University of Wollongong

Research Online

Faculty of Engineering and Information
Sciences - Papers: Part B

Faculty of Engineering and Information
Sciences

2021

Utilizing QR codes to verify the visual fidelity of image datasets for machine learning

Yang-Wai Chow

University of Wollongong, caseyc@uow.edu.au

Willy Susilo

University of Wollongong, wsusilo@uow.edu.au

Jianfang Wang

Richard Buckland

Joon Sang Baek

University of Wollongong, baek@uow.edu.au

See next page for additional authors

Follow this and additional works at: <https://ro.uow.edu.au/eispapers1>

Part of the [Engineering Commons](#), and the [Science and Technology Studies Commons](#)

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Utilizing QR codes to verify the visual fidelity of image datasets for machine learning

Abstract

Machine learning is becoming increasingly popular in modern technology and has been adopted in various application areas. However, researchers have demonstrated that machine learning models are vulnerable to adversarial examples in their inputs, which has given rise to a field of research known as adversarial machine learning. Potential adversarial attacks include methods of poisoning datasets by perturbing input samples to mislead machine learning models into producing undesirable results. While such perturbations are often subtle and imperceptible from the perspective of a human, they can greatly affect the performance of machine learning models. This paper presents two methods of verifying the visual fidelity of image-based datasets by using QR codes to detect perturbations in the data. In the first method, a verification string is stored for each image in a dataset. These verification strings can be used to determine whether or not an image in the dataset has been perturbed. In the second method, only a single verification string is stored and can be used to verify whether an entire dataset is intact.

Keywords

codes, learning, machine, datasets, image, fidelity, verify, utilizing, visual, qr

Disciplines

Engineering | Science and Technology Studies

Publication Details

Chow, Y., Susilo, W., Wang, J., Buckland, R., Baek, J., Kim, j. & Li, N. (2021). Utilizing QR codes to verify the visual fidelity of image datasets for machine learning. *Journal of Network and Computer Applications*, 173 102834-1-102834-11.

Authors

Yang-Wai Chow, Willy Susilo, Jianfang Wang, Richard Buckland, Joon Sang Baek, jongkil Kim, and Nan Li

Utilizing QR Codes to Verify the Visual Fidelity of Image Datasets for Machine Learning

Yang-Wai Chow^a, Willy Susilo^{a,*}, Jianfeng Wang^b, Richard Buckland^c,
Joonsang Baek^a, Jongkil Kim^a, Nan Li^d

^a*Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Australia*

^b*State Key Laboratory of Integrated Service Networks (ISN), Xidian University, China*

^c*School of Computer Science and Engineering, University of New South Wales, Australia*

^d*School of Electrical Engineering and Computing, University of Newcastle, Australia*

Abstract

Machine learning is becoming increasingly popular in modern technology and has been adopted in various application areas. However, researchers have demonstrated that machine learning models are vulnerable to adversarial examples in their inputs, which has given rise to a field of research known as adversarial machine learning. Potential adversarial attacks include methods of poisoning datasets by perturbing input samples to mislead machine learning models into producing undesirable results. While such perturbations are often subtle and imperceptible from the perspective of a human, they can greatly affect the performance of machine learning models. This paper presents two methods of verifying the visual fidelity of image-based datasets by using QR codes to detect perturbations in the data. In the first method, a verification string is stored for each image in a dataset. These verification strings can be used to determine whether or not an image in the dataset has been perturbed. In the second method, only a single verification string is stored and can be used to verify whether an entire dataset is intact.

Keywords: Adversarial machine learning, cyber security, QR code, visual

*Corresponding author

Email addresses: caseyc@uow.edu.au (Yang-Wai Chow), wsusilo@uow.edu.au (Willy Susilo), jfwang@xidian.edu.cn (Jianfeng Wang), richardb@unsw.edu.au (Richard Buckland), baek@uow.edu.au (Joonsang Baek), jongkil@uow.edu.au (Jongkil Kim), nan.li@newcastle.edu.au (Nan Li)

1. Introduction

The widespread popularity of Machine Learning (ML) in modern technology and its rapid advancements in recent years have paved the way for its adoption in a variety of application areas. ML techniques empower a range of diverse applications, including malware detection [1], self-driving cars, network intrusion detection, natural language processing, behavioral analytics, speech recognition, and so on. Nevertheless, researchers have demonstrated that ML models are vulnerable to adversarial examples in its inputs. The purpose of these adversarial techniques is to employ malicious inputs to fool a ML model into producing erroneous outputs [2, 3]. Consequently, this has given rise to a field of research known as adversarial machine learning [4].

There are a number of different adversarial attacks that can be deployed against ML techniques, for instance, an adversary can poison a dataset by perturbing samples in the training data [4]. Over the years, researchers have examined and demonstrated the effectiveness of such poisoning attacks [5, 6, 7]. Adversarial attacks are a serious threat to the success of ML in practice, as small and subtle perturbations in its inputs can mislead a ML model into outputting incorrect predictions, thereby negating the usefulness of the ML model. For ML in computer vision, such perturbations made to images in the ML dataset are often imperceptible to the human visual system [8].

This paper focuses on methods of protecting image-based datasets by verifying the visual fidelity of the data. The effectiveness of ML techniques is affected by the availability of training data. In general, ML models require large amounts of training data to be effective. In light of this necessity for significant amounts of training data, many ML models are trained using public datasets that are freely available online. These public datasets are often copied and distributed without any mechanism for protecting the integrity of the data. This makes these datasets vulnerable to alterations by an adversary.

To address this problem, the research presented here investigates two methods of verifying the visual fidelity of image-based datasets by detecting perturbations in the data using QR codes. The advantage of the proposed methods is that a copy of the original dataset does not have to be stored or

used for verification. In the first method, a verification string is generated for each image in a dataset using a QR code. The size of a verification string is much smaller than the original image, and it can be used to verify the visual fidelity of the image. To verify image fidelity, a verification process is used to recover a QR code for each image. If the resulting QR code is noisy or cannot be recovered through this verification process, this indicates that the dataset has been altered.

However, since this method requires a verification string for each image in a dataset, the storage requirement increases linearly with the number of images in the dataset. While this is fine if storage space is not an issue, it may not be an attractive solution for applications with limited storage capacity, e.g., for mobile computing and pervasive social networking that may have storage restrictions [9]. Therefore, a second method is proposed where only a single verification string is required, and can be used to verify the fidelity of images in an entire dataset. The limitation of the second method is that if the dataset has been altered, one cannot determine which image has been altered, only that the dataset is not intact as it is different from the original.

1.1. Contribution.

The research presented in this paper investigates the problem of protecting image-based ML datasets against alteration by an adversary. The proposed methods, which were first presented in Chow et al. [10], attempt to provide mechanisms for verifying the visual fidelity of images in a dataset without the need to store and use the original dataset for this purpose. To do this, verification strings are generated from the visually important content of the images and these are associated with QR codes. The reason for using QR codes for this purpose is due to its inherent data capacity and error correction properties, which are inbuilt in the QR code structure. Two methods for creating verification strings to verify the fidelity of an image dataset are presented. The advantage of the first method, i.e. the Linear Verification String (LVS) method, is that the fidelity of each image can be verified individually. However, this comes at the cost of higher storage requirements. The advantage of the second method, i.e. the Aggregate Verification String (AVS) method, is that only a single verification string is required to determine whether an entire dataset is intact. Nevertheless, it does not allow for one to determine the visual fidelity of individual images.

2. Background

The proposed methods are based on several concepts, including the QR code structure, the Discrete Wavelet Transform (DWT), the Arnold transform and trapdoor permutation. This section presents a background to these concepts followed by a description of related work.

2.1. Preliminaries

2.1.1. QR Code

The Quick Response (QR) code is a two-dimensional (2D) barcode, which was invented by the company Denso Wave [11]. Since its inception, the QR code has seen widespread adoption in a variety of diverse application domains, ranging from electronic payments to product advertisements, social media, and so on. In this research, the reason for using the QR code is to exploit its inbuilt error correction mechanism, which is an inherent part of the QR code structure. This mechanism enables a QR code to be correctly decoded even if part of the QR code symbol is corrupted.

A QR code is made up of light and dark modules, which are organized into function patterns and an encoding region [12]. The size and data capacity of a QR code is determined by its version and error correction level. There are forty different QR code versions and four error correction levels, namely, low (L), medium (M), quartile (Q) and high (H). These four error correction levels correspond to error tolerances of approximately 7%, 15%, 25% and 30%, respectively. The higher the error correction level, the greater the capacity for data recovery. However, conversely the higher the error correction level, the larger the amount of error correction redundancy that must be encoded in the QR code. Hence, QR codes with higher error correction levels have lower capacity for encoding actual data as compared with the same QR code version with a low error correction level.

2.1.2. DWT

The Discrete Wavelet Transform (DWT) is a frequency domain transform technique that is widely used in signal processing. For 2D images, it involves decomposing an image into frequency channels of constant bandwidth on a logarithmic scale [13]. An image is decomposed into four sub-bands, which are labeled low-low (LL), low-high (LH), high-low (HL) and high-high (HH). The sub-bands can be further decomposed at the next level, and this process can continue until the desired number of levels is achieved.

LL_2	HL_2	HL_1
LH_2	HH_2	
LH_1		HH_1

Figure 1: Sub-bands of a two-level DWT.

A depiction of how an image can be decomposed into two levels of DWT sub-bands is shown in Fig. 1. The LL sub-band contains most of the information of the original image [14]. As such, it represents the highest visual fidelity of an image, as the human visual system is more sensitive to its contents. Data from the LL_2 sub-band was used in experiments conducted in this study.

2.1.3. Arnold transform.

Adjacent pixels in an image have strong correlation to each other. The Arnold transform, shown in Eq. 1, is an invertible transform that can be used to disrupt the correlation between adjacent pixels [15]. An original image that undergoes a number of Arnold transform iterations results in a chaotic image. The transform is invertible as the original image can be recovered after a certain number of transform iterations are applied to the chaotic image. The reason for using this transform in this study is because image perturbations introduce noise to an image. Applying the Arnold transform to an image scrambles the pixels, and thus, scatters the noise over the image. This increases the potential of detecting perturbations made to the image and of being able to recover the QR code despite the perturbations.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \mod N \quad (1)$$

2.1.4. Trapdoor permutation.

Let D be a finite set. A permutation family Π over D specifies a randomized algorithm for generating (descriptions of) a permutation and its inverse, denoted as $(s, t) \xleftarrow{R} \text{Generate}$; an evaluation algorithm $\text{Evaluate}(s, \cdot)$; and an inversion algorithm $\text{Invert}(t, \cdot)$. We require that for all (s, t) produced by the algorithm Generate , $\text{Evaluate}(s, \cdot)$ be a permutation of D and

$Invert(t, Evaluate(s, \cdot))$ be the identity map. A trapdoor permutation family is one way if it is hard to invert, given just the forward permutation description s . Formally, a trapdoor permutation family is (t, ϵ) -one way if no t -time algorithm $\mathcal{A}$ has advantage greater than ϵ . Hence,

$$\text{Adv Invert}_{\mathcal{A}} \stackrel{\text{def}}{=} \Pr \left[x = \mathcal{A}(s, Evaluate(s, x)) : (s, t) \xleftarrow{R} \text{Generate}, x \xleftarrow{R} D \right],$$

where the probability is over the coin tosses of *Generate* and $\mathcal{A}$.

2.2. Related Work

Researchers have previously proposed the use of QR codes for a number of diverse applications in information security. Examples of information security techniques that adopt the use of QR codes, include methods for authentication in mobile cloud computing [16], authentication for wearable devices [17], storing private and public information [18], secret sharing [19], visual cryptography [20], digital image watermarking [21] and so on.

The methods presented in this research are related to zero-based image watermarking. Unlike traditional image watermarking techniques, which necessitate that a watermark be embedded within an image, zero-based approaches do not require the embedding of a watermark [22]. For example, Liu and Yan [14] proposed a secret sharing scheme based on zero-based watermarking. In their approach, the watermark was not embedded within the images, but rather cover images were used to create shares that were to be distributed to participants, with another share that was to be registered with a certification authority. Ownership of the images could then be verified when information in the shares were combined with contents in the original images.

Zero-based image watermarking schemes that adopt the use of QR codes have also previously been proposed by the research community. As an example, an authentication method for medical images using zero-based watermarking and QR codes has been investigated. In this scheme, a patient's identification details and a link to their data was encoded in the form of a QR code that served as the watermark [23]. Similarly, Li et al. [24] proposed a QR code based zero watermarking scheme in conjunction with visual cryptography for authenticating identification photos.

In addition, QR codes have previously been used in conjunction with traditional digital image watermarking. For instance, a watermarking scheme based on the combination of discrete cosine transform, QR codes and chaotic

theory was proposed by Kang et al. [25]. In other work, a digital rights management technique for protecting documents by repeatedly inserting a QR code into a DWT sub-band of a document was investigated [26]. Others have also proposed different QR code watermarking approaches, for example, an approach that incorporates an attack detection feature to detect malicious interference by an attacker [27], another approach that embeds QR code watermarks using a just noticeable difference model to increase imperceptibility [28], and a hybrid approach that uses the discrete cosine transform in conjunction with the discrete wavelet transform for QR code watermarking [21].

3. Proposed Methods

The two proposed methods are described in this section. The first method allows for the verification of each image in a dataset. It provides a mechanism whereby a user can check whether individual images in a dataset have been perturbed. However, the first method comes at a cost of storing a verification string for each image. This may not be an appealing solution in situations where storage space is an issue. As such, a second method is proposed with very low storage requirements and provides a mechanism to verify whether an entire dataset is intact. Nevertheless, using the second method, while a user can determine whether or not a dataset is intact, the user cannot determine which of the images have been perturbed in the event that the dataset has been altered from its original version.

3.1. Method 1 - Linear Verification String (LVS) method

The notion behind this method is to generate a verification string for each image in a dataset. Hence, it is called the *Linear Verification String (LVS)* method. The purpose of a verification string is to be able to ascertain whether an image has been altered from the original image. An advantage of generating a verification string is so that respective images in a dataset do not have to be compared with their original image. Moreover, the verification strings will require much less storage space when compared with the size of an entire image dataset. This is due to the fact that the size of a verification string, which is a bit string consisting of 0s and 1s, is much smaller than the size of an image.

The motivation behind this approach is based on zero-based image watermarking [14]. Unlike traditional watermarking techniques that requires a

Figure 2: Overview of the process to generate a verification string for each image in a dataset.

watermark to be embedded within an image, zero-based approaches do not alter the image. This is in line with the objective of the proposed approach, which is to detect whether images in a dataset have been altered. Hence, the method itself must not change the images.

3.1.1. Generating Verification Strings.

Fig. 2 gives an overview of the process used to generate verification strings for all images in a dataset. It can be seen from the figure that to create the verification strings, a QR code that contains a secret message, S , and a key, K , for encryption is required. K is a random bit string, which can be generated in a number of different ways, for example, by using a pseudo-random number generator or by using the hash of a password. Encryption involves performing $S \oplus K$ for all modules in S . As such, the length of K must equal the number of modules in S . For instance, a QR code version 1 which consists of 21×21 modules requires the key to contain 441 random bits.

The purpose of performing encryption is so that an adversary will not be able to obtain any information about S from the verification strings. Arnold transform will then be performed on the encrypted secret, S_E , to scramble the pixels. The reason for doing this is to scatter any noise that may result

from perturbations to an image over the entire image. The scrambled secret, S_T , will be used for creating the verification strings. Note from Fig. 2 that the same S_T can be used for all images, as this will avoid the necessity of having to generate multiple QR code messages and keys.

Each image in the dataset then undergoes the same process. Each image is decomposed into DWT components at the desired level (DWT at level 2 was used to obtain the experiment results presented in Section 4). The LL sub-band, which represents the highest visual fidelity, will be binarized using a dithering process. The objective of dithering is to binarize the image into black and white bits (i.e. 0s and 1s), while maintaining the average gray level distribution. In the results presented in Section 4, the Floyd-Steinberg dithering technique, which is an approach that is based on error diffusion [29], was used for the experiments.

After the dithering process, each image is upsampled using nearest neighbor sampling to match the size of S_T . The upsampling process is the reason why S can be of any size larger than I_{DWT} . Finally, a verification string, V_i , is produced for each image in the dataset by XORing S_T with their respective dithered and upsampled DWT sub-band.

For grayscale images, this process can be carried out directly. For color images, there are two options. The first is to convert the color images into grayscale images, prior to running the algorithm. The second is to separate the color channels, e.g., into red, green and blue color channels, and perform the process on each individual color channel, in which case the verification string’s length will increase based on the number of color channels. Nevertheless, the advantage of the second option of generating verification strings for each color channel allows for the method to detect alterations in individual color channels.

Details of the algorithm to generate verification strings for all images in a dataset is provided in Algorithm 1. Algorithm 2 in turn, modifies the original verification string generation algorithm by performing the process on separate color channels for color images. Inputs to both algorithms are a secret QR code, S , an encryption key, K , and all images in a dataset. The algorithm outputs a verification string for each image in the dataset.

3.1.2. Verifying Image Fidelity.

The verification strings can be used to verify the visual fidelity of images in the dataset, and to determine whether any of the images have been perturbed. An overview of this process is depicted in Fig. 3. Similar to the

Algorithm 1 Algorithm for generating verification strings.

Input: A QR code, S , a key, K , and images in a dataset, I_i , where $i \in \{1, 2, \dots, n\}$.

Output: Verification strings, V_i , where $i \in \{1, 2, \dots, n\}$

Step 1. Encrypt information in S by XORing the random bits in K with the modules in S to produce S_E .

Step 2. Generate a chaotic image S_T by scrambling the bits in S_E using Arnold transform for a number of iterations.

For each image, I_i , in the dataset, do

Step 3. Convert I_i to I_{DWTi} by performing DWT to the desired level.

Step 4. Extract the LL sub-band from I_{DWTi} .

Step 5. Dither the pixels to binarize the extracted LL sub-band into black and white bits (i.e. 0s and 1s).

Step 6. Produce I_{Di} by upsampling the dithered LL sub-band to match the size of S_T .

Step 7. Generate the verification string V_i using $I_{Di} \oplus S_T$.

process for generating verification strings, each image in the dataset is decomposed into DWT components. The LL sub-band at the pre-determined level is dithered and upsampled using nearest neighbor sampling to match the size of the verification string. Each pair of these are XORed, i.e. $V_i \oplus I_{Di}$, to produce S'_{Ti} . Note that if an image was perturbed, $S'_{Ti} \neq S_T$ for that image.

To recover the QR code, Arnold transform is inversed and K is used for decryption to produce a recovered QR code for each image, S_{Ri} . If no images in the dataset were altered, the QR code can be recovered perfectly for all images in the dataset. However, if an image was perturbed, S_{Ri} for that image will result in a noisy QR code.

Algorithm 3 details the steps involved in verifying the fidelity of images in a dataset. The required inputs to the algorithm are the key, K , that was used for encryption, the verification strings and the images in the dataset. For each image in the dataset, the algorithm outputs the recovered QR code. The visual fidelity of each image can be determined based on whether the resulting S_{Ri} is a clean or a noisy QR code. If S_{Ri} is noisy, a clean reconstructed QR code can be obtained by averaging the black and white pixels per module. If there are more white pixels, the module should be white, and

Algorithm 2 Algorithm for generating verification strings for separate color channels.

Input: A QR code, S , a key, K , and images in a dataset, I_i , where $i \in \{1, 2, \dots, n\}$.

Output: Verification strings, V_i , where $i \in \{1, 2, \dots, n\}$

Step 1. Encrypt information in S by XORing the random bits in K with the modules in S to produce S_E .

Step 2. Generate a chaotic image S_T by scrambling the bits in S_E using Arnold transform for a number of iterations.

For each image, I_i , in the dataset, do

Step 3. Split I_i into its corresponding color channels, I_{ij} , where $j \in \{r, g, b\}$ for red, green and blue color channels.

For each image color channel, I_{ij} , do

Step 4. Convert I_{ij} to I_{DWTij} by performing DWT to the desired level.

Step 5. Extract the LL sub-band from I_{DWTij} .

Step 6. Dither the pixels to binarize the extracted LL sub-band into black and white bits (i.e. 0s and 1s).

Step 7. Produce I_{Dij} by upsampling the dithered LL sub-band to match the size of S_T .

Step 8. Generate components of the verification string V_{ij} using $I_{Dij} \oplus S_T$.

Step 9. Produce the verification string V_i by appending the components V_{ij} for all $j \in \{r, g, b\}$, i.e. $V_i = V_{ir} + V_{ig} + V_{ib}$.

Figure 3: Overview of the process to verify each image in a dataset.

vice versa. If the verification strings were generated based on separate color channels, Algorithm 4 presents steps in the process to reconstruct QR codes for verifying the color channels of the images.

Algorithm 3 Algorithm for verifying image fidelity.

Input: The key, K , verification strings, V_i , and images in a dataset, I_i , where $i \in \{1, 2, \dots, n\}$.

Output: Recovered QR codes, S'_{Ri} , where $i \in \{1, 2, \dots, n\}$

For each image, I_i , in the dataset, do

Step 1. Convert I_i to I_{DWTi} by performing DWT to the desired level.

Step 2. Extract the LL sub-band from I_{DWTi} .

Step 3. Dither the pixels to binarize the extracted LL sub-band into black and white bits (i.e. 0s and 1s).

Step 4. Produce I_{Di} by upsampling the dithered LL sub-band to match the size of S_T .

Step 5. Generate S'_{Ti} using $V_i \oplus I_{Di}$.

Step 6. Inverse the Arnold transform on S'_{Ti} to produce S'_{Ei} .

Step 7. Decrypt S'_{Ei} using K to recover the QR code, S_{Ri} .

3.2. Method 2 - Aggregate Verification String (AVS) method

The main drawback of the previously described LVS method, is its requirement to have to store the verification strings of all images in a dataset. Storage requirements are proportional to the number of images in a dataset. While the LVS method may seem impractical, it can be used to independently identify any alteration in an individual image I_i , because the verification string V_i must be stored. Hence, when storage space is not an issue (such as the use of cloud storage), this method is feasible and attractive.

In the event that storage space is limited, the second method address this issue. The aim of the second method is to reduce the required storage size to the size of a *single* verification string. In addition, the use of trapdoor permutation (which can be implemented using an RSA algorithm for example) and a one way hash function, is used to provide additional layers of security.

An overview of the algorithm is depicted in Figure 4. The initial process is similar to the LVS method. After the dithering process, I_{Di} is obtained, which will be XORed with the key K_i and the QR code S . Hence, $I_{Ei} \leftarrow S \oplus K_i \oplus I_{Di}$.

Algorithm 4 Algorithm for verifying image fidelity for separate color channels.

Input: The key, K , verification strings, V_i , and images in a dataset, I_i , where $i \in \{1, 2, \dots, n\}$.

Output: Recovered QR codes, S'_{Ri} , where $i \in \{1, 2, \dots, n\}$

For each image, I_i , in the dataset, do

Step 1. Split I_i into its corresponding color channels, I_{ij} , where $j \in \{r, g, b\}$ for red, green and blue color channels.

For each image color channel, I_{ij} , do

Step 2. Convert I_{ij} to I_{DWTij} by performing DWT to the desired level.

Step 3. Extract the LL sub-band from I_{DWTij} .

Step 4. Dither the pixels to binarize the extracted LL sub-band into black and white bits (i.e. 0s and 1s).

Step 5. Produce I_{Dij} by upsampling the dithered LL sub-band to match the size of S_T .

Step 6. Extract V_{ij} from V_i .

Step 7. Generate S'_{Tij} using $V_{ij} \oplus I_{Dij}$.

Step 8. Inverse the Arnold transform on S'_{Tij} to produce S'_{Eij} .

Step 9. Decrypt S'_{Eij} using K to recover the QR code, S'_{Rij} .

Figure 4: Overview of the AVS method.

However, unlike the LVS method, $K_i \xleftarrow{R} \text{Evaluate}(s, \cdot)$, where the input of the *Evaluate* algorithm is obtained from the hash value of the $I_{E(i-1)}$, which is the output of I_{Ei} from the previous image block. For K_1 of the first image block, an initial string such as the hash value of S or any other initial vector can be used. With this chaining mechanism, the final output block will only comprise of a single verification string, V , regardless the number of images involved. The algorithm for the AVS method is provided in Algorithm 5.

Since trapdoor permutation is used in the construction, the verification will require the use of the $\text{Invert}(t, \cdot)$ algorithm, which cannot be executed without the value of the trapdoor. Therefore, when verifying the fidelity of images in a dataset, if the QR code cannot correctly be recovered at any image block, this means that the image has been altered and the dataset is not intact. On the other hand, if the QR code can be recovered in all image blocks, the dataset is completely intact.

In summary, to highlight the differences between LVS and AVS, in the AVS method, a constant size output is achieved regardless the number of the image blocks. Furthermore, the security level is enhanced by incorporating a symmetric encryption scheme in LVS with a trapdoor permutation algorithm in AVS.

3.3. Security

The security of the AVS algorithm is outlined as follows. First of all, let's recap the algorithm involved.

Algorithm 5 AVS algorithm.

Input: A QR code, S , the first key, K_1 , and images in a dataset, I_i , where $i \in \{1, 2, \dots, n\}$.

Output: A verification string, V

For each image, I_i , in the dataset, do

Step 1. Convert I_i to I_{DWTi} by performing DWT to the desired level.

Step 2. Extract the LL sub-band from I_{DWTi} .

Step 3. Dither the pixels to binarize the extracted LL sub-band into black and white bits (i.e. 0s and 1s).

Step 4. Produce I_{Di} by upsampling the dithered LL sub-band to match the size of S .

Step 5. Generate I_{Ei} using $S \oplus K_i \oplus I_{Di}$. If $i \neq 1$, $K_i = \text{hash}(I_{E(i-1)})$.

For the last image, I_n , output the verification string, $V = I_{En}$.

1. $I_{DWTi} \leftarrow DWT(I_i)$.
2. $I_{Di} \leftarrow \text{Dither and Up-sample}(I_{DWTi})$.
3. $I_{Ei} \leftarrow S \oplus K_i \oplus I_{Di}$, where S is obtained from the QR code, and the key is set to be K_i .
4. $K_i \leftarrow \text{Evaluate}(s, H(I_{E(i-1)}))$, where (s, t) is a pair of public/private key of the user.

Note that the $\text{Evaluate}(\cdot, \cdot)$ function is a trapdoor permutation. The security of this system relies on the difficulty of finding the inverse of this Evaluate algorithm, and hence, the trapdoor permutation. If the trapdoor permutation is implemented with RSA, then

$$\text{Evaluate}(s, H(I_{E(i-1)})) := H(I_{E(i-1)})^s \pmod{N}$$

for a modulus N . This means, the attacker must be able to find the $\text{Invert}(t, \cdot)$ function, which is defined as

$$\text{Invert}(t, z) := z^t \pmod{N}$$

where $st := 1 \pmod{\phi(N)}$. Hence, as long as the RSA problem is secure, then the above algorithm is secure. $\square$

It can further easily be deduced that given the final verification string V (see Fig. 4), it is infeasible to backtrack and obtain all of the other verification

Figure 5: Example of S , S_E and S_T (a) QR code containing a secret message; (b) the encrypted QR code; (c) result after Arnold transform.

parts for each image I_i without knowledge of the secret trapdoor. Hence, the security of the scheme is ensured. ■

4. Experiments and Discussion

This section presents example results obtained from experiments that test the proposed methods. To perform the tests, experiments were conducted by implementing the algorithms and a number of different test images were used in conjunction with the implementations. The test programs were implemented using the OpenCV library.

Fig. 5(a) shows an example of S , i.e. a QR code containing a secret message. The QR code is of version 3, which contains 29×29 modules, with error correction level H. An example of S after encryption, i.e. S_E , is shown in Fig. 5(b). Fig. 5(c) was produced by scrambling the bits in S_E using Arnold transform to generate S_T . Note that in line with Algorithm 1 and Algorithm 2, the same S_T was used in the experiments to obtain the results shown in Table 1 and Table 3.

Table 1 shows examples of some test images that were used in the experiments. These images are commonly used for testing image processing techniques; namely, the lena, peppers and mandrill images, respectively. The table also shows the DWT LL_2 sub-band after the dithering and upsampling processes for the respective test images, i.e. I_{Di} . Finally, the table shows visual depictions of the verification strings V_i for the respective test images, which were generated as a result of $I_{Di} \oplus S_T$. Note that the verification strings are bit strings consisting of 0s and 1s, the table depicts these in a visual form (i.e. 0s are represented using black pixels, and conversely, 1s are represented as white pixels).

Table 1: Example images that were produced as a result of Algorithm 1.

Image, I_i	Dithered and upsampled LL_2 Sub-band, I_{Di}	Visual Depiction of V_i

Table 2 shows example results that were obtained when attempting to verify the image fidelity of perturbed test images. This shows results of Algorithm 3, when tested on images that underwent basic perturbations. The test image was then perturbed using three common techniques; namely, JPEG compression, noise and blurring. For JPEG compression, a value of 95% quality was used in the experiments. To test perturbations resulting from the introduction of noise, 5 random pixels in the test images were altered. For blurring, Gaussian blurring was used with $\sigma = 0.5$. These are small perturbations that typically go unnoticed from the human visual perspective.

The first column in the table shows the test images that were used, while the second column shows the type of perturbation that was performed on the image before running the verification process. The third column in Table 2 depicts the results obtained when attempting to recover the QR code (i.e. S_{Ri}) from the image that underwent perturbation. Note that the recovered QR codes are all noisy, hence, this shows that the image was perturbed and is different from the original image. It can be seen that even small changes to the image (e.g., 5 random pixels), results in a noisy recovered QR code, which demonstrates the effectiveness of the proposed approach in ascertaining whether an image in a dataset has been perturbed. The final column shows QR codes that were reconstructed from S_{Ri} , which were obtained as a result of cleaning up the noisy QR code by averaging the black and white pixels per module (i.e. if there are more white pixels in the noisy QR code constituting a module, the module is reconstructed as a white module, and vice versa). Note that the gray modules represent modules that were incorrectly reconstructed when compared with the original QR code, S .

Table 3 shows example images obtained from running Algorithm 2 on a color image. The first column of the table shows the color image that was used, and the second column shows images of the respective color channels. Results of dithering and upsampling the DWT LL_2 sub-band for each color channel of the image is shown in the third column, and a visual depiction of the generated verification strings is shown in the last column.

The purpose of Algorithm 4 was to verify whether a color image was perturbed using different color channels. Example results obtained from an experiment to test this is presented in Table 4. The previously described perturbations were done on the color image, prior to the verification process. The images of the different color channels can be seen in the first column of Table 4, while the second column shows the type of perturbation that was

Table 2: Examples resulting from Algorithm 3 on perturbed images.

Image, I_i	Type of Perturbation	Recovered QR Code, S_{Ri}	Reconstructed QR Code
Compression: Noise: Blurring:

Compression: Noise: Blurring:

Compression: Noise: Blurring: 20

Table 3: Example images resulting from Algorithm 2, which separates the image color channels.

Color Image, I_i	RGB Color Channel, I_{ij}	I_{Dij} per Color Channel	Visual Depiction of V_{ij}
R:	R:	V_{ir} :
	G:	G:	V_{ig} :
	B:	B:	V_{ib} :

done on the image. Columns three and four show the recovered QR code, S_{Ri} , and reconstructed QR code, respectively. It can be seen from the results in the third column that the recovered QR codes are all noisy as a result of perturbations made to the image. Hence, it can be determined that the image is not intact.

5. Conclusion

In this paper, the problem of protecting image-based machine learning datasets against alteration by an adversary was examined. Two methods were presented, namely, the Linear Verification String (LVS) method and the Aggregate Verification String (AVS) method. The purpose of these methods is to provide mechanisms for verifying the visual fidelity of images in a dataset without the need to store and use the original dataset for verification.

In both the LVS and AVS methods, verification strings were generated from the important visual content of the images and associated with QR codes. In the LVS method, a verification string is generated for each image in a dataset. For verification, a verification process is used whereby each verification string is used to verify the visual fidelity of individual images in the dataset. However, the drawback of the LVS method is that the storage requirement increases linearly with the number of images in a dataset. To overcome this problem, the AVS method was proposed. The AVS method only requires the storage and use of a single verification string to verify whether an entire image dataset is intact. Nonetheless, in the AVS method, one can only ascertain whether the dataset has been altered, but cannot determine the visual fidelity of individual images.

Acknowledgment

The authors would like to acknowledge the support of the NSW Cybersecurity Network grant and the National Natural Science Foundation of China grant (Nos. 61572382 and 61702401) that were awarded for this project.

References

- [1] D. Gibert, C. Mateu, J. Planes, The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, *Journal of Network and Computer Applications* (2020)

Table 4: Examples for different image color channels resulting from Algorithm 4 on a perturbed color image.

RGB Color Channel, I_{ij}	Type of Perturbation	Recovered QR Code, S_{Ri}	Reconstructed QR Code
R:	Compression: Noise: Blurring:
G:	Compression: Noise: Blurring:
B:	Compression: Noise: 23 Blurring:

102526doi:<https://doi.org/10.1016/j.jnca.2019.102526>.

URL <http://www.sciencedirect.com/science/article/pii/S1084804519303868>

- [2] S. Hu, X. Shang, Z. Qin, M. Li, Q. Wang, C. Wang, Adversarial examples for automatic speech recognition: Attacks and counter-measures, *IEEE Communications Magazine* 57 (10) (2019) 120–126. doi:10.1109/MCOM.2019.1900006.
URL <https://doi.org/10.1109/MCOM.2019.1900006>
- [3] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, A. Swami, Practical black-box attacks against machine learning, in: R. Karri, O. Sinanoglu, A. Sadeghi, X. Yi (Eds.), *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, ACM, 2017, pp. 506–519. doi:10.1145/3052973.3053009.
URL <https://doi.org/10.1145/3052973.3053009>
- [4] B. Biggio, F. Roli, Wild patterns: Ten years after the rise of adversarial machine learning, *Pattern Recognition* 84 (2018) 317–331. doi:10.1016/j.patcog.2018.07.023.
URL <https://doi.org/10.1016/j.patcog.2018.07.023>
- [5] B. Biggio, B. Nelson, P. Laskov, Poisoning attacks against support vector machines, in: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Vol. 2, 2012*, pp. 1807–1814.
URL www.scopus.com
- [6] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, J. D. Tygar, Antidote: Understanding and defending against poisoning of anomaly detectors, in: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, IMC '09*, ACM, New York, NY, USA, 2009, pp. 1–14. doi:10.1145/1644893.1644895.
URL <http://doi.acm.org/10.1145/1644893.1644895>
- [7] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, F. Roli, Is feature selection secure against training data poisoning?, in: *32nd International Conference on Machine Learning, ICML 2015, Vol. 2, 2015*, pp. 1689–1698.
URL www.scopus.com

- [8] N. Akhtar, A. S. Mian, Threat of adversarial attacks on deep learning in computer vision: A survey, *IEEE Access* 6 (2018) 14410–14430. doi:10.1109/ACCESS.2018.2807385.
URL <https://doi.org/10.1109/ACCESS.2018.2807385>
- [9] Z. Yan, J. Liu, L. T. Yang, Special issue on pervasive social networking, *Journal of Network and Computer Applications* 86 (2017) 1 – 2, special Issue on Pervasive Social Networking. doi:<https://doi.org/10.1016/j.jnca.2017.03.023>.
URL <http://www.sciencedirect.com/science/article/pii/S108480451730139X>
- [10] Y. Chow, W. Susilo, J. Wang, R. Buckland, J. Baek, J. Kim, N. Li, Protecting the visual fidelity of machine learning datasets using QR codes, in: X. Chen, X. Huang, J. Zhang (Eds.), *Machine Learning for Cyber Security - Second International Conference, ML4CS 2019, Xi'an, China, September 19-21, 2019, Proceedings*, Vol. 11806 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 320–335. doi:10.1007/978-3-030-30619-9_23.
URL https://doi.org/10.1007/978-3-030-30619-9_23
- [11] Denso Wave Incorporated, QRcode.com, <http://www.qrcode.com/en/>.
- [12] International Organization for Standardization, Information technology — automatic identification and data capture techniques — qr code 2005 bar code symbology specification, *ISO/IEC 18004:2006* (2006).
- [13] S. Mallat, A theory for multiresolution signal decomposition: The wavelet representation, *IEEE Trans. Pattern Anal. Mach. Intell.* 11 (7) (1989) 674–693. doi:10.1109/34.192463.
URL <http://dx.doi.org/10.1109/34.192463>
- [14] F. Liu, W. Q. Yan, *Various Applications of Visual Cryptography*, Springer International Publishing, Cham, 2014, pp. 127–143.
- [15] Z.-H. Guan, F. Huang, W. Guan, Chaos-based image encryption algorithm, *Physics Letters A* 346 (1–3) (2005) 153 – 157. doi:<http://dx.doi.org/10.1016/j.physleta.2005.08.006>.
URL <http://www.sciencedirect.com/science/article/pii/S0375960105011904>

- [16] M. Alizadeh, S. Abolfazli, M. Zamani, S. Baharun, K. Sakurai, Authentication in mobile cloud computing: A survey, *Journal of Network and Computer Applications* 61 (2016) 59 – 80. doi:<https://doi.org/10.1016/j.jnca.2015.10.005>. URL <http://www.sciencedirect.com/science/article/pii/S1084804515002258>
- [17] S. Liu, S. Hu, J. Weng, S. Zhu, Z. Chen, A novel asymmetric three-party based authentication scheme in wearable devices environment, *Journal of Network and Computer Applications* 60 (2016) 144 – 154. doi:<https://doi.org/10.1016/j.jnca.2015.10.001>. URL <http://www.sciencedirect.com/science/article/pii/S1084804515002143>
- [18] I. Tkachenko, W. Puech, C. Destruel, O. Strauss, J. Gaudin, C. Guichard, Two-level QR code for private message sharing and document authentication, *IEEE Trans. Information Forensics and Security* 11 (3) (2016) 571–583. doi:10.1109/TIFS.2015.2506546. URL <https://doi.org/10.1109/TIFS.2015.2506546>
- [19] Y. Chow, W. Susilo, J. Tonien, E. Vlahu-Gjorgievska, G. Yang, Cooperative secret sharing using QR codes and symmetric keys, *Symmetry* 10 (4) (2018) 95. doi:10.3390/sym10040095. URL <https://doi.org/10.3390/sym10040095>
- [20] Z. Fu, Y. Cheng, B. Yu, Visual cryptography scheme with meaningful shares based on QR codes, *IEEE Access* 6 (2018) 59567–59574. doi:10.1109/ACCESS.2018.2874527. URL <https://doi.org/10.1109/ACCESS.2018.2874527>
- [21] Y. Chow, W. Susilo, J. Tonien, W. Zong, A QR code watermarking approach based on the DWT-DCT technique, in: J. Pieprzyk, S. Suriadi (Eds.), *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II*, Vol. 10343 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 314–331. doi:10.1007/978-3-319-59870-3_18. URL https://doi.org/10.1007/978-3-319-59870-3_18
- [22] H. Ishizuka, I. Echizen, K. Iwamura, K. Sakurai, A zero-watermarking-like steganography and potential applications, in: *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2014, pp. 459–462. doi:10.1109/IIH-MSP.2014.121.

- [23] V. Seenivasagam, R. Velumani, A QR code based zero-watermarking scheme for authentication of medical images in teleradiology cloud, *Computational and Mathematical Methods in Medicine* 2013 (516465) (2013) 16. doi:10.1155/2013/516465.
- [24] D. Li, Z. Liu, L. Cui, A zero-watermark scheme for identification photos based on QR code and visual cryptography, *International Journal of Security and Its Applications* 10 (1) (2016) 203–214. doi:10.14257/ijisia.2016.10.1.19.
- [25] Q. Kang, K. Li, J. Yang, A digital watermarking approach based on dct domain combining qr code and chaotic theory, in: 2014 Eleventh International Conference on Wireless and Optical Communications Networks (WOCN), 2014, pp. 1–7. doi:10.1109/WOCN.2014.6923098.
- [26] N. Cardamone, F. d’Amore, DWT and QR code based watermarking for document DRM, in: C. D. Yoo, Y. Q. Shi, H. Kim, A. Piva, G. Kim (Eds.), *Digital Forensics and Watermarking - 17th International Workshop, IWDW 2018, Jeju Island, Korea, October 22-24, 2018, Proceedings*, Vol. 11378 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 137–150. doi:10.1007/978-3-030-11389-6_11.
URL https://doi.org/10.1007/978-3-030-11389-6_11
- [27] P. P. Thulasidharan, M. S. Nair, {QR} code based blind digital image watermarking with attack detection code, {AEU} - *International Journal of Electronics and Communications* 69 (7) (2015) 1074 – 1084. doi:http://dx.doi.org/10.1016/j.aeue.2015.03.007.
URL <http://www.sciencedirect.com/science/article/pii/S1434841115001004>
- [28] H.-C. Lee, C.-R. Dong, T.-M. Lin, Digital watermarking based on jnd model and qr code features, in: *Advances in Intelligent Systems and Applications-Volume 2*, Springer, 2013, pp. 141–148.
- [29] R. W. Floyd, L. Steinberg, An Adaptive Algorithm for Spatial Greyscale, *Proceedings of the Society for Information Display* 17 (2) (1976) 75–77.