Proactive Image Manipulation Detection

Vishal Asnani[1], Xi Yin[2], Tal Hassner[2], Sijia Liu[1], Xiaoming Liu[1]
[1]Michigan State University, [2]Meta AI
[1]{asnanivi, liusiji5, liuxm}@msu.edu, [2]{yinxi, thassner}@fb.com

Abstract

Image manipulation detection algorithms are often trained to discriminate between images manipulated with particular Generative Models (GMs) and genuine/real images, yet generalize poorly to images manipulated with GMs unseen in the training. Conventional detection algorithms receive an input image passively. By contrast, we propose a proactive scheme to image manipulation detection. Our key enabling technique is to estimate a set of templates which when added onto the real image would lead to more accurate manipulation detection. That is, a template protected real image, and its manipulated version, is better discriminated compared to the original real image vs. its manipulated one. These templates are estimated using certain constraints based on the desired properties of templates. For image manipulation detection, our proposed approach outperforms the prior work by an average precision of 16% *for CycleGAN and* 32% *for GauGAN. Our approach is generalizable to a variety of GMs showing an improvement over prior work by an average precision of* 10% *averaged across* 12 *GMs. Our code is available at* https://www.github.com/vishal3477/proactive_IMD.

1. Introduction

It's common for people to share personal photos on social networks. Recent developments of image manipulation techniques via Generative Models (GMs) [13] result in serious concerns over the authenticity of the images. As these techniques are easily accessible [7,8,21,27,31,44,61], the shared images are at a greater risk for misuse after manipulation. Generation of fake images can be categorized into two types: entire image generation and partial image manipulation [46, 48]. While the former generates entirely new images by feeding a noise code to the GM, the latter involves the partial manipulation of a real image. Since the latter alters the semantics of real images, it is generally considered as a greater risk, and thus partial image manipulation detection is the focus of this work.

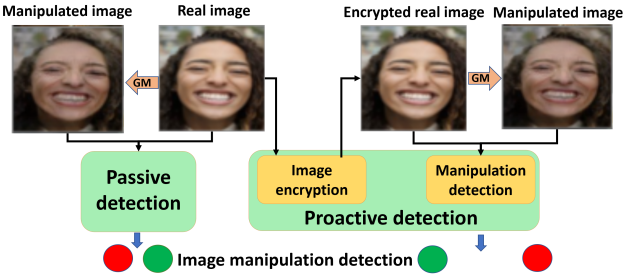

Figure 1. Passive *vs*. proactive image manipulation detection Classic passive schemes take an image as it is to discriminate a real image *vs*. its manipulated one created by a Generative Model (GM). In contrast, our proactive scheme performs encryption of the real image so that our detection module can better discriminate the encrypted real image *vs*. its manipulated counterpart.

Detecting such manipulation is an important step to alleviate societal concerns on the authenticity of shared images. Prior works have been proposed to combat manipulated media [12]. They leverage properties that are prone to being manipulated, including mouth movement [39], steganalysis features [51], attention mechanism [11, 23], *etc*. However, these methods are often overfitted to the image manipulation method and the dataset used in training, and suffer when tested on data with a different distribution.

All the aforementioned methods adopt a *passive* scheme since the input image, being real or manipulated, is accepted as is for detection. Alternatively, there is also a *proactive* scheme proposed for a few computer vision tasks, which involves adding signals to the original image. For example, prior works add a predefined template to real images which either disrupt the output of the GM [40, 41, 54] or tag images to real identities [46]. This template is either a one-hot encoding [46] or an adversarial perturbation [40, 41, 54].

Motivated by improving the generalization of manipulation detection, as well as the proactive scheme for other tasks, this paper proposes a *proactive* scheme for the purpose of image manipulation detection, which works as follows. When an image is captured, our algorithm adds an imperceptible signal (termed as *template*) to it, serving as an

Table 1. Comparison of our approach with prior works. Generalizable column means if the performance is reported on datasets unseen during training. [Keys: Img. man. det.: Image manipulation detection, Img. ind.: Image independent]

Method	Year	Detection scheme	Purpose	Manipulation type	Generalizable	Add perturbation	Recover perturbation	Template learning method	# of templates	Img. ind. templates
Cozzolino *et al*. [10]	2018	Passive	Img. man. det.	Entire/Partial	✔	✗	✗	-	-	-
Nataraj *et al*. [28]	2019	Passive	Img. man. det.	Entire/Partial	✔	✗	✗	-	-	-
Rossler *et al*. [39]	2019	Passive	Img. man. det.	Entire/Partial	✗	✗	✗	-	-	-
Zhang *et al*. [59]	2019	Passive	Img. man. det.	Partial	✔	✗	✗	-	-	-
Wang *et al*. [48]	2020	Passive	Img. man. det.	Entire/Partial	✔	✗	✗	-	-	-
Wu *et al*. [51]	2020	Passive	Img. man. det.	Entire/Partial	✗	✗	✗	-	-	-
Qian *et al*. [35]	2020	Passive	Img. man. det.	Entire/Partial	✗	✗	✗	-	-	-
Dang *et al*. [11]	2020	Passive	Img. man. det.	Partial	✗	✗	✗	-	-	-
Masi *et al*. [26]	2020	Passive	Img. man. det.	Partial	✗	✗	✗	-	-	-
Nirkin *et al*. [29]	2021	Passive	Img. man. det.	Partial	✗	✗	✗	-	-	-
Asnani *et al*. [3]	2021	Passive	Img. man. det.	Entire/Partial	✔	✗	✗	-	-	-
Segalis *et al*. [41]	2020	Proactive	Deepfake disruption	Partial	✗	✔	✗	Adversarial attack	1	✔
Ruiz *et al*. [40]	2020	Proactive	Deepfake disruption	Partial	✗	✔	✗	Adversarial attack	1	✔
Yeh *et al*. [54]	2020	Proactive	Deepfake disruption	Partial	✗	✔	✗	Adversarial attack	1	✔
Wang *et al*. [46]	2021	Proactive	Deepfake tagging	Partial	✗	✔	✔	Fixed template	> 1	✗
Ours	-	Proactive	Img. man. det.	Partial	✔	✔	✔	Unsupervised learning	> 1	✔

encryption. If this encrypted image is shared and manipulated through a GM, our algorithm accurately distinguishes between the encrypted image and its manipulated version by recovering the added template. Ideally, this encryption process could be incorporated into the camera hardware to protect all images after being captured. In comparison, our approach differs from related proactive works [40,41,46,54] in its purpose (detection vs other tasks), template learning (learnable vs predefined), the number of templates, and the generalization ability.

Our key enabling technique is to learn a template set, which is a non-trivial task. First, there is no ground truth template for supervision. Second, recovering the template from manipulated images is challenging. Third, using one template can be risky as the attackers may reverse engineer the template. Lastly, image editing operations such as blurring or compression could be applied to encrypted images, diminishing the efficacy of the added template.

To overcome these challenges, we propose a template estimation framework to learn a set of *orthogonal templates*. We perform image manipulation detection based on the recovery of the template from encrypted real and manipulated images. Unlike prior works, we use unsupervised learning to estimate this template set based on certain constraints. We define different loss functions to incorporate properties including small magnitude, more high frequency content, orthogonality and classification ability as constraints to learn the template set. We show that our framework achieves superior manipulation detection than State-of-The-Art (SoTA) methods [10, 28, 46, 59]. We propose a novel evaluation protocol with 12 different GMs, where we train on images manipulated by one GM and test on unseen GMs. In summary, the contributions of this paper include:

- We propose a novel proactive scheme for image manipulation detection.
- We propose to learn a set of templates with desired properties, achieving higher performance than a single template approach.
- Our method substantially outperforms the prior works on image manipulation detection. Our method is more generalizable to different GMs showing an improvement of 10% average precision averaged across 12 GMs.

2. Related Works

Passive deepfake detection. Most deepfake detection methods are passive. Wang *et al*. [48] perform binary detection by exploring frequency domain patterns from images. Zhang *et al*. [59] propose to extract the median and high frequencies to detect the upsampling artifacts by GANs. Asnani *et al*. [3] propose to estimate fingerprint using certain desired properties for generative models which produce fake images. Others use autoencoders [10], hand-crafted features [28], face-context discrepancies [29], mouth and face motion [39], steganalysis features [51], xception-net [9], frequency domain [26] and attention mechanisms [11]. These aforementioned passive deepfake detection methods suffer from generalization. We propose a novel proactive scheme for manipulation detection, aiming to improve the generalization.

Proactive schemes. Recently, some proactive methods are proposed by adding an adversarial noise onto the real image. Ruiz *et al*. [40] perform deepfake disruption by using adversarial attack in image translation networks. Yeh *et al*. [54] disrupt deepfakes to low quality images by performing adversarial attacks on real images. Segalis *et al*. [41] disrupt manipulations related to face-swapping by adding small perturbations. Wang *et al*. [46] propose a method to tag images by embedding messages and recovering them after manipulation. Wang *et al*. [46] use a one-hot encoding message instead of adversarial perturbations. Compared with these works, our method focuses on image manipulation detection rather than deepfake disruption or deepfake tagging. Our method learns a set of templates and recovers

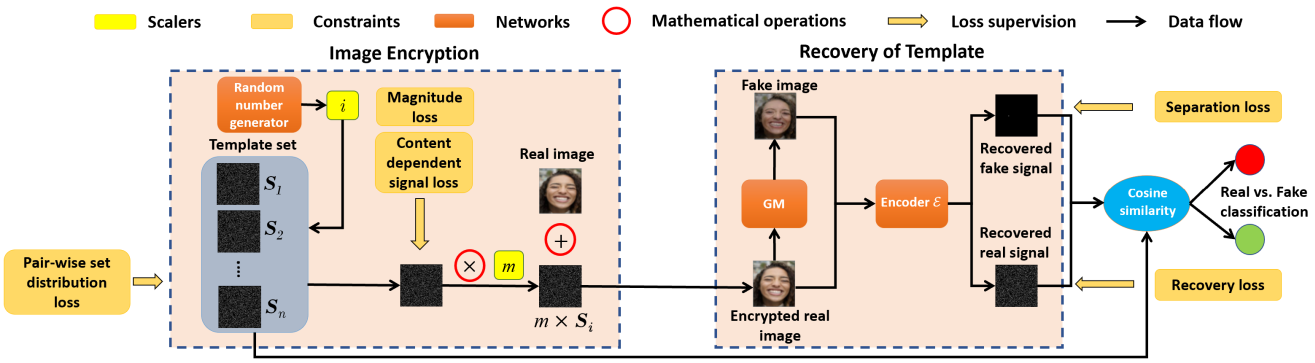

Figure 2. Our proposed framework includes two stages: 1) selection and addition of templates; and 2) the recovery of the estimated template from encrypted real images and manipulated images using an encoder network. The GM is used in the inference mode. Both stages are trained in an end-to-end manner to output a set of templates. For inferences, the first stage is mandatory to encrypt the images. The second stage is used only when there is a need of image manipulation detection.

the added template for image manipulation detection. Our method also generalizes better to unseen GMs than prior works. Tab. 1 summarizes the comparison with prior works.

Watermarking and cryptography methods. Digital watermarking methods have been evolving from using classic image transformation techniques to deep learning techniques. Prior work have explored different ways to embed watermarks through pixel values [5] and spatial domain [42]. Others [18, 20, 52] use frequency domains including transformation coefficients obtained via SVD, discrete wavelet transform (DWT), discrete cosine transform (DCT) and discrete fourier transform (DFT) to embed watermarks. Recently, deep learning techniques proposed by Zhu *et al*. [60], Baluja *et al*. [4] and Tancik *et al*. [43] use an encoder-decoder architecture to embed watermarks into an image. All of these methods aim to either hide sensitive information or protect the ownership of digital images. While our algorithm shares the high-level idea of image encryption, we develop a novel framework for an entirely different purpose, *i.e*., proactive image manipulation detection.

3. Proposed Approach

3.1. Problem Formulation

We only consider GMs which perform partial image manipulation that takes a real image as input for manipulation. Let $\boldsymbol{X}^a$ be a set of real images which when given as input to a GM G would output $G(\boldsymbol{X}^a)$, a set of manipulated images. Conventionally, passive image manipulation detection methods perform binary classification on $\boldsymbol{X}^a$ *vs*. $G(\boldsymbol{X}^a)$. Denote $\boldsymbol{X} = \{\boldsymbol{X}^a, G(\boldsymbol{X}^a)\} \in \mathbb{R}^{128\times128\times3}$ as the set of real and manipulated images, the objective function for passive detection is formulated as follows:

$$\min_{\theta}\Big\{-\sum_{j}\Big(y_j.\log(\mathcal{H}(\boldsymbol{X}_j;\theta))-(1-y_j).\log(1-\mathcal{H}(\boldsymbol{X}_j;\theta))\Big)\Big\}. \tag{1}$$

where y is the class label and $\mathcal{H}$ refers to the classification network used with parameters θ.

In contrast, for our proactive detection scheme, we apply a transformation $\mathcal{T}$ to a real image from set $\boldsymbol{X}^a$ to formulate a set of encrypted real images represented as: $\mathcal{T}(\boldsymbol{X}^a)$. We perform image encryption by adding a learnable template to the image which acts as a defender's signature. Further, the set of encrypted real images $\mathcal{T}(\boldsymbol{X}^a)$ is given as input to the GM, which produces a set of manipulated images $G(\mathcal{T}(\boldsymbol{X}^a))$. We propose to learn a set of templates rather than a single one to increase security as it is difficult to reverse engineer all templates. Thus for a real image $\boldsymbol{X}^a_j \in \boldsymbol{X}^a$, we define $\mathcal{T}$ via a set of n orthogonal templates $\mathcal{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, ...\boldsymbol{S}_n\}$ where $\boldsymbol{S}_i \in \mathbb{R}^{128\times128}$ as follows:

$$\mathcal{T}(\boldsymbol{X}^a_j) = \boldsymbol{X}^a_j + \boldsymbol{S}_i, \text{ where } i \in \{1, 2, ..., n\}. \tag{2}$$

After applying the transformation $\mathcal{T}$, the objective function defined in Eqn. 1 can be re-written as:

$$\min_{\theta,\boldsymbol{S}_i}\Big\{-\sum_{j}\Big(y_j.\log(\mathcal{H}(\mathcal{T}(\boldsymbol{X}_j);\theta,\boldsymbol{S}_i))+ (1-y_j).\log(1-\mathcal{H}(\mathcal{T}(\boldsymbol{X}_j);\theta,\boldsymbol{S}_i))\Big)\Big\}. \tag{3}$$

The goal is to find $\boldsymbol{S}_i$ for which corresponding images in $\boldsymbol{X}^a$ and $\mathcal{T}(\boldsymbol{X}^a)$ have no significant visual difference. More importantly, if $\mathcal{T}(\boldsymbol{X}^a)$ is modified by any GM, this would improve the performance for image manipulation detection.

3.2. Proposed Framework

As shown in Fig. 2, our framework consists of two stages: image encryption and recovery of template. The first stage is used for selection and addition of templates, while the second stage involves the recovery of templates from images in $\mathcal{T}(\boldsymbol{X}^a)$ and $G(\mathcal{T}(\boldsymbol{X}^a))$. Both stages are

(a) (b) (c) (d) (e) (f)

Figure 3. Visualization of (a) a template set with the size of 3, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Each row corresponds to image manipulation by different GM (top: StarGAN, middle: CycleGAN, bottom: GauGAN). The template recovered from encrypted real images is more similar to the template set than the one from manipulated images. The addition of the template creates no visual difference between real and encrypted real images. We provide more examples of real images evaluated using our framework in the supplementary material.

trained in an end-to-end manner with GM parameters fixed. For inference, each stage is applied separately. The first stage is a mandatory step to encrypt the real images while the second stage would only be used when image manipulation detection is needed.

3.2.1 Image Encryption

We initialize a set of n templates as shown in Fig. 2, which is optimized during training using certain constraints. As formulated in Eqn. 2, we randomly select and add a template from our template set to every real image. Our objective is to estimate an optimal template set from which any template is capable of protecting the real image in $\boldsymbol{X}^a$.

Although we constrain the magnitude of the templates using L_2 loss, the added template still degrades the quality of the real image. Therefore, when adding the template to real images, we control the strength of the added template using a hyperparameter m. We re-define $\mathcal{T}$ as follows:

$$\mathcal{T}(\boldsymbol{X}^a_j) = \boldsymbol{X}^a_j + m \times \boldsymbol{S}_i \text{ where } i \in \{1, 2, ..., n\}. \quad (4)$$

We perform an ablation study of varying m in Sec. 4.3, and find that setting m at 30% performs the best.

3.2.2 Recovery of Templates

To perform image manipulation detection as shown in Fig. 2, we attempt to recover our added template from images in $\mathcal{T}(\boldsymbol{X}^a)$ using an encoder $\mathcal{E}$ with parameters $\theta_\mathcal{E}$. For any real image $\boldsymbol{X}^a_j \in \boldsymbol{X}^a$, we define the recovered template from encrypted real image $\mathcal{T}(\boldsymbol{X}^a_j)$ as $\boldsymbol{S}_R = \mathcal{E}(\mathcal{T}(\boldsymbol{X}^a_j))$ and from manipulated image $G(\mathcal{T}(\boldsymbol{X}^a_j))$ as $\boldsymbol{S}_F = \mathcal{E}(G(\mathcal{T}(\boldsymbol{X}^a_j)))$. As template selection from the template set is random, the encoder receives more training pairs to learn how to recover any template from an image, which contributes positively to the robustness of the recovery process. We visualize our trained template set $\mathcal{S}$, and the recovered templates $\boldsymbol{S}_{R/F}$ in Fig. 3.

The main intuition of our framework design is that $\boldsymbol{S}_R$ should be much more similar to the added template and vice-versa for $\boldsymbol{S}_F$. Thus, to perform image manipulation detection, we calculate the cosine similarity between $\boldsymbol{S}_{R/F}$ and all learned templates in the set $\mathcal{S}$ rather than merely using a classification objective. For every image, we select the maximum cosine similarity across all templates as the final score. Therefore, we update logit scores in Eqn. 3 by cosine similarity scores as shown below:

$$\min_{\theta_\mathcal{E}, \boldsymbol{S}_i} \Big\{ -\sum_j \Big(y_j . \log(\max_{i=1...n} (\text{Cos}(\mathcal{E}(\mathcal{T}(\boldsymbol{X}_j); \theta_\mathcal{E}), \boldsymbol{S}_i))) + (1 - y_j) . \log(1 - \max_{i=1...n} (\text{Cos}(\mathcal{E}(\mathcal{T}(\boldsymbol{X}_j); \theta_\mathcal{E}), \boldsymbol{S}_i))) \Big) \Big\}. \quad (5)$$

3.2.3 Unsupervised Training of Template Set

Since there is no ground truth for supervision, we define various constraints to guide the learning process. Let $\boldsymbol{S}$ be the template selected from set $\mathcal{S}$ to be added onto a real image. We formulate five loss functions as shown below.

Magnitude loss. The real image and the encrypted image should be as similar as possible visually as the user does not want the image quality to deteriorate after template addition. Therefore, we propose the first constraint to regularize the magnitude of the template:

$$J_m = ||\boldsymbol{S}||_2^2. \quad (6)$$

Recovery loss. We use an encoder network to recover the added template. Ideally, the encoder output, *i.e.*, the recovered template $\boldsymbol{S}_R$ of the encrypted real image, should be the same as the original added template $\boldsymbol{S}$. Thus, we propose to maximize the cosine similarity between these two templates:

$$J_r = 1 - \text{Cos}(\boldsymbol{S}, \boldsymbol{S}_R). \quad (7)$$

Content independent template loss. Our main aim is to learn a set of universal templates which can be used for detecting manipulated images from unseen GMs. These templates, despite being trained on one dataset, can be applied to images from a different domain. Therefore, we encourage the high frequency information in the template to be data independent. We propose a constraint to minimize low frequency information:

$$J_c = ||\mathcal{L}(\mathbb{F}(\boldsymbol{S}), k)||_2^2, \quad (8)$$

where $\mathcal{L}$ is the low pass filter selecting the $k \times k$ region in the center of the 2D Fourier spectrum, while assigning the high frequency region to zero. $\mathbb{F}$ is the Fourier transform.

Separation loss. We want the recovered template $\boldsymbol{S}_F$ from manipulated images $G(\mathcal{T}(\boldsymbol{X}))$ to be different than all the templates in set $\mathcal{S}$. Thus, we optimize $\boldsymbol{S}_F$ to be orthogonal to all the templates in the set $\mathcal{S}$. Therefore, we take the template for which the cosine similarity between $\boldsymbol{S}_F$ and the template is maximum, and minimize its respective cosine similarity:

$$J_s = \max_{i=1...n} (\text{Cos}(\mathcal{N}(\boldsymbol{S}_i), \mathcal{N}(\boldsymbol{S}_F))), \tag{9}$$

where $\mathcal{N}(\boldsymbol{S})$ is the normalizing function defined as $\mathcal{N}(\boldsymbol{S}) = (\boldsymbol{S} - \min(\boldsymbol{S}))/(\max(\boldsymbol{S}) - \min(\boldsymbol{S}))$. Since this loss minimizes the cosine similarity to be 0, we normalize the templates before similarity calculation.

Pair-wise set distribution loss. A template set would ensure that if the attacker is somehow able to get access to some of the templates, it would still be difficult to reverse engineer other templates. Therefore, we propose a constraint to minimize the inter-template cosine similarity to prompt the diversity of the templates in $\mathcal{S}$:

$$J_p = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \text{Cos}(\mathcal{N}(\boldsymbol{S}_i), \mathcal{N}(\boldsymbol{S}_j)). \tag{10}$$

The overall loss function for template estimation is thus:

$$J = \lambda_1 J_m + \lambda_2 J_r + \lambda_3 J_c + \lambda_4 J_s + \lambda_5 J_p, \tag{11}$$

where λ_1, λ_2, λ_3, λ_4, λ_5 are the loss weights for each term.

4. Experiments

4.1. Settings

Experimental setup and dataset. We follow the experimental setting of Wang *et al*. [48], and compare with four baselines: [48], [59], [10] and [28]. For training, [48] uses $720K$ images from which the manipulated images are generated by ProGAN [19]. However, as our method requires a GM to perform partial manipulation, we choose STGAN [21] in training as ProGAN synthesizes entire images. We use $24K$ images in CelebA-HQ [19] as the real images and pass them through STGAN to obtain manipulated images for training. For testing, we use 200 real images and pass them through *unseen* GMs such as StarGAN [7], GauGAN [31] and CycleGAN [61]. The real images for testing GMs are chosen from the respective dataset they are trained on, *i.e*. CelebA-HQ for StarGAN, Facades [61] for CycleGAN, and COCO [6] for GauGAN.

To further evaluate generalization ability of our approach, we use 12 additional unseen GMs that have diverse network architectures and loss functions, and are trained on different datasets. We manipulate each of 200 real images with these 12 GMs which gives $2,400$ manipulated images. The real images are chosen from the dataset that the respective GM is trained on. The list of GMs and their training datasets are provided in the supplementary.

Table 2. Performance comparison with prior works.

Method	Train GM	Set size	Test GM Average precision (%)		
			CycleGAN	StarGAN	GauGAN
[28]	CycleGAN	-	**100**	88.20	56.20
[10]	ProGAN	-	77.20	91.70	83.30
[59]	AutoGAN	-	**100**	**100**	61.00
[48]	ProGAN	-	84.00	**100**	67.00
Ours	STGAN	3	96.12	**100**	91.62
		20	99.66	**100**	90.58
	AutoGAN	3	97.87	97.89	86.57
		20	97.05	97.18	84.24
	STGAN + AutoGAN	3	**100**	**100**	**99.69**

Table 3. Performance comparison with Wang *et al*. [48].

Method	Train GM	Test GM TDR (%) at low FAR (0.5%)		
		CycleGAN	StarGAN	GauGAN
[48]	ProGAN	55.98	93.88	37.14
Ours	STGAN	**88.50**	**100.00**	**43.00**

Implementation details. Our framework is trained end-to-end for 10 epochs via Adam optimizer with a learning rate of 10^{-5} and a batch size of 4. The loss weights are set to ensure similar magnitudes at the beginning of training: $\lambda_1 = 100$, $\lambda_2 = 30$, $\lambda_3 = 5$, $\lambda_4 = 0.003$, $\lambda_5 = 10$. If not specified, we set the template set size $n = 3$. We set $k = 50$ in the content independent template loss. All experiments are conducted using one NVIDIA Tesla K80 GPU.

Evaluation metrics. We report average precision as adopted by [48]. To mimic real-world scenarios, we further report true detection rate (TDR) at a low false alarm rate (FAR) of 0.5%.

4.2. Image Manipulation Detection Results

As shown in Tab. 2, when our training GM is STGAN, we can outperform the baselines by a large margin on GauGAN-based test data, while the performance on StarGAN-based test data remains the same at 100%. When training on STGAN, our method achieves lower performance on CycleGAN. We hypothesis that it is because AutoGAN and CycleGAN share the same model architecture. To validate this, we change our training GM to AutoGAN and observe improvement when tested on CycleGAN. However, the performance drops on other two GMs because the amount of training data is reduced ($24K$ for STGAN and $1.5K$ for AutoGAN). Increasing the number of templates can improve the performance for when trained on STGAN and test on CycleGAN, but degrades for others. The degradation is more when train on AutoGAN. It suggests that it is challenging to find a larger template set on a smaller training set. Finally, using both STGAN and AutoGAN training

Table 4. Average precision of 12 testing GMs when our method is trained on only STGAN. All the GMs have different architectures and are trained on diverse datasets. The average precision of almost all GMs are over 90% showing the generalization ability of our method.

GM	UNIT [22]	MUNIT [15]	StarGAN2 [8]	BicycleGAN [62]	CONT_Enc. [32]	SEAN [63]	ALAE [33]	Pix2Pix [17]	DualGAN [55]	CouncilGAN [30]	ESRGAN [50]	GANimation [34]	Average
[48]	64.94	95.33	**100**	**100**	98.18	67.81	92.73	91.26	**98.91**	74.13	57.04	55.19	82.97
Ours	**100**	**100**	**100**	99.05	**98.75**	**97.63**	**93.10**	**92.50**	92.49	**89.71**	**87.30**	**58.69**	**92.43**

Table 5. Performance comparison of our proposed method with Ruiz *et al*. [40]. The performance for our proposed method is better than [40] when the testing GM is unseen. Both methods use StarGAN as the training GM.

Method	Test GM Average precision (%)			
	StarGAN	CycleGAN	GANimation	Pix2Pix
[40]	**100**	51.50	52.43	49.08
Ours	**100**	**95.26**	**60.12**	**91.85**

data can achieve the best performance.

TDR at low FAR. We also evaluate using TDR at low FAR in Tab. 3. This is more indicative of the performance in the real world application where the number of real images are exponentially larger than manipulated images. For comparison, we evaluate the pretrained model of [48] on our test set. Our method performs consistently better for all three GMs, demonstrating the superiority of our approach.

Generalization ability. To test our generalization ability, we perform extensive evaluations across a large set of GMs. We compare the performance of our method with [48] by evaluating its pretrained model on a test set of different GMs. Our framework performs quite well on almost all the GMs compared to [48] as shown in Tab. 4. This further demonstrates the generalization ability of our framework in the real world where an image can be manipulated by any unknown GM. Compared to [48], our framework achieves an improvement in the average precision of almost 10% averaged across all 12 GMs.

Comparison with proactive scheme work. We compare our work with previous work in proactive scheme [40]. As [40] proposes to disrupt the GM's output, they only provide the distortion results of the manipulated image. To enable binary classification, we take their adversarial real and disrupted fake images to train a classifier with the similar network architecture as our encoder. Tab. 5 shows that [40] works perfectly when the testing GM is the same as the training GM. Yet if the testing GM is unseen, the performance drops substantially. Our method performs much better showing the high generalizability.

Comparison with steganography works. Our method aligns with the high-level idea of digital steganographhy methods [4, 5, 42, 52, 63] which are used to hide an image onto other images. We compare our approach to the recent deep learning-based steganography method, Baluja *et al*. [4], with its publicly available code. We hide and retrieve the template using the pre-trained model provided by [4]. Our approach has far better average precision for each test GM compared to [4] as shown in Tab. 6. This validates the effectiveness of template learning and concludes that the digital steganography methods are less generalizable across unknown GMs than our approach.

Table 6. Performance comparison of our proposed method with steganography and adversarial attack methods.

Method	Type	Test GM Average precision (%)		
		CycleGAN	StarGAN	GauGAN
Baluja [4]	Steganography	85.64	88.06	81.26
PGD [25]	Adversarial	90.28	98.22	57.71
FGSM [14]	attack	89.21	98.29	63.81
Ours	-	**99.95**	**100**	**98.23**

Comparison with benign adversarial attacks. Adversarial attacks are used to optimize a perturbation to change the class of the image. The learning of the template using our framework is similar to a benign usage of adversarial attacks. We conduct an ablation study to compare our method with common attacks such as benign PGD and FGSM. We remove the losses in Eqs. 6, 8, and 10 responsible for learning the template and replace them with an adversarial noise constraint. Our approach has better average precision for each test GM than both adversarial attacks as shown in Tab. 6. We observe that adversarial noise performed similar to passive schemes offering poor generalization to unknown GMs. This shows the importance of using our proposed constraints to learn the universal template set.

Data augmentation. We apply various data augmentation schemes to evaluate the robustness of our method. We adopt some of the image editing techniques from Wang *et al*. [48], including (1) Gaussian blurring, (2) JPEG compression, (3) blur + JPEG (0.5), and (4) blur + JPEG (0.1), where 0.5 and 0.1 are the probabilities of applying these image editing operations. In addition, we add resizing, cropping, and Gaussian noise. The implementation details of these techniques are in the supplementary. These techniques are applied after addition of our template to the real images.

We evaluate in three scenarios when augmentation is applied in (1) training, (2) testing, (3) both training and testing. As shown in Tab. 7, for the augmentation techniques adopted from [48], we outperform [48] in almost all techniques. We observe significant improvement when blurring or JPEG compression is applied jointly but the improvement is less when they are applied separately.

Table 7. Average precision (%) with various augmentation techniques in training and testing for three GMs. We apply data augmentation to three scenarios: (1) in training only (2) in testing only and (3) in both training and testing. [Keys: aug.=augmentation, B.=blur, J.=JPEG compression, Gau. No.=Gaussian Noise]

Augmentation		Augmentation type	Method	Test GMs		
Train	Test			CycleGAN	StarGAN	GauGAN
✗	✗	No augmentation	[48]	84.00	**100**	67.00
			Ours	96.12	**100**	91.62
✔	✗	Blur	[48]	90.10	**100**	74.70
			Ours	93.55	**100**	92.35
		JPEG	[48]	93.20	91.80	97.50
			Ours	98.74	98.30	91.85
		B+J (0.5)	[48]	96.80	95.40	98.10
			Ours	94.44	**100**	98.16
		B+J (0.1)	[48]	93.50	84.50	89.50
			Ours	95.79	**100**	95.94
		Resizing	Ours	**100**	**100**	98.97
		Crop		84.45	84.92	94.43
		Gau. No.		99.95	**100**	**99.11**
✗	✔	Blur	Ours	95.74	84.87	70.74
		JPEG		91.91	82.96	84.16
		B+J (0.5)		89.23	82.18	75.53
		Resizing		93.12	77.41	91.45
		Crop		84.04	73.87	70.12
		Gau. No.		73.83	69.47	66.70
✔	✔	Blur	Ours	92.16	**100**	90.15
		JPEG		94.00	97.92	85.91
		B+J (0.5)		87.37	84.92	74.68
		Resizing		99.98	**100**	92.73
		Cropping		77.63	89.22	79.96
		Gau. No.		97.44	**100**	82.32

As for the different scenarios on when data augmentation is applied, scenario 2 performs the worst because the augmentation applied in testing has not been seen during training. Scenario 3 performs better than scenario 2 in most cases. There is a much larger performance drop when blurring and JPEG are applied together than separately. Cropping performs the worst for both Scenario 1 and 3.

4.3. Ablation Studies

Template set size. We study the effects of the template set size. As shown in Fig. 4, the average precision increases as the set size is expanding from 1 and saturates around the set size 10. In the meantime, the average cosine similarity between templates within the set increases consistently, as it gets harder to find many orthogonal templates. We also test our framework's run-time for different set sizes. On a Tesla K80 GPU, for the set size of $1, 3, 10, 20$ and 50, the per-image run-time of our manipulation detection is 26.19, 27.16, 28.44, 34.26, and 43.76 ms respectively. Thus, despite increasing the set size enhances our accuracy and security, there is a trade-off with the detection speed which is a important factor too. For comparison, we also test the pre-trained model of [48] which gives a per-image run-time of 54.55 ms. Our framework is much faster even with a larger set size which is due to the shallow network in our proactive scheme compared to a deeper network in passive scheme.

Template strength. We use a hyperparameter m to control the strength of our added template. We ablate m and show the results in Fig. 5. Intuitively, the lower the strength of the template added, the lower the detection performance since it would be harder for the encoder to recover the original template. Our results support this intuition. For all three GMs, the precision increases as we enlarge the template strength, and converges after 50% strength. We also show the PSNR between the encrypted real image and the original real image. The PSNR decreases as we enlarge the strength as expected. We choose $m = 30\%$ for a trade-off between the detection precision and the visual quality.

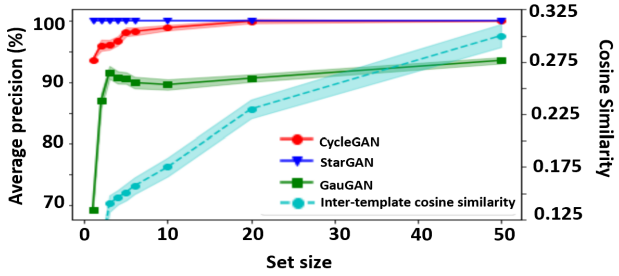

Figure 4. Ablation study with varying template set sizes. The performance improves when the set size increases, while the inter-template cosine similarity also increases.

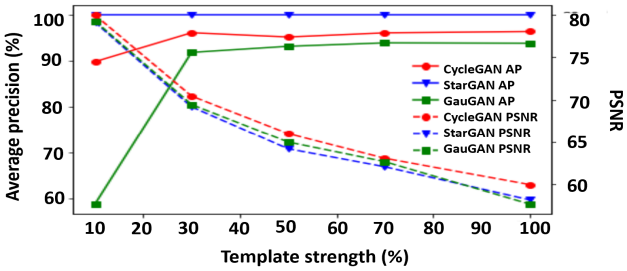

Figure 5. Ablation with varying template strengths in the encrypted real images. The lower the template strength, the higher the PSNR is and the harder it is for our encoder to recover it, which leads to lower detection performance.

Loss functions. Our training process is guided by an objective function with five losses (Eqn. 11). To demonstrate the necessity of each loss, we ablate by removing each loss and compare with our full model. As shown in Tab. 8, removing any one of the losses results in performance degradation. Specifically, removing the pair-wise set distribution loss, recovery loss or separation loss causes a larger drop.

To better understand the importance of the data-driven template set, we fix the template set during training, *i.e.*, removing the three losses directly operating on the template and only considering recovery and separation losses for training. We observe a significant performance drop, which shows that the learnable template is indeed crucial for effective image manipulation detection.

Finally, we remove the encoder from our framework and

Table 8. Ablation study to remove losses used in our training. Removing any one loss deteriorates the performance compared to our proposed method. Fixing the template or performing direct classification made the results worse. This shows the importance of a variable template and using an encoder for classification purposes.

Loss removed	Test GM Average precision (%)		
	CycleGAN	StarGAN	GauGAN
Magnitude loss (J_m)	94.43	**100**	87.44
Pair-wise set distribution loss (J_p)	66.60	79.99	74.55
Recovery loss (J_r)	51.59	94.18	90.61
Content independent template loss (J_c)	92.01	**100**	80.54
Separation loss (J_s)	92.24	**100**	64.06
J_m, J_p and J_c (fixed template)	46.93	59.88	43.64
J_r and J_s (removing encoder)	50.00	98.24	55.00
None (ours)	**96.12**	**100**	**91.62**

use a classification network with similar number of layers. Instead of recovering templates, the classification network is directly trained to perform binary image manipulation detection via cross-entropy loss. The performance drops significantly. This observation aligns with the previous works [10,47,59] stating that CNN networks trained on images from one GM show poor generalizability to unseen GMs. The performance drops for all three GMs but CycleGAN and GauGAN are affected the most, as the datasets are different. For our proposed approach, when we are recovering the template, the encoder ignores all the low frequency information of the images which are data dependent. Thus, being more data (*i.e.*, image content) independent, our encoder is able to achieve a higher generalizability.

Template selection. Given a real image, we randomly select a template from the learnt template set to add to the image. Thus, every image has an equal chance of selecting any one template from the set, resulting in many combinations for the entire test set. This raises the question of finding a worst and best combination of templates for all images in the test set. To answer this, we experiment with a template set size of 50 as a large size may offer higher variation in performance. For each image in $\mathcal{T}(\boldsymbol{X}^a)$ and $G(\mathcal{T}(\boldsymbol{X}^a))$, we calculate the cosine similarity between added template $\boldsymbol{S}$ and recovered template $\boldsymbol{S}_{R/F}$. For the worst/best case of every image, we select the template with the minimum/maximum difference between the real and manipulated image cosine similarities. As shown in Tab. 9, GauGAN gives much more variation in the performance compared to CycleGAN and StarGAN. This shows that the template selection is an important step for image manipulation detection. This brings up the idea of training a network to select the best template for a specific image, by using the best case described above as a pseudo ground truth to supervise the network. We hypothesis template selection could be important, but with experiments, the difference of performance among different templates is nearly zero and the network's selection doesn't help in the performance compared with selecting the template randomly as shown in Tab. 9. Therefore, we cannot have a pseudo ground truth to train another network for template selection.

Table 9. Ablation of template selection schemes at set size of 50.

Selection scheme	Test GM Average precision (%)		
	CycleGAN	StarGAN	GauGAN
Random selection	99.90 ± 0.02	$\mathbf{100 \pm 0.00}$	93.56 ± 0.52
Biasing one template	99.05 ± 0.37	$\mathbf{100 \pm 0.00}$	91.21 ± 0.97
Network based	95.46	**100**	90.47
Worst case	94.85	**100**	80.55
Best case	**99.95**	**100**	**98.23**

Another option for template selection is to select the *same* template for every test image which is equivalent to using one template compromising the security of our method. Nevertheless, we test this option to see the performance variation of biasing one template for all images. The performance variation is larger than our random selection scheme. This shows that each template has a similar contribution to image manipulation detection.

5. Conclusion

In this paper, we propose a proactive scheme for image manipulation detection. The main objective is to estimate a set of templates, which when added to the real images improves the performance for image manipulation detection. This template set is estimated using certain constraints and any template can be added onto the image right after it is being captured by any camera. Our framework is able to achieve better image manipulation detection performance on different unseen GMs, compared to prior works. We also show the results on a diverse set of 12 additional GMs to demonstrate the generalizability of our proposed method.

Limitations. First, although our work aims to protect real images in a proactive manner and can detect whether an image has been manipulated or not, it cannot perform general deepfake detection on entirely synthesized images. Second, we try our best to collect a diverse set of GMs to validate the generalization of our approach. However, there are many other GMs that do not have open-sourced codes to be evaluated in our framework. Lastly, how to supervise the training of a network for template selection is still an unanswered question.

Potential societal impact. We propose a proactive scheme which uses encrypted real images and their manipulated versions to perform manipulation detection. While this offers more generalizable detection, the encrypted real images might be used for training GMs in the future, which could make the manipulated images more robust against our framework, and thus warrents more research.

References

[1] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. MesoNet: a compact facial video forgery detection network. In *WIFS*, 2018.

[2] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, 2017.

[3] Vishal Asnani, Xi Yin, Tal Hassner, and Xiaoming Liu. Reverse engineering of generative models: Inferring model hyperparameters from generated images. *arXiv preprint arXiv:2106.07873*, 2021. 2

[4] Shumeet Baluja. Hiding images in plain sight: Deep steganography. 2017. 3, 6

[5] Abdullah Bamatraf, Rosziati Ibrahim, and Mohd Najib B Mohd Salleh. Digital watermarking algorithm using LSB. In *ICCAIE*, 2010. 3, 6

[6] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *CVPR*, 2018. 5, 12

[7] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018. 1, 5, 12

[8] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse image synthesis for multiple domains. In *CVPR*, 2020. 1, 6, 12

[9] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 2

[10] Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Riess, Matthias Nießner, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510*, 2018. 2, 5, 8

[11] Hao Dang, Feng Liu, Joel Stehouwer, Xiaoming Liu, and Anil K Jain. On the detection of digital face manipulation. In *CVPR*, 2020. 1, 2

[12] Debayan Deb, Xiaoming Liu, and Anil Jain. Unified detection of digital and physical face attacks. In *arXiv preprint arXiv:2104.02156*, 2021. 1

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 1

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015. 6

[15] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018. 6, 12

[16] Zhi-Jing Huang, Shan Cheng, Li-Hua Gong, and Nan-Run Zhou. Nonlinear optical multi-image encryption scheme with two-dimensional linear canonical transform. *Optics and Lasers in Engineering*, 124:105821, 2020.

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 6, 12

[18] Mei Jiansheng, Li Sukang, and Tan Xiaomei. A digital watermarking algorithm based on DCT and DWT. In *WISA*, 2009. 3

[19] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018. 5, 12

[20] Mohammad Ibrahim Khan, Md Maklachur Rahman, and Md Iqbal Hasan Sarker. Digital watermarking for image authentication based on combined DCT, DWT and SVD transformation. *International Journal of Computer Science Issues*, 10:223, 2013. 3

[21] Ming Liu, Yukang Ding, Min Xia, Xiao Liu, Errui Ding, Wangmeng Zuo, and Shilei Wen. STGAN: A unified selective transfer network for arbitrary image attribute editing. In *CVPR*, 2019. 1, 5, 12

[22] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NeurIPS*, 2017. 6, 12

[23] Xiaohong Liu, Yaojie Liu, Jun Chen, and Xiaoming Liu. PSCC-Net: Progressive spatio-channel correlation network for image manipulation detection and localization. In *arXiv preprint arXiv:2103.10596*, 2021. 1

[24] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 12

[25] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 6

[26] Iacopo Masi, Aditya Killekar, Royston Marian Mascarenhas, Shenoy Pratik Gurudatt, and Wael AbdAlmageed. Two-branch recurrent network for isolating deepfakes in videos. In *ECCV*, 2020. 2

[27] Safa C. Medin, Bernhard Egger, Anoop Cherian, Ye Wang, Joshua B. Tenenbaum, Xiaoming Liu, and Tim K. Marks. MOST-GAN: 3D morphable StyleGAN for disentangled face image manipulation. In *AAAI*, 2022. 1

[28] Lakshmanan Nataraj, Tajuddin Manhar Mohammed, BS Manjunath, Shivkumar Chandrasekaran, Arjuna Flenner, Jawadul H Bappy, and Amit K Roy-Chowdhury. Detecting GAN generated fake images using co-occurrence matrices. *Electronic Imaging*, 2019:532–1, 2019. 2, 5

[29] Yuval Nirkin, Lior Wolf, Yosi Keller, and Tal Hassner. Deepfake detection based on discrepancies between faces and their context. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 2021. 2

[30] Ori Nizan and Ayellet Tal. Breaking the cycle - colleagues are all you need. In *CVPR*, 2020. 6, 12, 13

[31] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. GauGAN: semantic image synthesis with spatially adaptive normalization. In *ACM*, 2019. 1, 5, 12

[32] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 6, 12

[33] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *CVPR*, 2020. 6, 12

[34] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. GANimation: One-shot anatomically consistent facial animation. *International Journal of Computer Vision*, 128:698–713, 2020. 6, 12

[35] Yuyang Qian, Guojun Yin, Lu Sheng, Zixuan Chen, and Jing Shao. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In *ECCV*, 2020. 2

[36] Zhang Qiu-yu, Jitian Han, and Yutong Ye. Multi-image encryption algorithm based on image hash, bit-plane decomposition and dynamic DNA coding. *IET Image Processing*, 15:885–896, 2020.

[37] Weize Quan, Kai Wang, Dong-Ming Yan, and Xiaopeng Zhang. Distinguishing between natural and computer-generated images using convolutional neural networks. *IEEE Transactions on Information Forensics and Security*, 13:2772–2787, 2018.

[38] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 12

[39] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *CVPR*, 2019. 1, 2

[40] Nataniel Ruiz, Sarah Adel Bargal, and Stan Sclaroff. Disrupting deepfakes: Adversarial attacks against conditional image translation networks and facial manipulation systems. In *ECCV*, 2020. 1, 2, 6

[41] Eran Segalis and Eran Galili. OGAN: Disrupting deepfakes with an adversarial attack that survives training. *arXiv preprint arXiv:2006.12247*, 2020. 1, 2

[42] Amit Kumar Singh, Nomit Sharma, Mayank Dave, and Anand Mohan. A novel technique for digital image watermarking in spatial domain. In *PDGC*, 2012. 3, 6

[43] Matthew Tancik, Ben Mildenhall, and Ren Ng. StegaStamp: Invisible hyperlinks in physical photographs. In *CVPR*, 2020. 3

[44] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning GAN for pose-invariant face recognition. In *CVPR*, 2017. 1

[45] Radim Tyleček and Radim Šára. Spatial pattern templates for recognition of objects with regular structure. In *GCPR*, 2013. 12

[46] Run Wang, Felix Juefei-Xu, Meng Luo, Yang Liu, and Lina Wang. FakeTagger: Robust safeguards against deepfake dissemination via provenance tracking. In *ACMM*, 2021. 1, 2

[47] Run Wang, Felix Juefei-Xu, Lei Ma, Xiaofei Xie, Yihao Huang, Jian Wang, and Yang Liu. FakeSpotter: A simple yet robust baseline for spotting ai-synthesized fake faces. In *IJCAI*, 2020. 8

[48] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. CNN-generated images are surprisingly easy to spot... for now. In *CVPR*, 2020. 1, 2, 5, 6, 7

[49] Xiaogang Wang and Xiaoou Tang. Face photo-sketch synthesis and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31:1955–1967, 2008. 12

[50] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data. In *CVPR*, 2021. 6, 12

[51] Xi Wu, Zhen Xie, YuTao Gao, and Yu Xiao. SSTNET: Detecting manipulated faces through spatial, steganalysis and temporal features. In *ICASSP*, 2020. 1, 2

[52] Erkan Yavuz and Ziya Telatar. Improved SVD-DWT based digital image watermarking against watermark ambiguity. In *SAC*, 2007. 3, 6

[53] Huo-Sheng Ye, Nan-Run Zhou, and Li-Hua Gong. Multi-image compression-encryption scheme based on quaternion discrete fractional hartley transform and improved pixel adaptive diffusion. *Signal Processing*, 175:107652, 2020.

[54] Chin-Yuan Yeh, Hsi-Wen Chen, Shang-Lun Tsai, and Sheng-De Wang. Disrupting image-translation-based deepfake algorithms with adversarial attacks. In *WACVW*, 2020. 1, 2

[55] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. DualGAN: Unsupervised dual learning for image-to-image translation. In *CVPR*, 2017. 6, 12

[56] A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *CVPR*, 2014. 12

[57] A. Yu and K. Grauman. Semantic jitter: Dense supervision for visual comparisons via synthetic images. In *ICCV*, 2017. 12

[58] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[59] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in GAN fake images. In *WIFS*, 2019. 2, 5, 8

[60] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. HiDDeN: Hiding data with deep networks. In *ECCV*, 2018. 3

[61] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 1, 5, 12

[62] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NeurIPS*, 2017. 6, 12

[63] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. SEAN: Image synthesis with semantic region-adaptive normalization. In *CVPR*, 2020. 6, 12

Proactive Image Manipulation Detection
– Supplementary material –

1. Cross Encoder-Template Set Evaluation

Our framework encrypts a real image using a template from the template set. This encryption would aid in the image manipulation detection if the image is corrupted by any unseen GM. The framework is divided in two stages namely, image encryption and recovery of template where each stage works independently in inference. We therefore provide an ablation to study the performance using different encoder and template set, *i.e.*, we evaluate recovering ability of an encoder using a template set trained with different initialization seeds. The results are shown in Tab. 1. We observe that even though the template set and the encoder are initialized with different seeds, the performance of our framework doesn't vary much. This shows the stability of our framework even though the initialization seeds of both stages during training are different.

Table 1. Cross encoder-template set evaluation with different initialization seeds.

Initialization seed		Test GM Average precision (%)		
Encoder	Template set	StarGAN	CycleGAN	GauGAN
1	1	96.12	100	91.62
	2	94.65	100	91.15
	3	94.83	100	91.46
2	1	95.48	100	91.56
	2	95.54	100	90.85
	3	95.84	100	91.06
3	1	95.56	100	91.32
	2	95.62	100	91.42
	3	96.14	100	90.41

2. Template Strength

We provide the ablation for hyperparameter m used to control the strength of the added template in Sec. 4.3. We observe that the performance is better if we increase the template strength. However, this comes at a trade-off with PSNR which declines if the template strength increases. This is also justified in Fig. 1 which shows the images with different strength of added template. The images become noisier as the template strength is increased. This is not desirable as there shouldn't be much distortion in the encrypted real image due to our added template. Therefore for our experiments, we select 30% as the strength for the added template.

3. Implementation Details

Image editing techniques We use various image editing techniques in Sec. 4.2. All the techniques are applied after addition of our template. We provide the implementation details for all these techniques below:

1. Blur: We apply Gaussian blur to the image with 50% probability using σ sampled from $[0, 3]$,

2. JPEG: We JPEG-compress the image with 50% probability images using Imaging Library (PIL), with quality sampled from Uniform$\{30, 31, ..., 100\}$.

3. Blur + JPEG (p): The image is possibly blurred and JPEG-compressed, each with probability p.

4. Resizing: We perform the training using 50% of the images with $256 \times 256 \times 3$ resolution and rest with $128 \times 128 \times 3$ resolution images in CelebA-HQ dataset.

5. Crop: We randomly crop the images with 50% probability on each side with pixels sampled from $[0, 30]$. The images are resized to $128 \times 128 \times 3$ resolution.

6. Gaussian noise: We add Gaussian noise with zero mean and unit variance to the images with 50% probability.

Network architecture Fig. 2 shows the network architecture used in different experiments for our framework's evaluation. For our framework, our encoder has 2 stem convolution layers and 10 convolution blocks to recover the added template from encrypted real images. Each block comprises of convolution, batch normalization and ReLU activation.

In ablation experiments for Table 8, we use a classification network with the similar number of layers as our encoder. This is done to show the importance of recovering templates using encoder. This classification networks has 8 convolution blocks followed by three fully connected layers with ReLU activation in between the layers. The network outputs 2 dimension logits used for image manipulation detection.

4. List of GMs

We use a variety of GMs to test the generalization ability of our framework. These GMs have varied network architectures and many of them are trained on different datasets. We summarize all the GMs in Tab. 2. We also provide visualization for different real image samples used in evaluating the performance for all these GMs in Fig. 3 - 18. We show the added template and the recovered templates

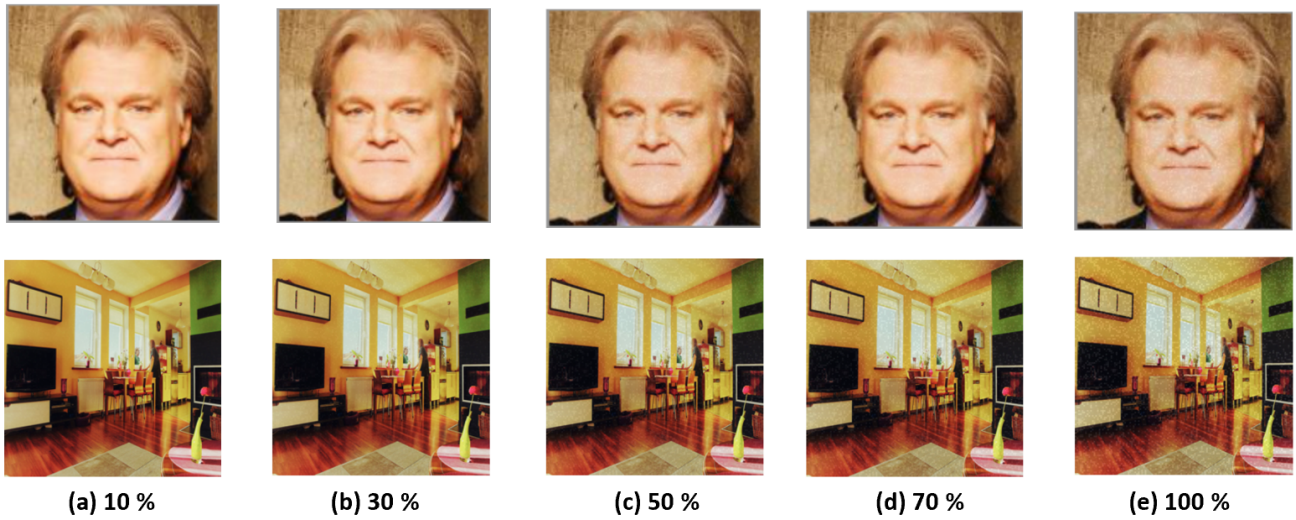

Figure 1. Visualization of input images with different template strength. As the template strength is increased, the images become noisier.

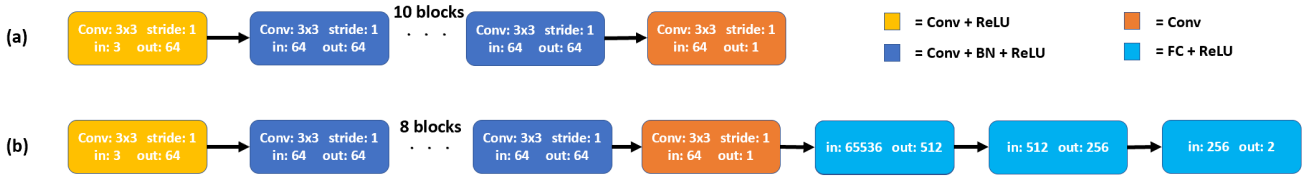

Figure 2. Network architecture for our (a) encoder (b) classifier network for image manipulation detection.

Table 2. List of GMs with their datasets and input image resolution used for evaluating our framework's generalization ability.

GM	STGAN [21]	StarGAN [7]	CycleGAN [61]	GauGAN [31]	UNIT [22]	MUNIT [15]	StarGAN2 [8]	BicycleGAN [62]
Dataset	CelebA-HQ [19]	CelebA-HQ [19]	Facades [45]	COCO [6]	GTA2City [38]	Edges2Shoes [56, 57]	CelebA-HQ [19]	Facades [45]
Resolution	$128 \times 128 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$512 \times 931 \times 3$	$256 \times 512 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$
GM	CONT_Encoder [32]	SEAN [63]	ALAE [33]	Pix2Pix [17]	DualGAN [55]	CouncilGAN [30]	ESRGAN [50]	GANimation [34]
Dataset	Paris Street-View [30]	CelebA-HQ [19]	CelebA-HQ [19]	Facades [45]	Sketch-Photo [49]	CelebA [24]	CelebA [24]	CelebA [24]
Resolution	$64 \times 64 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$128 \times 128 \times 3$	$128 \times 128 \times 3$

in "gist_rainbow" cmap for better visualization and indicate the cosine similarity of the recovered template with the added template. As shown in Fig. 3 for training with STGAN, the encrypted real images have higher cosine similarity compared to their manipulated counterparts. However, during testing, the difference between the two cosine similarities decreases as shown in Fig. 4 - 18 for different GMs.

5. Dataset License Information

We use diverse datasets for our experiments which include face and non-face datasets. For face datasets, we use existing datasets including CelebA [24] and CelebA-HQ [19]. The CelebA dataset contains images entirely from the internet and has no associated IRB approval. The authors mention that the dataset is available for non-commercial research purposes only, which we strictly adhere to. We only use the database internally for our work and primarily for evaluation. CelebA-HQ consists images collected from the internet. Although there is no associated IRB approval, the authors assert in the dataset agreement that the dataset is only to be used for non-commercial research purposes, which we strictly adhere to.

We use some non-face datasets too for our experiments. The Facades [45] dataset was collected at the Center for Machine Perception and is provided under Attribution-ShareAlike license. Edges2Shoes [56, 57] is a large shoe dataset consisting of images collected from `https://www.zappos.com`. The authors mention that this dataset is for academic, non-commercial use only. GTA2City [38] dataset consists of a large number of densely labelled frames extracted from computer games. The authors mention that the data is for research and educational use only. The sketch-photo [49] datset refers to the CUHK face sketch FERET database. The authors assert in the dataset agreement that the dataset is only to be used for noncom-

mercial research purposes, which we strictly adhere to. Paris street-view [30] dataset contains images collected using google street view and is to be used for noncommercial research purposes.

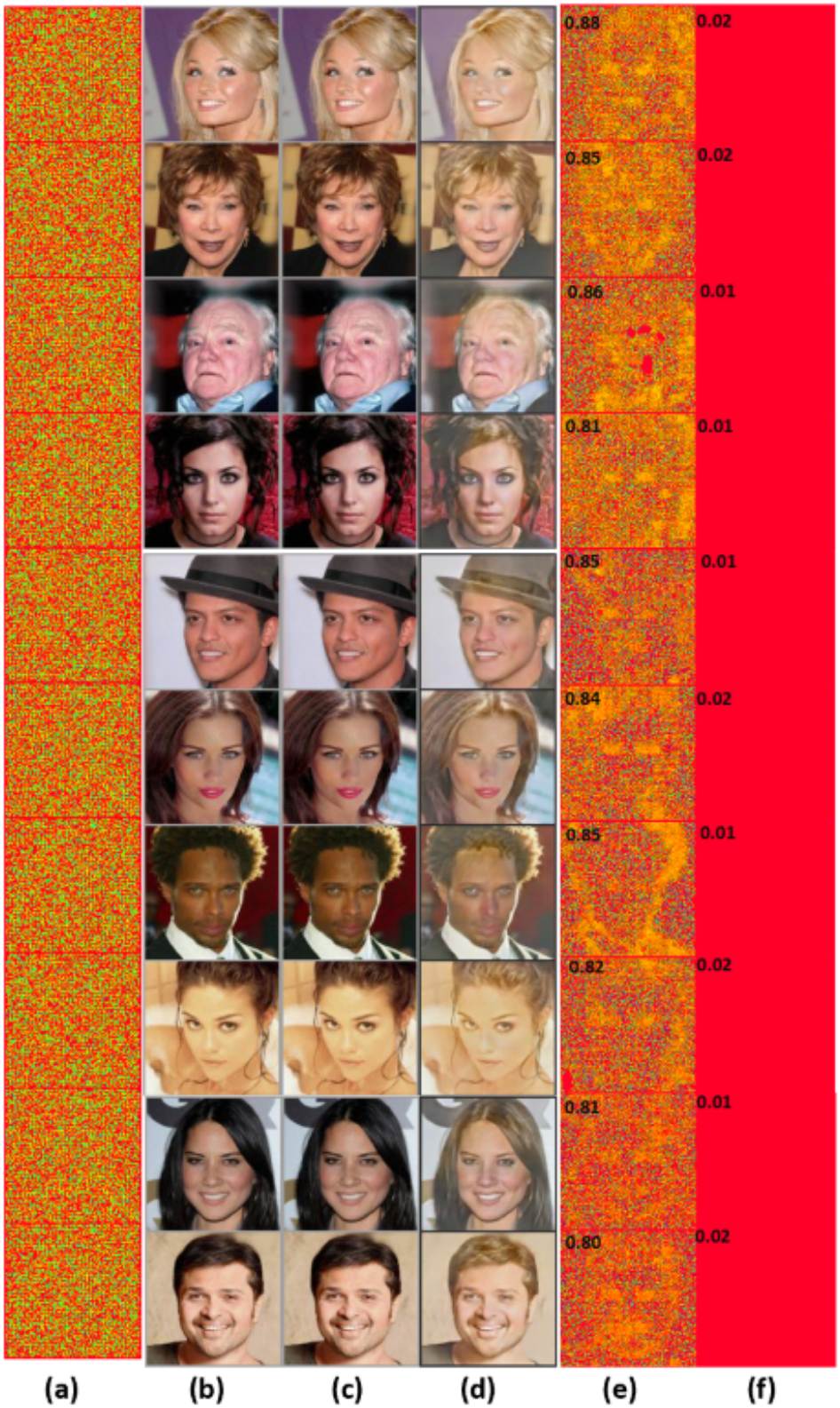

Figure 3. Visualization of samples used for GM STGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

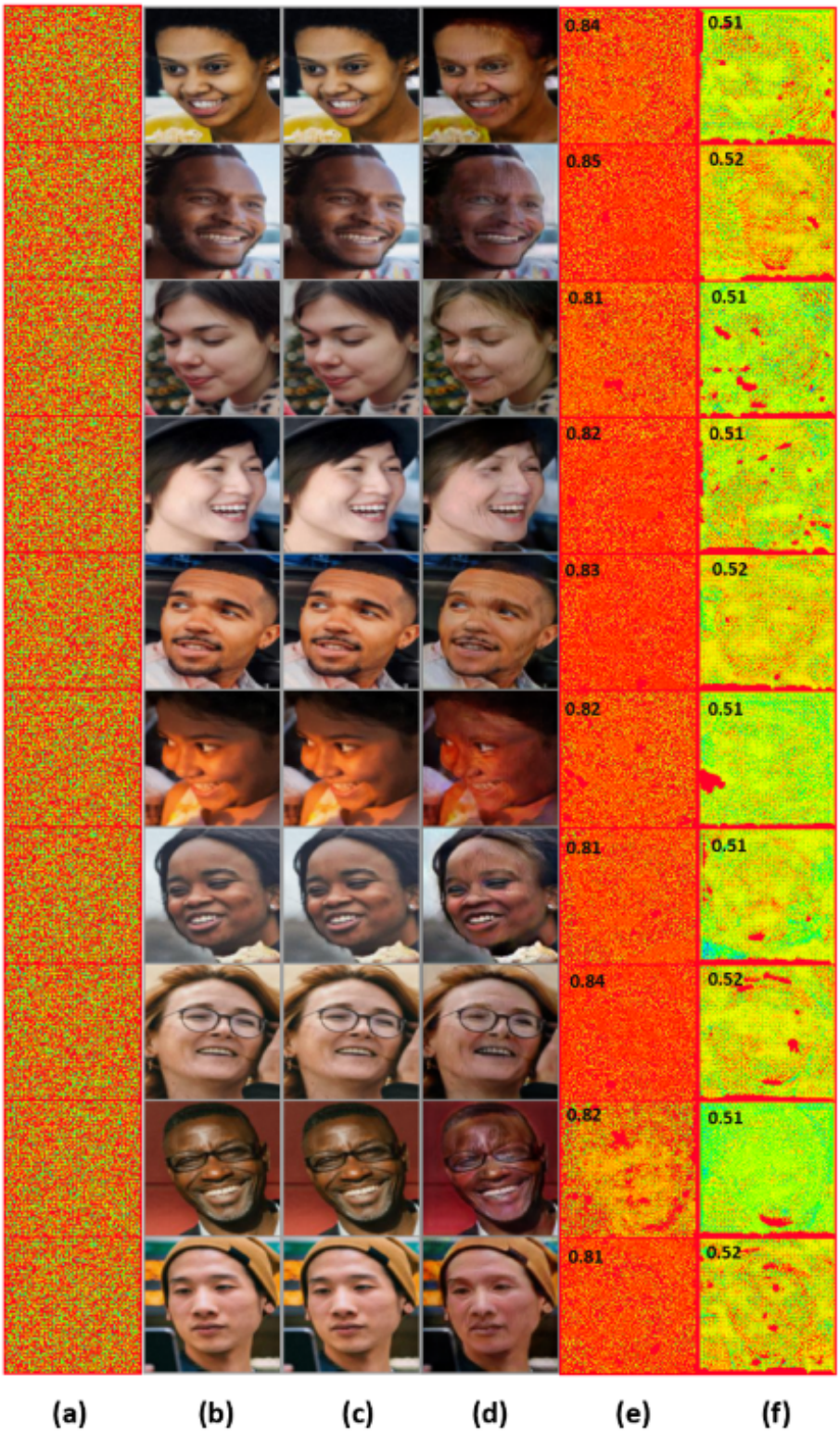

Figure 4. Visualization of samples used for GM StarGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

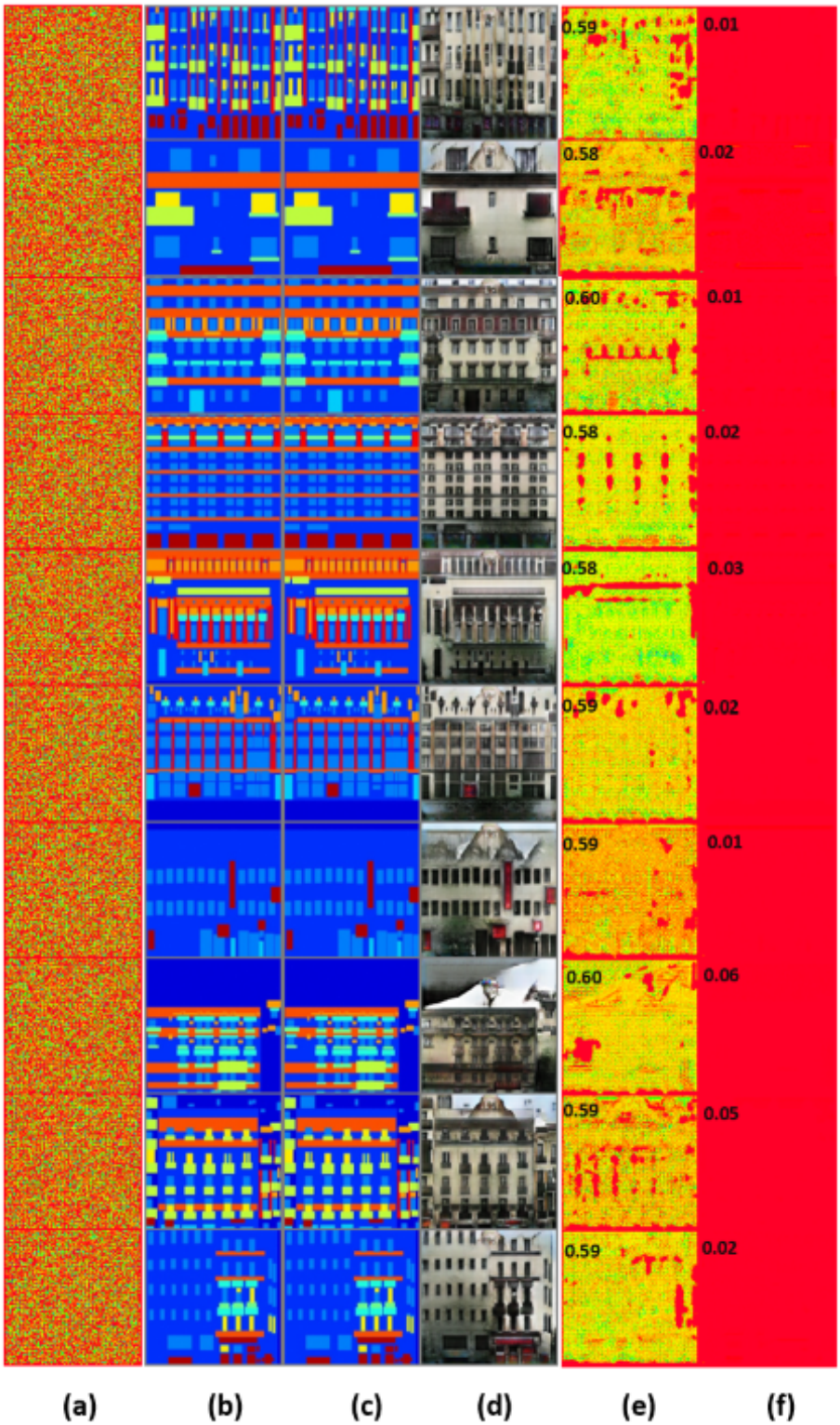

Figure 5. Visualization of samples used for GM CycleGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

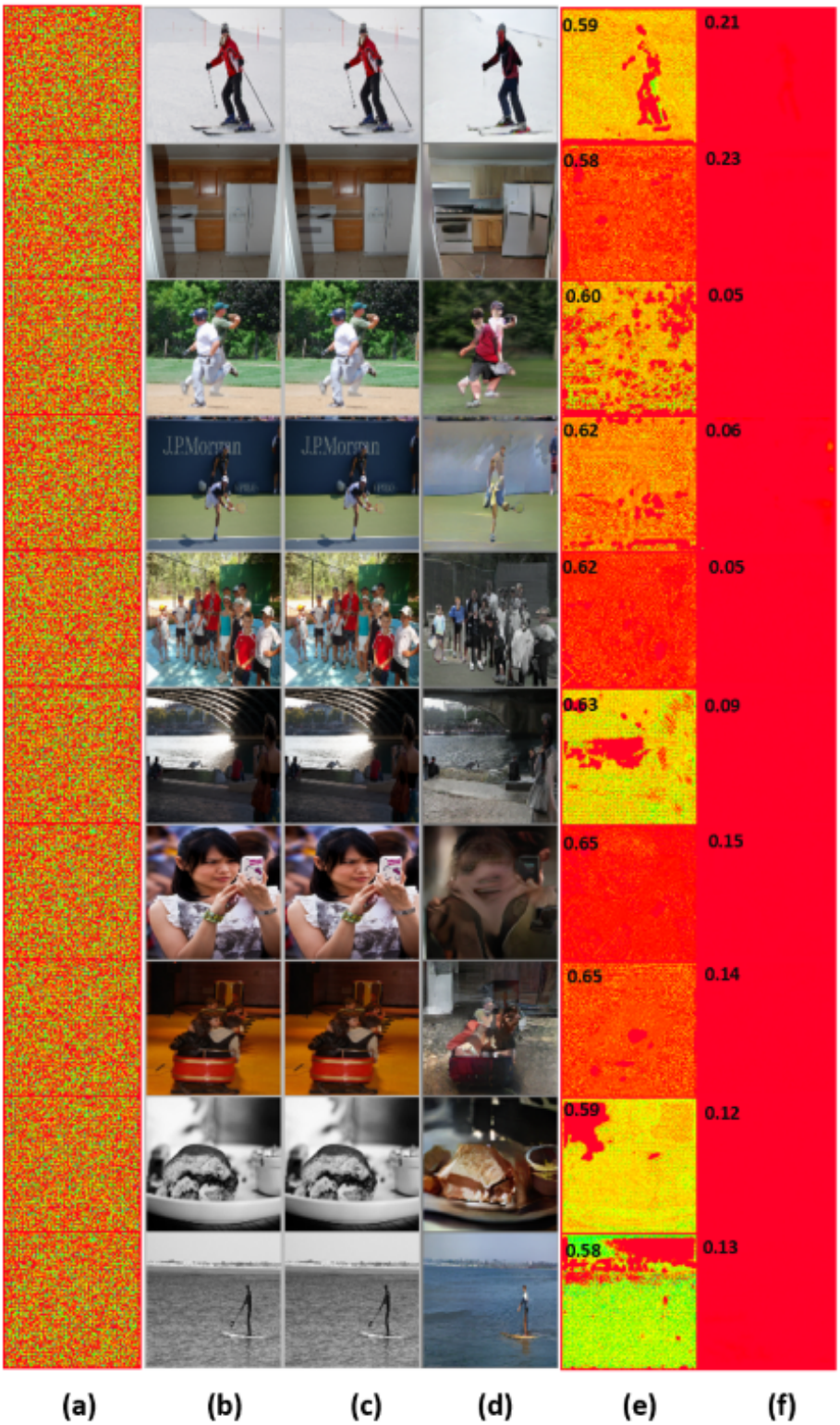

Figure 6. Visualization of samples used for GM GauGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

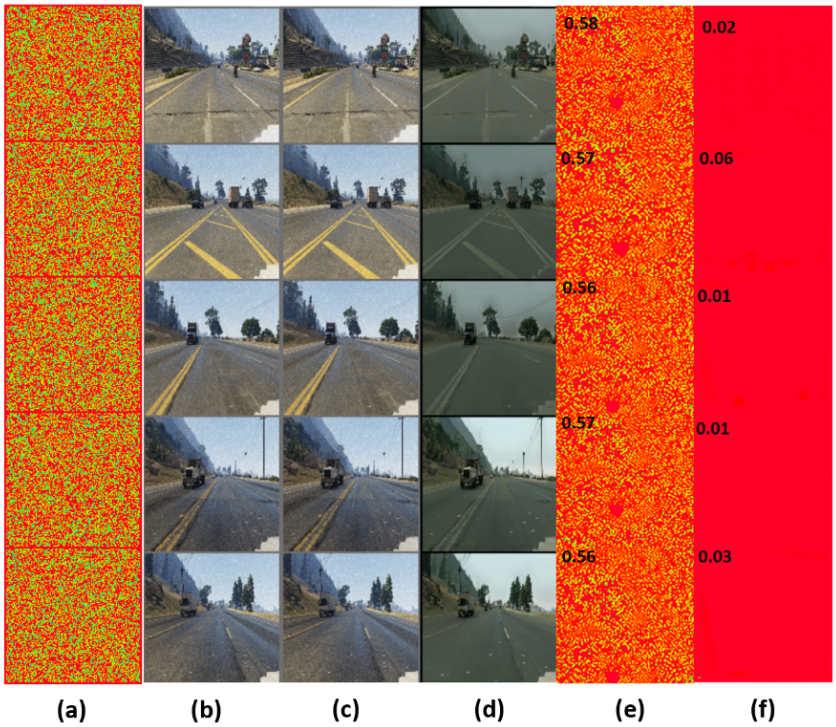

Figure 7. Visualization of samples used for GM UNIT; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

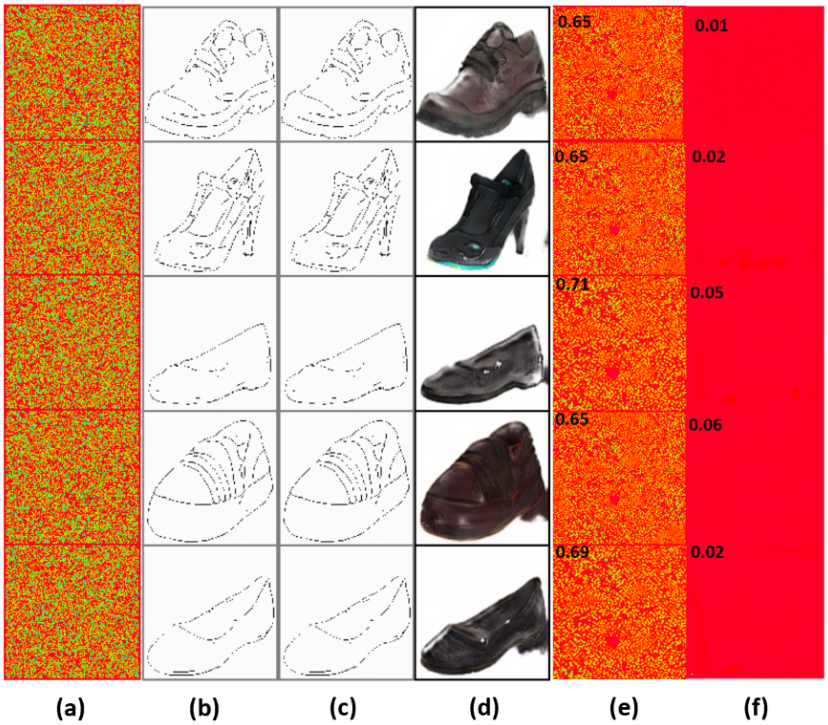

Figure 8. Visualization of samples used for GM MUNIT; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

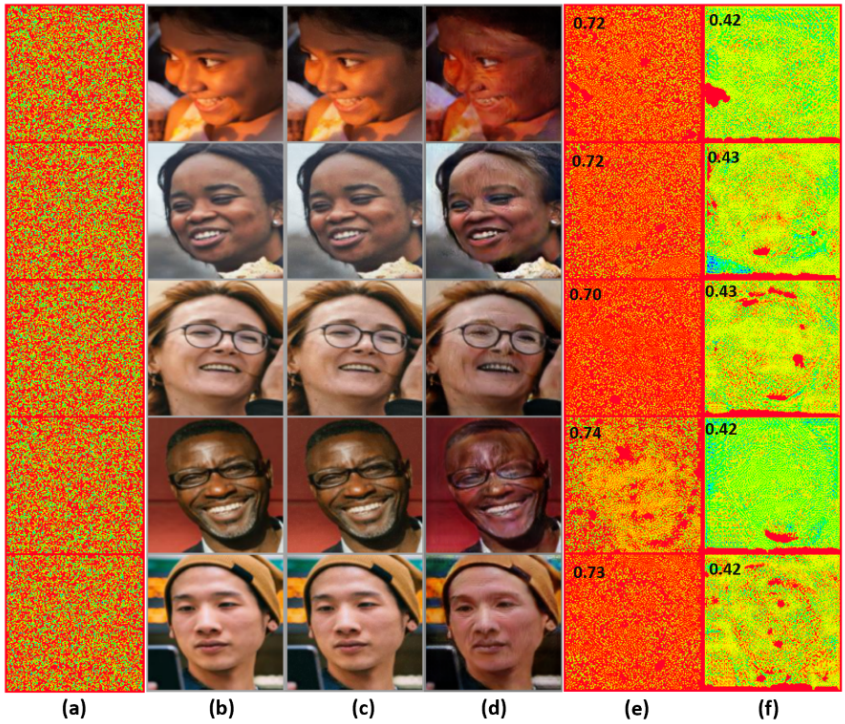

Figure 9. Visualization of samples used for GM StarGANv2; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

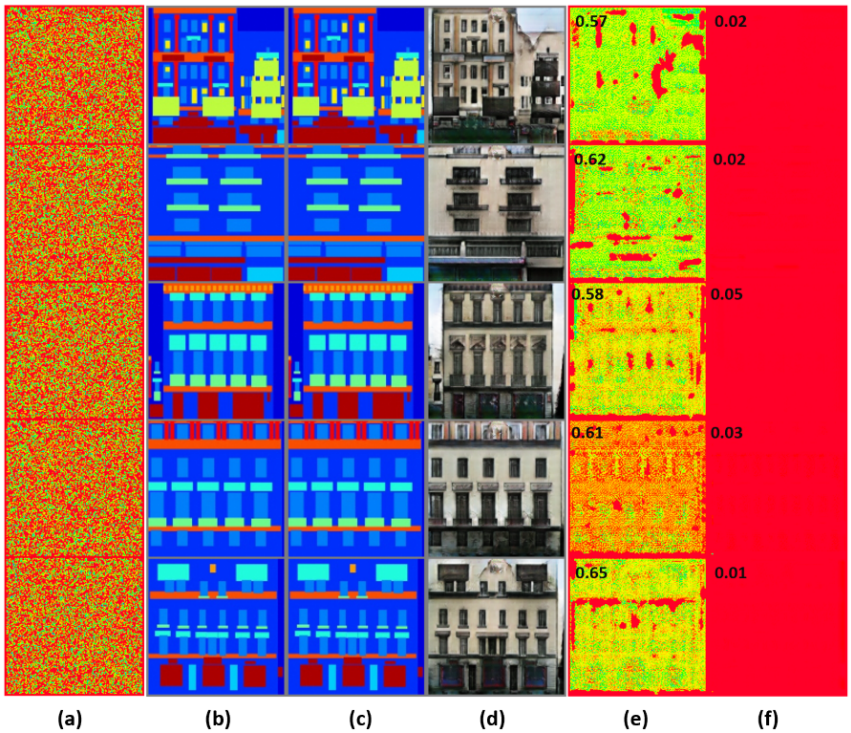

Figure 10. Visualization of samples used for GM BicycleGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

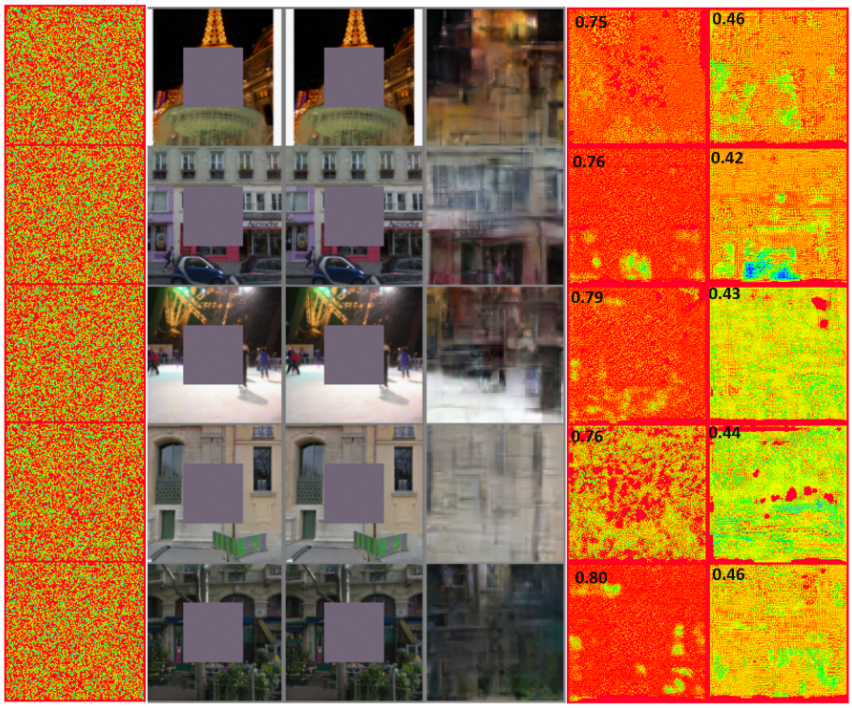

Figure 11. Visualization of samples used for GM CONT_Encoder; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

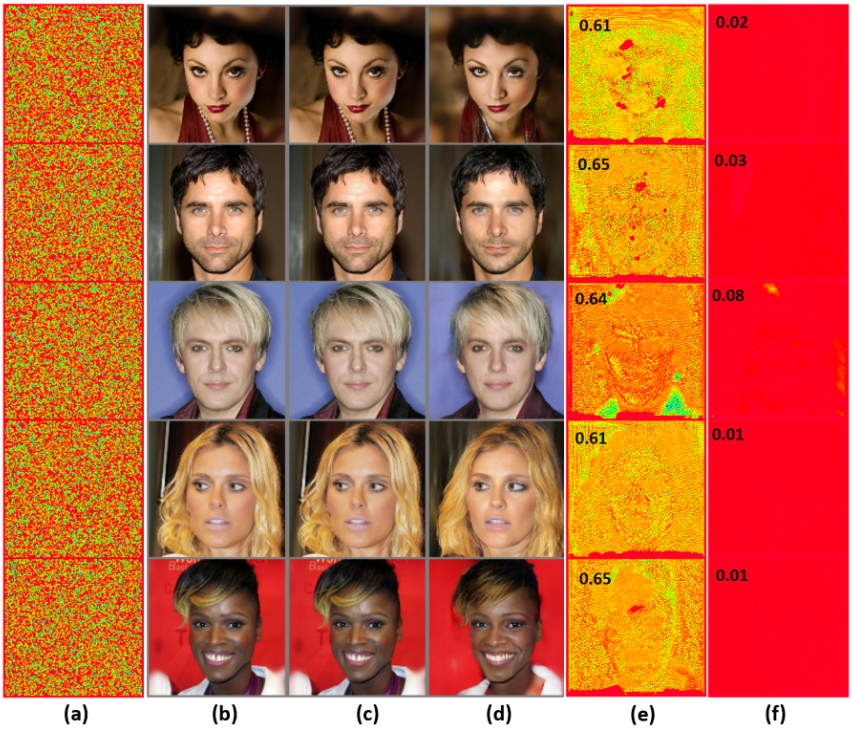

Figure 12. Visualization of samples used for GM SEAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

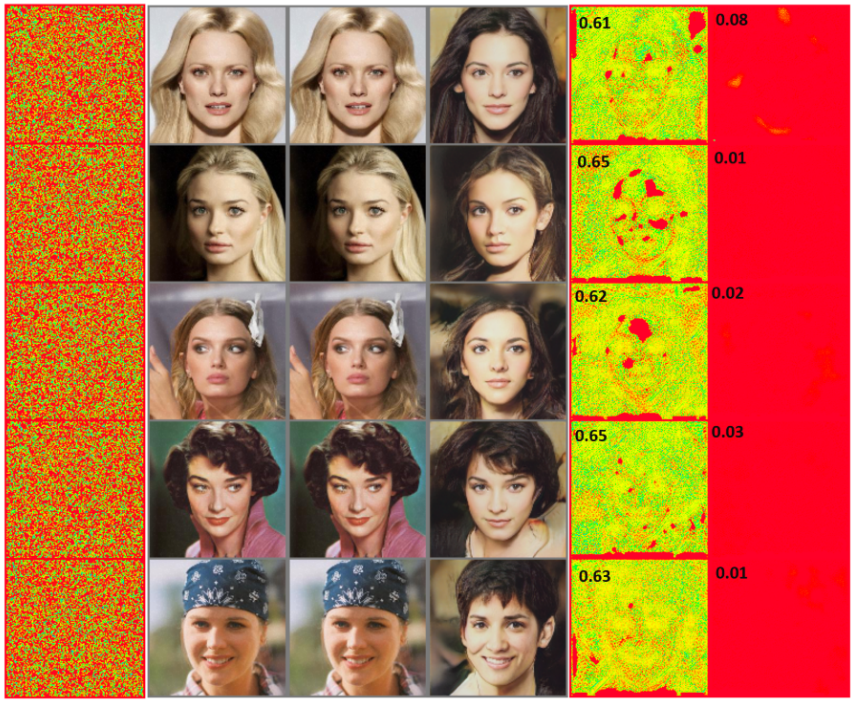

Figure 13. Visualization of samples used for GM ALAE; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

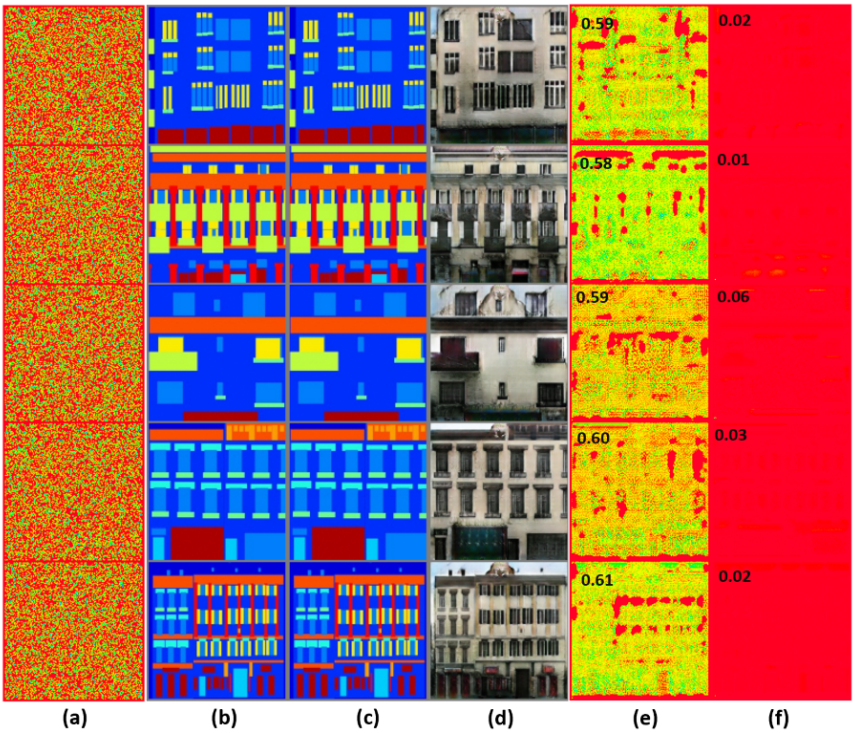

Figure 14. Visualization of samples used for GM Pix2Pix; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

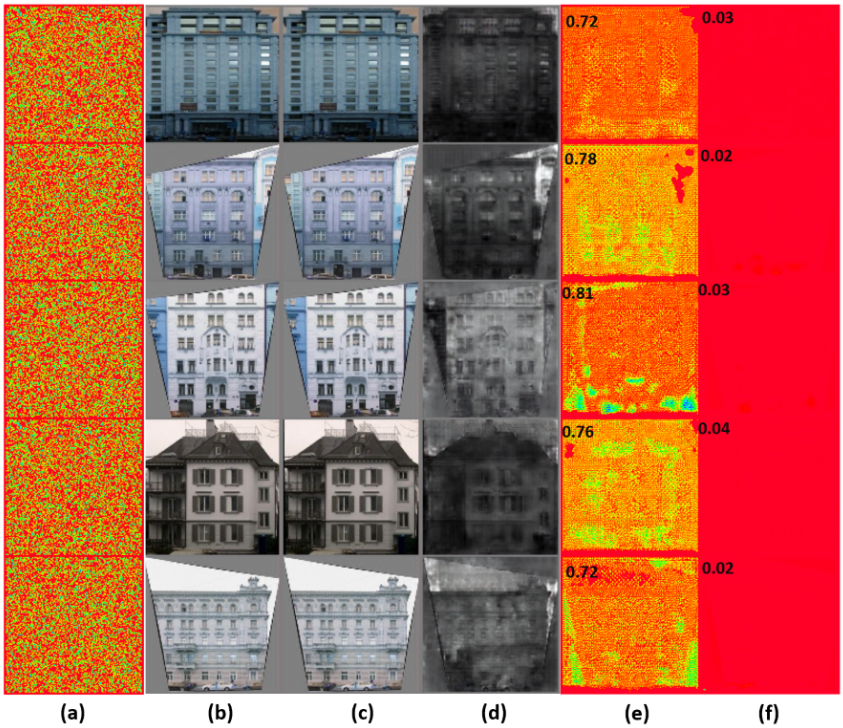

Figure 15. Visualization of samples used for GM DualGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

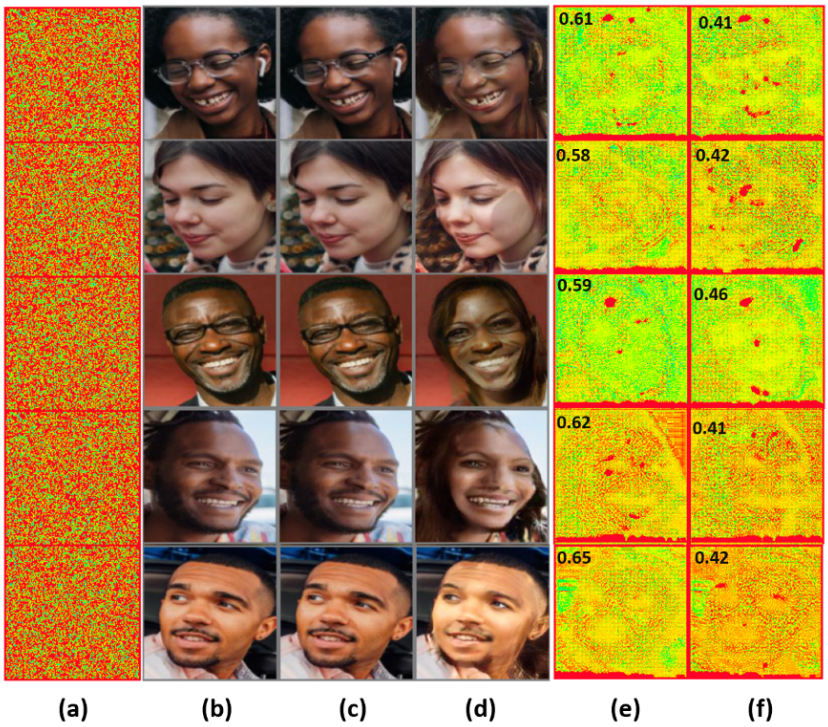

Figure 16. Visualization of samples used for GM CouncilGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

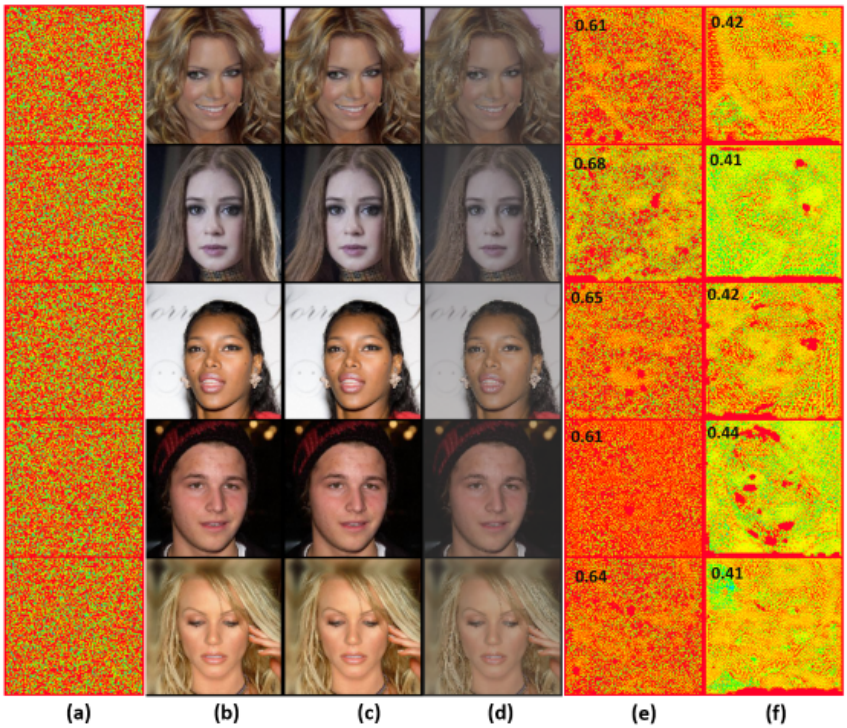

Figure 17. Visualization of samples used for GM ESRGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

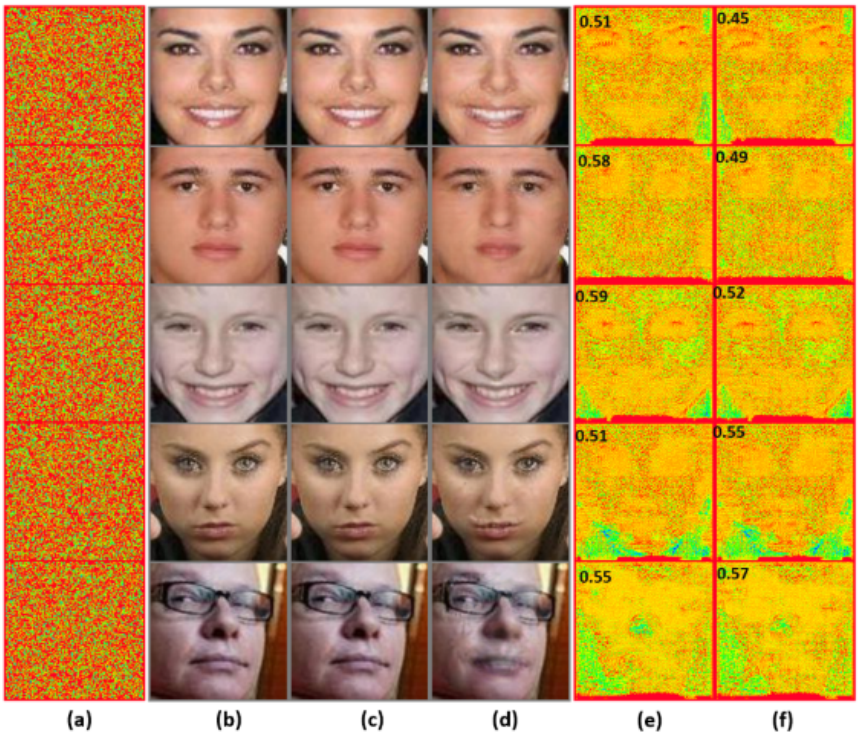

Figure 18. Visualization of samples used for GM GANimation; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.