2019

Protecting the Visual Fidelity of Machine Learning Datasets Using QR Codes

Yang-Wai Chow
University of Wollongong, caseyc@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Jianfeng Wang
Xidian University

Richard Buckland
University of New South Wales

Joon Sang Baek
University of Wollongong, baek@uow.edu.au

See next page for additional authors

Recommended Citation

Chow, Yang-Wai; Susilo, Willy; Wang, Jianfeng; Buckland, Richard; Baek, Joon Sang; Kim, jongkil; and Li, Nan, "Protecting the Visual Fidelity of Machine Learning Datasets Using QR Codes" (2019). *Faculty of Engineering and Information Sciences - Papers: Part B*. 3303.
https://ro.uow.edu.au/eispapers1/3303

Protecting the Visual Fidelity of Machine Learning Datasets Using QR Codes

Abstract

Machine learning is becoming increasingly popular in a variety of modern technology. However, research has demonstrated that machine learning models are vulnerable to adversarial examples in their inputs. Potential attacks include poisoning datasets by perturbing input samples to mislead a machine learning model into producing undesirable results. Such perturbations are often subtle and imperceptible from a human's perspective. This paper investigates two methods of verifying the visual fidelity of image based datasets by detecting perturbations made to the data using QR codes. In the first method, a verification string is stored for each image in a dataset. These verification strings can be used to determine whether an image in the dataset has been perturbed. In the second method, only a single verification string stored and is used to verify whether an entire dataset is intact.

Disciplines

Engineering | Science and Technology Studies

Publication Details

Chow, Y., Susilo, W., Wang, J., Buckland, R., Baek, J., Kim, J. & Li, N. (2019). Protecting the Visual Fidelity of Machine Learning Datasets Using QR Codes. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11806 320-335.

Authors

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

Protecting the Visual Fidelity of Machine Learning Datasets using QR codes

Yang-Wai Chow[1], Willy Susilo[1], Jianfeng Wang[2], Richard Buckland[3], Joonsang Baek[1], Jongkil Kim[1], Nan Li[4]

[1] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, Australia
{caseyc, wsusilo, baek, jongkil}@uow.edu.au,
[2] State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xidian, China
jfwang@xidian.edu.cn,
[3] School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
richardb@unsw.edu.au,
[4] School of Electrical Engineering and Computing, University of Newcastle, Newcastle, Australia
nan.li@newcastle.edu.au

Abstract. Machine learning is becoming increasingly popular in a variety of modern technology. However, research has demonstrated that machine learning models are vulnerable to adversarial examples in their inputs. Potential attacks include poisoning datasets by perturbing input samples to mislead a machine learning model into producing undesirable results. Such perturbations are often subtle and imperceptible from a human's perspective. This paper investigates two methods of verifying the visual fidelity of image based datasets by detecting perturbations made to the data using QR codes. In the first method, a verification string is stored for each image in a dataset. These verification strings can be used to determine whether an image in the dataset has been perturbed. In the second method, only a single verification string stored and is used to verify whether an entire dataset is intact.

Keywords: Adversarial machine learning; cyber security; QR code; visual fidelity; watermarking

1 Introduction

The popularity of Machine Learning (ML) has rapidly grown in recent years, and it has made its way into a variety of modern technology. ML techniques empower a range of diverse applications, including self-driving cars, network intrusion detection, speech recognition, and so on. However, research has demonstrated that ML models are vulnerable to adversarial examples in its inputs. The purpose of this is to use malicious inputs to fool a ML model into producing erroneous outputs [18]. This has given rise to a research field known as adversarial machine learning [3].

There are a number of different adversarial attacks that can be deployed against ML, for example, an adversary can poison a dataset by perturbing samples in the training data [3]. Over the years, researchers have examined and demonstrated the effectiveness of such poisoning attacks [2, 19, 23]. Adversarial attacks are a serious threat to the success of ML in practice, as small and subtle perturbations in the inputs can mislead a ML model into outputting incorrect predictions. In computer vision, such perturbations are often imperceptible to the human visual system [1].

This paper focuses on protecting image based datasets by verifying the visual fidelity of the data. Due to the ML necessity of requiring large 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, this paper 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 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 QR code is noisy or cannot be recovered using this verification process, this means 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. While this is fine if storage space is not an issue, it may not be an attractive solution for applications with limited storage capacity. 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 one cannot determine whether individual images have been altered, but only whether the dataset is intact or has been altered from the original.

Our Contribution. In this paper, we investigate the problem of protecting image based ML datasets against alteration by an adversary. The proposed methods attempt to provide mechanisms for verifying the visual fidelity of images in a dataset without the need to use the original dataset. To do this, we generate verification strings from the visual contents of the images and associate these with QR codes. The reason for using QR codes is due to its inherent data capacity and error correction properties, which are in-built in the QR code structure. We present two methods for creating verification strings and for verifying the fidelity of images. The advantage of the first method, which we named 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, which we named 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 brief background to these concepts followed by a description of related work.

2.1 Preliminaries

QR Code. The Quick Response (QR) code is a two-dimensional (2D) barcode, which was invented by the company Denso Wave [7]. The purpose of using the QR code in this study is because the QR code structure has an inherent error correction mechanism. This mechanism enables QR codes to be correctly decoded even if part of it is corrupted.

A QR code is made up of light and dark modules, which are organized into function patterns and an encoding region [11]. 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. These error correction levels are L (low), M (medium), Q (quartile) and H (high); these correspond to error tolerances of approximately 7%, 15%, 25% and 30%, respectively.

DWT. The Discrete Wavelet Transform (DWT) is a transform domain 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 [17]. An image is decomposed into four sub-bands, which are labeled LL (low-low), LH (low-high), HL (high-low) and HH (high-high). Sub-bands can be further decomposed and this process can continue until the desired number of levels is achieved. 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 [16]. 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.

Arnold transform. Adjacent pixels in images have a strong correlation to each other. The Arnold transform, shown in Eq. 1, is a invertible transform that

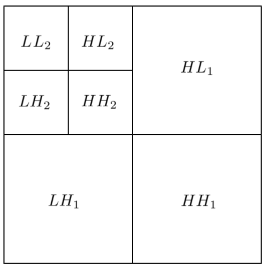

Fig. 1. Sub-bands of a two-level DWT.

can be used to disrupt the correlation between adjacent pixels [10]. An original image that undergoes a number of Arnold transform iterations results in a 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 noise over the image. This increases the potential of being able to recover the QR code despite image perturbations.

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

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 $(s,t) \overset{R}{\leftarrow} Generate$; an evaluation algorithm $Evaluate(s,\cdot)$; and an inversion algorithm $Invert(t,\cdot)$. We require that for all (s,t) produced by the algorithm $Generate$, $Evaluate(s,\cdot)$ 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,

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

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 various applications in computer security. For example, for storing private and public information [22], secret sharing [5], visual cryptography [9], digital watermarking [6] and so on.

The methods proposed in this paper are related to zero-based watermarking. Unlike traditional watermarking techniques, which necessitates that a watermark be embedded within an image, zero-based approaches do not require the embedding of a watermark [12]. Liu and Yan [16] proposed a secret sharing scheme based on zero watermarking. In their approach, the watermark was not embedded within images, but rather cover images are used to create shares that are distributed to participants and one which is registered with a certification authority.

In addition, QR codes have previously been used in conjunction with 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. [13]. In other work, a digital rights management technique for protecting documents by repeatedly inserting a QR code into the DWT sub-band of a document was investigated [4]. Various other QR code watermarking approaches have also been proposed [6, 14, 21].

Zero-based watermarking scheme using QR codes have also been proposed. As an example, an authentication method for medical images using zero watermarking and QR codes was devised. In this scheme, a patient's identification details and a link their data was encoded in the form of a QR code that serves as the watermark [20]. Similarly, Li et al. [15] proposed a QR code based zero watermarking scheme in conjunction with visual cryptography for authenticating identification photos.

3 Proposed Methods

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 limited storage space is an issue. As such, the second method has a very low storage requirement, and provides a mechanism to verify whether an entire dataset is intact. Nevertheless, in the second method, while a user can determine if a dataset is not intact, the user cannot identify which of the images have been altered.

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, we call it 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 images in a dataset do not have to be compared with their respective original image. Moreover, the verification strings will require much less storage space when compared with the entire image dataset.

The motivation behind this approach is based on zero-based image watermarking [16]. Unlike traditional watermarking techniques that requires a 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 samples in a dataset have been altered.

Generating Verification Strings. Fig. 2 gives an overview of the process used to generate verification strings for all the images in a dataset. It can be seen from the figure that to create the verification strings, a QR code that contains the 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 pseudorandom 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.

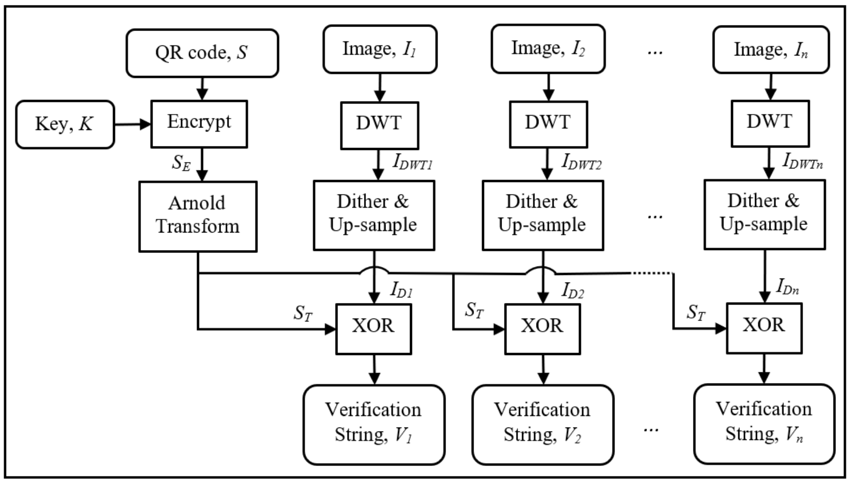

Fig. 2. Overview of the process to generate a verification string for each image in a dataset.

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, level 2 DWT was used in experiments in this paper. Note that color images will first have to be converted to greyscale. 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 grey level distribution. In the experiments, we adopted the Floyd-Steinberg dithering technique, an approach that is based on error diffusion [8].

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. Finally, a verification string, V_i, is produced for each image in the dataset by XORing S_T with the dithered and upsampled results.

The details of an algorithm to generate verification strings for all images in a dataset is provided in Algorithm 1. Inputs to the algorithm 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.

Algorithm 1 Algorithm for generating verification strings.

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

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. Exact 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 $I_{Di} \oplus S_T$.
```

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 were perturbed. An overview of this process is depicted in Fig. 3. Similar to the process of 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 2 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

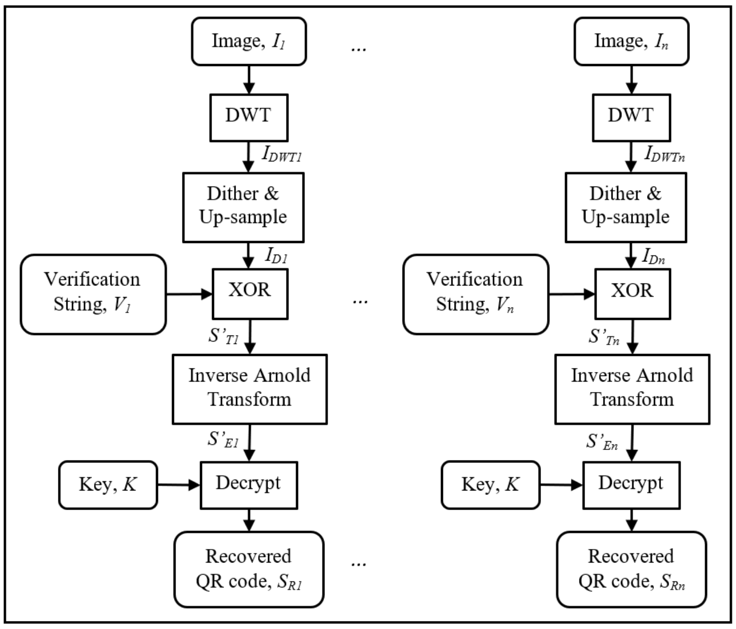

Fig. 3. Overview of the process to verify each image in a dataset.

```
Algorithm 2 Algorithm for verifying image fidelity.
  Input: The key, $K$, verification strings, $V_i$, and images in a dataset, $I_i$, where $i \in \{1, 2, ..., n\}$.
  Output: Recovered QR codes, $S'_{Ri}$, where $i \in \{1, 2, ..., n\}$

  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. Exact 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, $S_{Ri}$.
```

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 the 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 vice versa.

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 the 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 our second method, we address the storage space issue, and aim to reduce its size to a *single* verification string. In addition, we make use of trapdoor permutation (which can be implemented using an RSA algorithm for example) and a one way hash function, to provide additional layers of security.

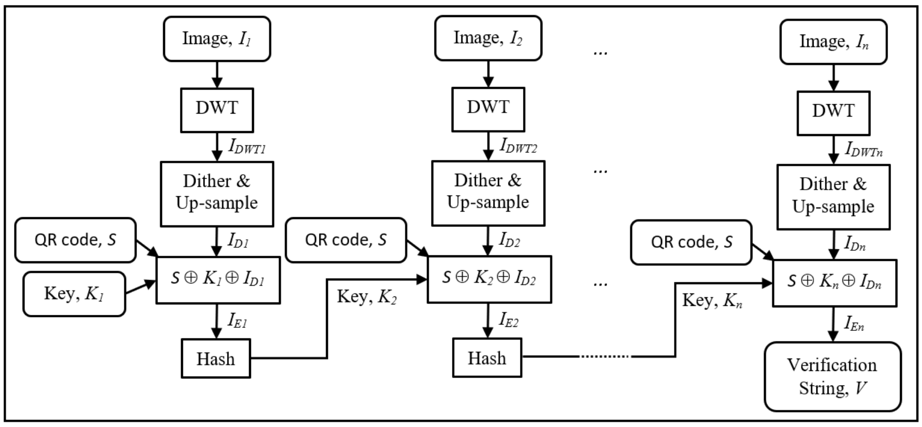

Fig. 4. Overview of the AVS method.

An overview of the AVS method is depicted in Figure 4, and the algorithm is detailed in Algorithm 3. The initial process is similar to the LVS method. After the dithering process, we obtain I_{Di}, which will be XORed with the key K_i and the QR code S. Hence, $I_{Ei} \leftarrow S \oplus K_i \oplus I_{Di}$. However, unlike the LVS method, $K_i \xrightarrow{R} Evaluate(s, \cdot)$, 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 the first image block, we can use the initial string, such as the hash value of S or any other initial vector. With this chained mechanism, the final output block will only comprise of a single verification string, V, regardless the number of images involved.

Algorithm 3 AVS algorithm.

```
Input: A QR code, $S$, the first key, $K_1$, and images in a dataset, $I_i$, where $i \in \{1, 2, ..., n\}$.
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 $i \neq 1$, $K_i = hash(I_{E(i-1)})$.

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

Furthermore, as we use trapdoor permutation in the construction, the verification will require the use of the $Invert(t, \cdot)$ algorithm, which cannot be executed without the value of the trapdoor. Therefore, when verifying the 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, we have achieved a constant size output regardless the number of the image blocks in AVS. Furthermore, we enhance the security level from incorporating a symmetric encryption scheme in LVS with a trapdoor permutation algorithm in AVS.

4 Results and Discussion

An implementation of Algorithms 1 and 2 was implemented using the OpenCV library. This was tested using a number of test images. Fig. 5 and 6 show examples of results obtained from the experiments. Additional results can be found in Fig. 7 and 8 in the Appendix section of this paper.

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, the same S_T was used in the experiment to obtain the results shown in Fig. 6, 7 and 8. The test images

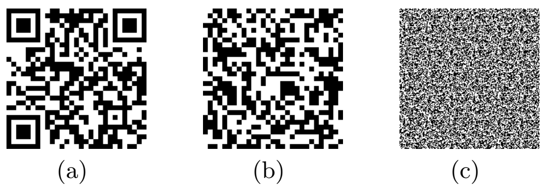

Fig. 5. Example of S, S_E and S_T (a) QR code containing secret messge; (b) encrypted QR code; (c) result after Arnold transform.

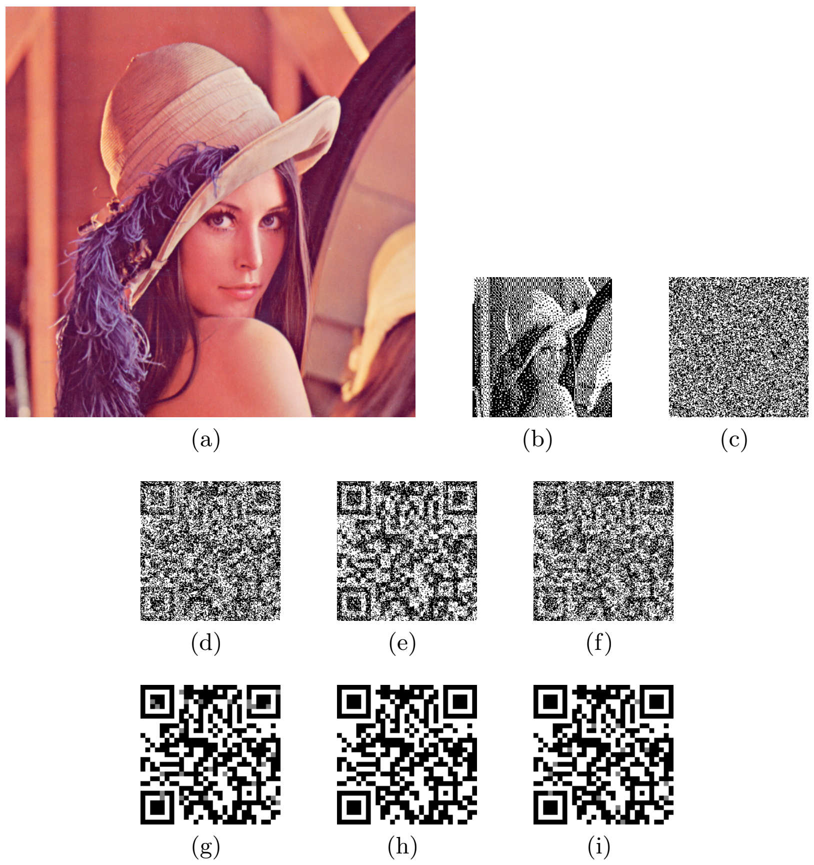

Fig. 6. Example results for the 'lena' image (a) input image; (b) dithered LL_2 subband; (c) visual depiction of the verification string; (d) S_R after JPEG compression; (e) S_R after noise; (f) S_R after blurring; (g) reconstructed QR code from Fig. 6(d); (h) reconstructed QR code from Fig. 6(e); (i) reconstructed QR code from Fig. 6(f).

were commonly used image processing test images; namely, the lena, peppers and mandrill images, respectively.

Results on the lena image are provided in Fig. 6. Fig. 6(a) shows the input image, I. In Fig. 6(b), the LL_2 sub-band of Fig. 6(a) was dithered and the result was upsampled to match the size of S_T. A visual depiction of the verification string, V, that was generated as a result of XORing all bits in I_D with all bits in S_T is shown in Fig. 6(a). Note that this is stored as a bit string rather than as an image to conserve memory storage requirements.

The test image was then perturbed using three common techniques; namely, JPEG compression, noise and blurring. For JPEG compression, OpenCV JPEG compression with a value of 95% quality was used. Fig. 6(d) shows the result of using the verification string to recover the QR code from the JPEG compressed image. Fig. 6(g) in turn shows a clean QR code that was reconstructed from S_R shown in Fig. 6(d), the grey modules indicate incorrect modules in the reconstruction. To test perturbations resulting from noise, 5 random pixels in the test image were altered. Fig. 6(e) and Fig. 6(h) show the recovered QR code and a reconstructed version, respectively. For the blurring test, Gaussian blurring was used with $\sigma = 0.5$. The recovered QR code after blurring and the reconstructed versions are shown in Fig. 6(f) and Fig. 6(i), respectively. Similar experiments using other test images are provided in the Appendix.

5 Conclusion

This paper investigates the problem of protecting image based ML datasets against alteration by an adversary. Two methods are presented in this paper; namely, the Linear Verification String (LVS) method and the Aggregate Verification String (AVS) method. The purpose of these methods is to provide a mechanism for verifying the visual fidelity of images in a dataset without the need to use the original dataset. In both methods, we generate verification strings from the visual contents of the images and associate these 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 solve this problem, we proposed the AVS method which only requires the 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, the NUW Alliance grant and the National Natural Science

Foundation of China (Nos. 61572382 and 61702401) that were awarded for this research.

References

1. N. Akhtar and A. S. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.
2. B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, volume 2, pages 1807–1814, 2012.
3. B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.
4. N. Cardamone and F. d'Amore. DWT and QR code based watermarking for document DRM. In C. D. Yoo, Y. Q. Shi, H. Kim, A. Piva, and G. Kim, editors, *Digital Forensics and Watermarking - 17th International Workshop, IWDW 2018, Jeju Island, Korea, October 22-24, 2018, Proceedings*, volume 11378 of *Lecture Notes in Computer Science*, pages 137–150. Springer, 2018.
5. Y. Chow, W. Susilo, J. Tonien, E. Vlahu-Gjorgievska, and G. Yang. Cooperative secret sharing using QR codes and symmetric keys. *Symmetry*, 10(4):95, 2018.
6. Y. Chow, W. Susilo, J. Tonien, and W. Zong. A QR code watermarking approach based on the DWT-DCT technique. In J. Pieprzyk and S. Suriadi, editors, *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II*, volume 10343 of *Lecture Notes in Computer Science*, pages 314–331. Springer, 2017.
7. Denso Wave Incorporated. QRcode.com, http://www.qrcode.com/en/.
8. R. W. Floyd and L. Steinberg. An Adaptive Algorithm for Spatial Greyscale. *Proceedings of the Society for Information Display*, 17(2):75–77, 1976.
9. Z. Fu, Y. Cheng, and B. Yu. Visual cryptography scheme with meaningful shares based on QR codes. *IEEE Access*, 6:59567–59574, 2018.
10. Z.-H. Guan, F. Huang, and W. Guan. Chaos-based image encryption algorithm. *Physics Letters A*, 346(13):153 – 157, 2005.
11. International Organization for Standardization. Information technology — automatic identification and data capture techniques — qr code 2005 bar code symbology specification. ISO/IEC 18004:2006, 2006.
12. H. Ishizuka, I. Echizen, K. Iwamura, and K. Sakurai. A zero-watermarking-like steganography and potential applications. In *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 459–462, Aug 2014.
13. Q. Kang, K. Li, and 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)*, pages 1–7, Sept 2014.
14. H.-C. Lee, C.-R. Dong, and T.-M. Lin. Digital watermarking based on jnd model and qr code features. In *Advances in Intelligent Systems and Applications-Volume 2*, pages 141–148. Springer, 2013.
15. D. Li, Z. Liu, and 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):203–214, 2016.

16. F. Liu and W. Q. Yan. *Various Applications of Visual Cryptography*, pages 127–143. Springer International Publishing, Cham, 2014.
17. S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):674–693, 1989.
18. N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In R. Karri, O. Sinanoglu, A. Sadeghi, and X. Yi, editors, *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pages 506–519. ACM, 2017.
19. B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and 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, pages 1–14, New York, NY, USA, 2009. ACM.
20. V. Seenivasagam and 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):16, 2013.
21. P. P. Thulasidharan and M. S. Nair. {QR} code based blind digital image watermarking with attack detection code. *{AEU} - International Journal of Electronics and Communications*, 69(7):1074 – 1084, 2015.
22. I. Tkachenko, W. Puech, C. Destruel, O. Strauss, J. Gaudin, and C. Guichard. Two-level QR code for private message sharing and document authentication. *IEEE Trans. Information Forensics and Security*, 11(3):571–583, 2016.
23. H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli. Is feature selection secure against training data poisoning? In *32nd International Conference on Machine Learning, ICML 2015*, volume 2, pages 1689–1698, 2015.

Appendix

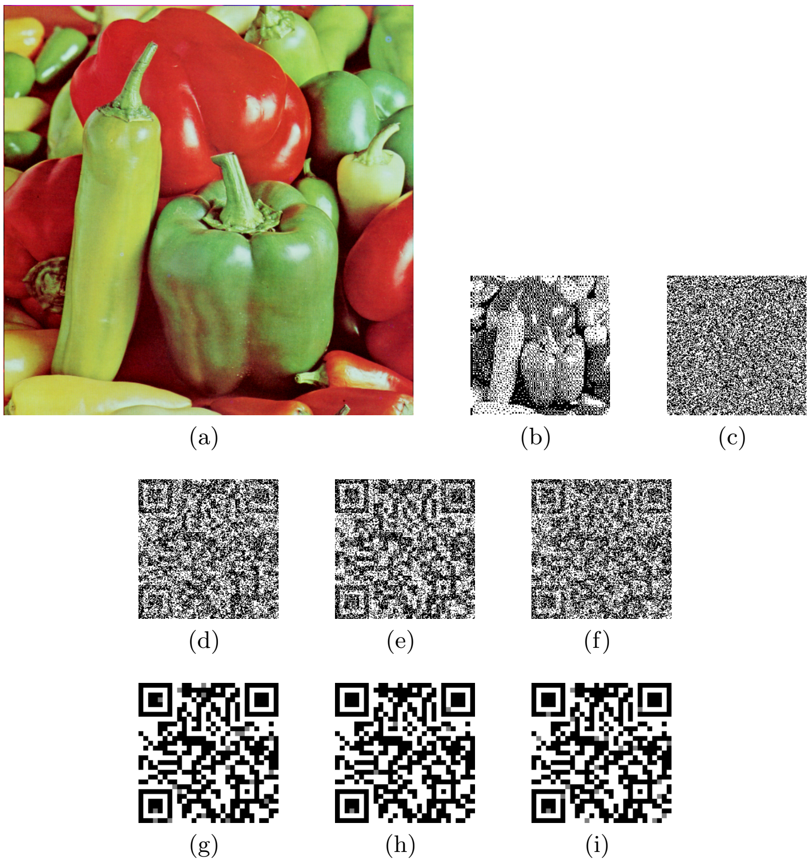

Fig. 7. Example results for 'peppers' image (a) input image; (b) dithered LL_2 subband; (c) visual depiction of the verification string; (d) S_R after JPEG compression; (e) S_R after noise; (f) S_R after blurring; (g) reconstructed QR code from Fig. 7(d); (h) reconstructed QR code from Fig. 7(e); (i) reconstructed QR code from Fig. 7(f).

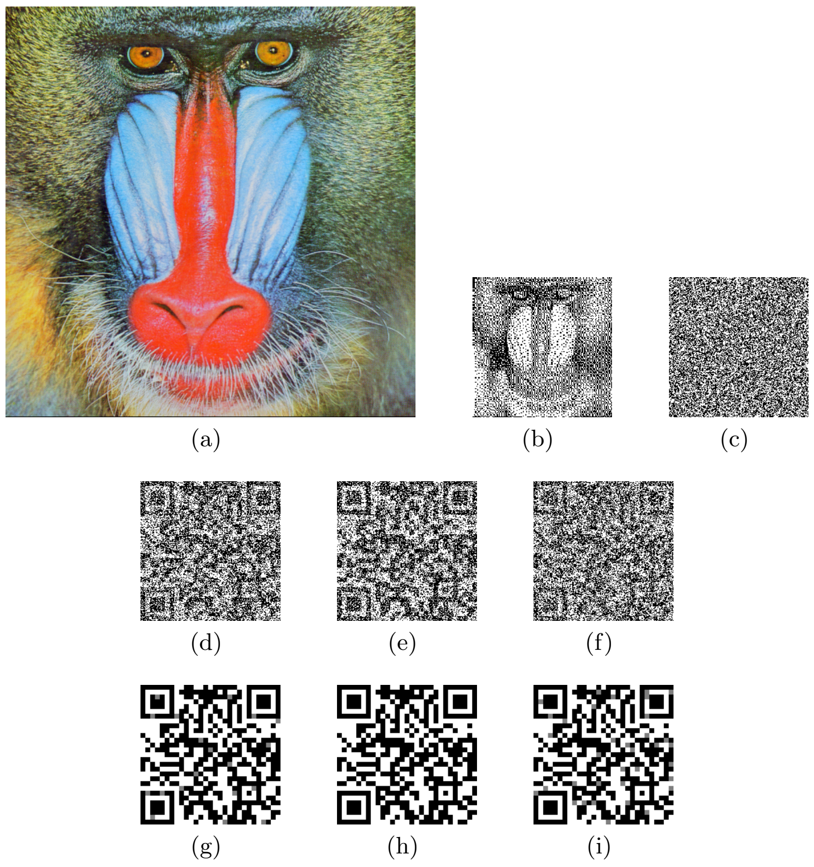

Fig. 8. Example results for 'mandrill' image (a) input image; (b) dithered LL_2 subband; (c) visual depiction of the verification string; (d) S_R after JPEG compression; (e) S_R after noise; (f) S_R after blurring; (g) reconstructed QR code from Fig. 8(d); (h) reconstructed QR code from Fig. 8(e); (i) reconstructed QR code from Fig. 8(f).