from the table above, it is evident that the algorithm does better than randomly guessing if the images are the same. The accuracy also increases with training time but struggles to outpace the exponential growth of the node count. Thus, the best performing set of parameters was a SOM of rank 3 with a size of 20. Using this variable composition leads to a modest training time with better accuracy than other configurations. The following table (Table 2) compares the defined technique to modern image hashing methods. The algorithms being compared are average, perceptual, difference, and wavelet hashing. Looking at the values in the table, there is potential in using SOMs for image fingerprinting. This potential is evident in image recognition, where two images are similar, but the traditional hashes are too dissimilar.

**Table 2.** Table showing the accuracy and error of current image fingerprinting techniques

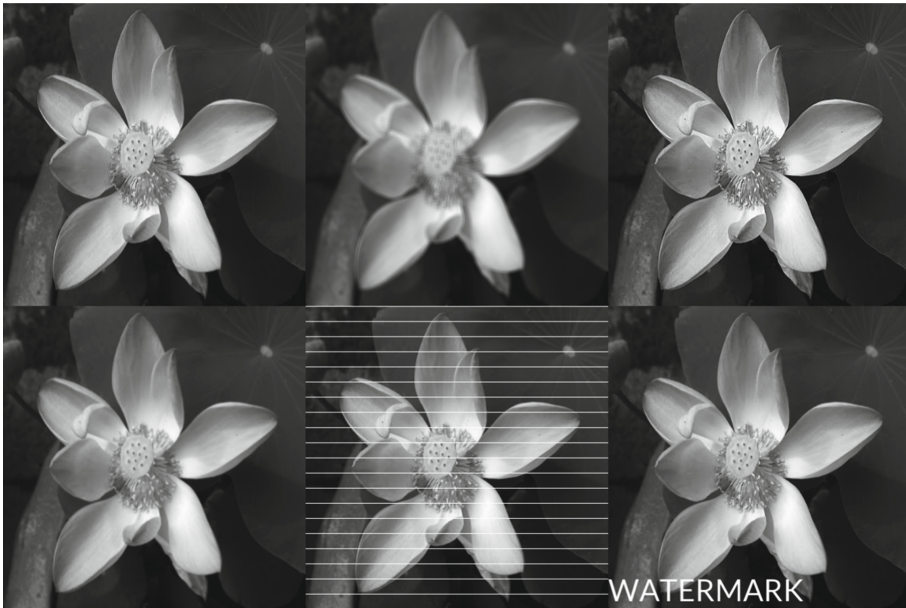
Technique	Fingerprint error	Recognition accuracy
Average hashing	0.15%	83.6%
Perceptual hashing	0.00%	81.5%
Difference hashing	0.00%	79.1%
Wavelet hashing	0.55%	62.0%

## 4.2 Noise Resilience

When fingerprinting, the technique must be noise resilient, as detail can be lost through compression, resizing, or watermarks. To demonstrate this technique's noise resilience, a set of 100 images that are the same size are used as input. Their fingerprints are compared to the same images but with noise applied in the form of loss of detail through compression and resizing, blurring, sharpening, changed pixel intensities or watermarks. Figure 6 demonstrates the different noise techniques and is followed by Table 3 with performance metrics.

**Table 3.** Effect of noise to image fingerprinting

	Blur	Sharpen	Smooth	Distorted	Watermark
Correct	93	98	96	95	77
Incorrect	7	2	4	5	23
Accuracy	93%	98%	96%	95%	77%



**Fig. 6.** Various noise filters that can be applied

## 5 Conclusion

In conclusion, this is a feasible technique for the fingerprinting of images. The noise-resilience of the technique ensures it would work with slightly altered images. Examples of alterations are watermarks or loss of detail through resizing and compression. While it can also perform image recognition, more research is required to improve the accuracy of the technique so that it can be competitive against techniques such as convolutional neural networks (one of the more commonly employed ANN architectures in the domain of image recognition problems) [1]. The main drawback of the technique is that training the SOM requires data similar to the images that will be fingerprinted, and training can be a time-consuming process. Future work on this topic could lead to new methods that decrease run times of the technique as well as increase fingerprinting accuracy. The applications of future work on this technique could also lead to improved image recognition and malware detection.

## References

1. Bhandare, A., et al.: Applications of convolutional neural networks. *Int. J. Comput. Sci. Inf. Technol.* **7**, 2206–2215 (2016). ISSN 0975-9646. <https://ijcsit.com/docs/Volume%207/vol7issue5/ijcsit20160705014.pdf>
2. Coxeter, H.S.M.: *Regular Polytopes*, 3rd edn., pp. 58–73. Dover Publication Inc., New York (1973). 292296

3. Du, L., Ho, A.T.S., Cong, R.: Perceptual hashing for image authentication: a survey. *Sig. Process. Image Commun.* **81**, 115713 (2020). ISSN 0923-5965. <https://doi.org/10.1016/j.image.2019.115713>. <http://www.sciencedirect.com/science/article/pii/S0923596519301286>
4. Kohonen, T.: The basic SOM. In: *Self-Organizing Maps*, pp. 105–176. Springer, Heidelberg (2001). ISBN 978-3-642-56927-2. [https://doi.org/10.1007/978-3-642-56927-2\\_3](https://doi.org/10.1007/978-3-642-56927-2_3)
5. Kohonen, T.: Variants of SOM. In: *Self-Organizing Maps*, pp. 191–243. Springer, Heidelberg (2001). ISBN 978-3-642-56927-2. [https://doi.org/10.1007/978-3-642-56927-2\\_5](https://doi.org/10.1007/978-3-642-56927-2_5)
6. Polsterer, K.L., Gieseke, F., Doser, B.: PINK: parallelized rotation and flipping INvariant Kohonen maps (October 2019). ascl: 1910.001
7. Riese, F.M., Keller, S., Hinz, S.: Supervised and semi-supervised self-organizing maps for regression and classification focusing on hyperspectral data. *Remote Sens.* **12**(1), 7 (2019). rs12010007. <https://doi.org/10.3390/rs12010007>
8. Seiffert, U., Michaelis, B.: Multi-dimensional self-organizing maps on massively parallel hardware. In: *Advances in Self-Organising Maps*. Springer, London (2001). [https://doi.org/10.1007/978-1-4471-0715-6\\_23](https://doi.org/10.1007/978-1-4471-0715-6_23)