*APSIPA Transactions on Signal and Information Processing, 2022, 11, e38*

**Original Paper**

# RGGID: A Robust and Green GAN-Fake Image Detector

Yao Zhu[1*], Xinyu Wang[1], Ronald Salloum[2], Hong-Shuo Chen[1] and C.-C. Jay Kuo[1]

[1]*Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90089, USA*

[2]*School of Computer Science and Engineering, California State University, San Bernardino, CA 92407, USA*

---

ABSTRACT

Generative adversarial networks (GANs) are often used to synthesize realistic looking images, which can be a source of dis/misinformation. To detect GAN-fake images effectively, a robust and lightweight detector is proposed and named RGGID (Robust and Green GAN-fake Image Detector) in this work. RGGID is developed under the assumption that GANs fail to generate high-frequency components of real images in high fidelity. Based on this assumption, we design a set of filters using a specific local neighborhood pattern of a pixel, called a PixelHop, and determine the associated discriminant channels. We obtain multiple PixelHops by varying the local patterns, use the validation data to identify discriminant channels, and ensemble their channel responses to yield state-of-the-art detection performance. RGGID offers a green solution since its model size is significantly smaller than that of deep neural networks. Furthermore, we apply common manipulations to real/fake source images, including JPEG compression, resizing and Gaussian additive noise, and demonstrate the robustness of RGGID to these manipulations.

---

*Keywords:* Fake image detection, generative adversarial networks, subspace learning, green learning, PixelHop

---

*Corresponding author: Yao Zhu, yaozhu@usc.edu.

---

Received 23 January 2022; Revised 24 July 2022

ISSN 2048-7703; DOI 10.1561/116.00000005

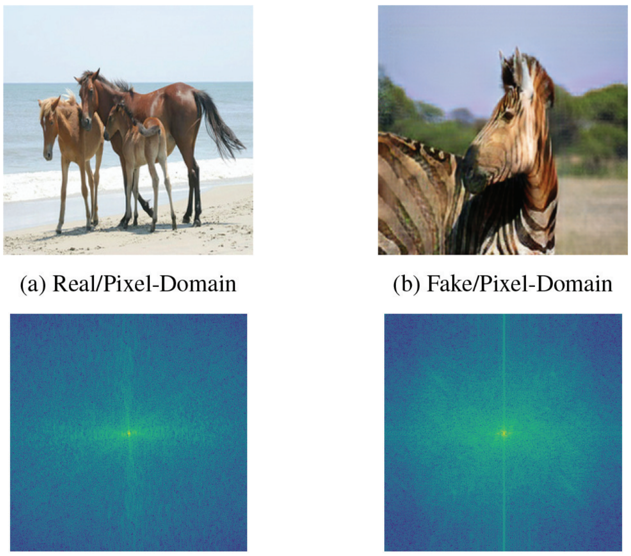
## 1 Introduction

We have witnessed the rapid development of image generation techniques based on convolutional neural networks (CNNs) in general and generative adversarial networks (GANs) [10] in particular. Various GANs have been developed to yield high quality image synthesis and translation performance. The quality of these generated images is so good that it is difficult to distinguish them from real images. This poses a threat to image authenticity and may contribute to a source of dis/misinformation in our society. Effective detection of GAN-fake images has received a lot of attention in recent years.

The challenges of GAN-fake image detection lie in two aspects. First, it is common to apply manipulations to real/fake images in real-world application scenarios. They include JPEG compression, resizing, Gaussian additive noise, etc. The distortions introduced by manipulations may mask small differences between real and fake images and make it even more difficult to perform fake image detection. Thus, it is essential to develop a robust GAN-fake image detector. Second, most state-of-the-art GAN-fake image detectors are built upon deep neural networks (DNNs). They offer good detection performance at the expense of large model sizes, a large number of training images, high training complexity, etc. When dealing with manipulated images, DNN classifiers adopt deeper networks and augment the training set by including all kinds of manipulated images, leading to even larger model sizes and higher training complexity. To address these two problems, we develop a robust and green GAN-fake image detector, named RGGID, in this work.

Our RGGID detector is designed based on the assumption that GANs fail to synthesize high-frequency components in local regions, such as edges and textures, in high fidelity. Following [24] and [20], we show real and fake horse images in the pixel- and the spectral-domains in Figure 1. As compared with the real spectral image, the fake spectral image contains artifacts in diagonal and anti-diagonal directions. This corroborates our assumption that GANs do not synthesize high-frequency components well. Here, we focus on complex local regions that have high-frequency components and employ a set of local filters, called filter banks or PixelHops, to extract features. We develop an ensemble scheme to ensure robust detection under different image manipulations. The RGGID solution outperforms DNN-based GAN-fake image detectors in detection performance. Furthermore, it has three additional advantages: 1) low computational and memory complexity (i.e., green), 2) robustness against image manipulations, and 3) mathematical transparency.

Our current work is an extension of our previous work in [25]. The method proposed in [25] was called A-PixelHop. It was developed and applied only to raw real/fake image detection. No image manipulations were considered in [25]. One of the main contributions of our current work lies in the study of robustness of RGGID against common image manipulations. Image ma-



(a) Real/Pixel-Domain (b) Fake/Pixel-Domain

(c) Real/Spectral-Domain (d) Fake/Spectral-Domain

Figure 1: Examples of real/fake image pairs (top) and their associated spectral-domain representation pairs (bottom).

nipulations introduce additional artifacts to real/fake images. They tend to mask the differences between real/fake images and make the detection problem even more challenging. It is shown by experimental results that RGGID is robust against image manipulations. In addition, we conduct analysis on the experimental results to gain further insights. With respect to our previous work in [25], the main extensions of this work include the following.

- We explore the robustness of RGGID against three common image manipulations.
- We analyze the influence of image manipulations on different semantic categories.
- We propose a new experimental setting called "Leave-None-Out," which is more reasonable, and improve detection performance on weak categories.

The rest of this paper is organized as follows. Related work is reviewed in Section 2. The RGGID method is presented in Section 3. Experimental results are shown in Section 4. The effect of image manipulations on different semantic categories is analyzed and a new experimental setting is presented in Section 5. Concluding remarks are given in Section 6.

## 2 Related Work

### 2.1 GAN-fake Image Detection

Modern image generation models are built upon Generative Adversarial Networks (GANs). CycleGAN [11] is a well-known GAN model that can change the image style, switch semantic objects and translate images from one domain to a different domain without paired training images. GauGAN [19] can translate human sketches to photo-realistic images. One common application of style transfer models is face manipulation. For example, StarGAN [7] can change the expression of a face, alter hair style, or modify skin color. StyleGAN [13] generates fully synthetic human faces with specific high-level attributes such as poses or identities. ProGAN [12] synthesizes high-resolution high-variation face images by progressively growing both the generator and discriminator. BigGAN [1] aims at generating high-quality high-resolution images by leveraging a sequence of best practices on training class-conditional images and scaling up batch sizes.

GAN artifacts have been carefully studied and exploited in GAN-fake image detection. One type of artifact results from the convolutional up-sampling structure of neural networks. Another kind of artifact appears in the form of color distortion, which was used to capture the dissonant or asymmetric characteristics of images in Li *et al.* [17] and Matern *et al.* [15]. Another source of artifacts arises from the artificial fingerprint associated with a GAN architecture. The persistence of these fingerprints across different GAN models, datasets and resolutions was studied in Yu *et al.* [21]. A GAN simulator, called AutoGAN, was introduced in Zhang *et al.* [24] to simulate artifacts of popular GAN models. Zhang *et al.* [24] identified an artifact that manifests itself as spectral peaks in the frequency domain, and thus proposed feeding the spectral-domain input to a classifier for GAN-fake image detection.

Several neural networks have been proposed for GAN-fake image detection. Nataraj *et al.* [18] used the co-occurrence matrix to derive hand-crafted features and fed them to a CNN for detection. Inspired by image steganalysis, Cozzolino *et al.* [8] proposed a CNN to mimic rich models [9] in feature extraction and real/fake classification. Recently, Wang *et al.* [20] trained a CNN classifier with a large number of ProGAN-generated images and evaluated it on images synthesized by eleven other GAN models. They showed the effectiveness of extensive data augmentation in improving the generalization ability of a CNN classifier.

Most research on GAN-fake image detectors has been developed and tested on raw real/fake images. However, most real world images do not exist in the raw image domain. They are compressed for ease of storage and transmission. They may be rescaled to fit different screen sizes. Furthermore, an attacker may add Gaussian noise to real/fake images to make their differentiation more challenging. There is much less work on the robustness of GAN-fake image detectors against image manipulations. Marra *et al.* [16] compared the performance of multiple neural networks under Twitter's compression. They also considered the compression setting mismatch between training and testing datasets to evaluate the robustness of CNN-based classifiers. Wang *et al.* [20] explored data augmentation to enhance the robustness of a detector.

### *2.2 Green Learning*

Green learning aims at an energy efficient way to achieve the goal of data-driven learning. The models should have lower training/inference computational complexity, have smaller model sizes, and require fewer training samples while maintaining similar classification or regression performance as deep-learning models. It is desired that their computation can be carried out solely on CPU or small GPU. Thus, green learning solutions are suitable for edge and mobile computing.

Distinct from the end-to-end optimization of deep learning, green learning adopts a modularized design by following the traditional pattern recognition learning paradigm. It consists of "unsupervised feature learning" and "supervised decision learning." The idea of unsupervised feature learning has been developed in a sequence of papers [5, 6, 14]. While filter parameters of CNNs are obtained by back-propagation, filter parameters in green learning are determined by statistical analysis of the neighborhood of a center pixel. Specifically, a variant of Principle Component Analysis (PCA), called the Saab (Subspace approximation via adjusted bias) transform was proposed in [14] to achieve the task.

Green learning has been successfully applied to various computer vision tasks such as image classification [5, 6] and 3D point classification [22, 23]. In the area of image forensics, green learning solutions or "green forensics," have also been developed, such as deepfake video detection [2] and fake geo-spatial image detection [3] with specific image contents. The former focuses on human face videos while the latter examines satellite images. In this work, we investigate GAN-fake image detection for a wide range of semantic contents, including object images (e.g., apples, oranges, zebras, and horses), scene images (e.g., winter, summer, city, and facades), and paintings of different styles (Ukiyo-e, Van Gogh, Cezanne, and Monet).

## 3 Proposed RGGID Method

An overview of the proposed RGGID method is given in Figure 2. It consists of the following four modules:

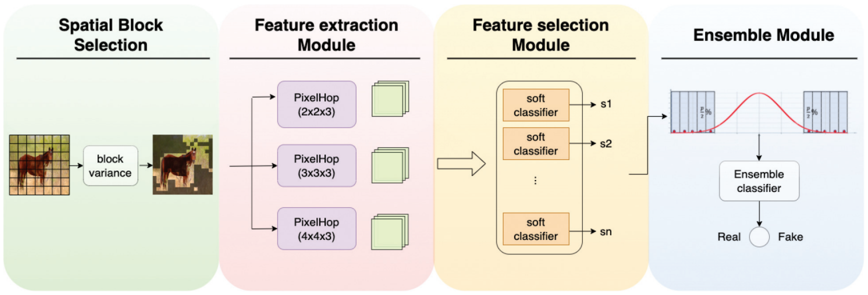


Figure 2: An overview of the proposed RGGID method.

1. **Spatial block selection.** We select blocks that contain a substantial amount of high frequency components.
2. **Feature extraction via parallel PixelHops.** We conduct local spectral analysis by studying frequency responses of multiple sets of local filters. Each set of local filters is called a filter bank or a PixelHop.
3. **Discriminant feature selection and block-level decision making.** We use the validation dataset to identify discriminant channels and use their channel responses as features for the block-level soft decisions.
4. **Image-level decision ensemble.** We ensemble the block-level soft decisions to yield the final image-level binary decision.

Each of them will be detailed below.

## *3.1 Spatial Block Selection*

Since our method is developed based on the assumption that GAN generators are not able to synthesize high-frequency components in high fidelity, we focus on spatial blocks that contain fine details. In the implementation, we partition images into non-overlapping blocks of size $16 \times 16$. Each block will be used as an independent unit for feature extraction, feature selection, and local decision making in the second and third modules. To select blocks containing fine details, the variance of image pixels in a block is computed. That is, we remove the block mean and sum the squares of pixel residuals. For each image, the top 40% blocks with the highest block variances are selected since they contain more energy of high frequency components. Figure 3 shows examples of selected blocks overlaid with the original images. It is evident that selected blocks are from high-frequency regions, such as the horse head and legs in the horse image, trees in the winter image, cars and buildings in the cityscape image, etc.

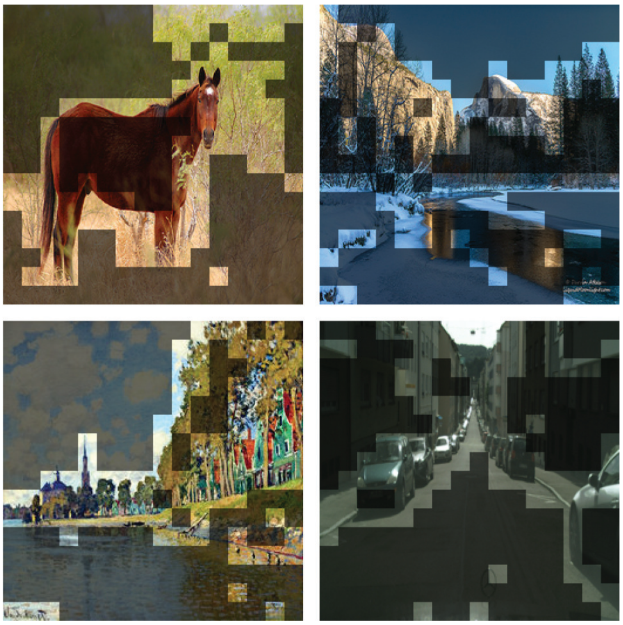


Figure 3: Examples of selected spatial blocks from images, where masked blocks are dropped in further analysis.

### *3.2 Feature Extraction via Parallel PixelHops*

For a squared region of spatial dimension $s \times s$ and spectral dimension $c$, we can define a local neighborhood of dimension $s \times s \times c$. For example, we can set $s = 2$ and $c = 3$ (due to the R, G, B channels of color images). Then, the neighbood has a dimension of 12 (i.e., 12 pixel values). We can consider different weighted sums of these 12 pixel values, which defines a set of filters. The set of filters is called a filter bank. One specific way to define the filter weights is described as follows.

- One DC filter, where all filter weights are set to the same value (i.e., a constant-value vector), and then the vector length is normalized to unity. This filter is called the DC filter and its response is called the DC response.

- Eleven AC filters, where the DC response is subtracted from all pixel values to yield the AC values, principal component analysis is conducted on a collection of neighborhoods, and the eigenvectors associated with non-zero eigenvalues define eleven AC filters.

A filter bank with its filter coefficients selected by this procedure is called a PixelHop. As shown in Figure 2, a PixelHop is used as a feature extraction unit. A PixelHop system is determined by 4 parameters:

1. Neighborhood size $s_1 \times s_2$, where $s_1$ and $s_2$ denote the width and the height, respectively. Typically, we choose squared neighborhoods such that $s_1 = s_2 = s$;

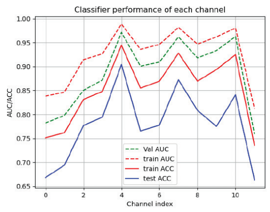
2. Number of spectral components of a pixel, denoted by $c$;

3. The stride number, denoted by $d$, which indicates the amount of movement of the neighborhood horizontally or vertically.

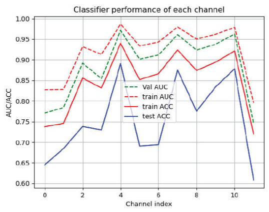
In our design, we select three squared neighborhoods of sizes $2 \times 2$, $3 \times 3$, and $4 \times 4$. The spectral component number, $c$, is equal to 3, and the stride number, $d$, is one. We apply the three PixelHops to blocks of size $16 \times 16$ in parallel without padding. As a result, they have 12, 36 and 48 filter (or channel) responses at $15 \times 15 = 225$, $14 \times 14 = 196$, and $13 \times 13 = 169$ spatial locations, respectively. These responses are called joint spatial-spectral responses. We are interested in channel responses. That is, for a given filter, we collect and order its spatial responses to form a feature vector. For example, for the $2 \times 2 \times 3$ PixelHop, we have 12 channel responses and each of them has a feature vector of dimension 225.

### 3.3 Discriminant Feature Selection and Block-level Decision Making

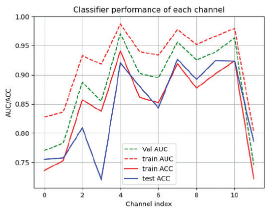
Different spectral channels have different discriminant power in real/fake image detection. As mentioned earlier, we use the responses at different spatial locations as the feature vector. Furthermore, we adopt a gradient boosting tree algorithm called XGBoost [4] as the classifier. To evaluate the discriminant power of a channel, we compare the classifier performance on training, validation and test datasets with the area-under-the-curve (AUC) and the accuracy (ACC) metrics. To give an example, we plot the performance curves of each channel for three semantic categories in the CycleGAN dataset [24] with two PixelHops in Figure 4. In this example, the images are raw real/fake images without any image manipulations. It is evident from the figure that some channels are more discriminant than others. Furthermore, the training, validation and testing datasets share the same discriminant channels. We select those channels with higher validation performance as target channels and train an XGBoost classifier for each channel. In the inference stage, we apply an XGBoost classifier to the spatial responses of the associated channel to obtain a soft decision ranging from 0 to 1, which indicates the probability of the block to be a real or fake image block.



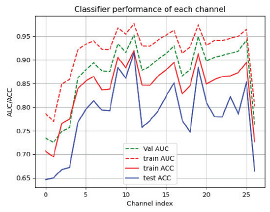
(a) Hor2zeb



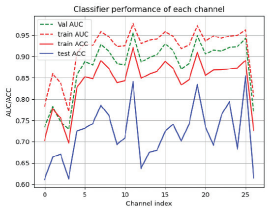
(b) Facade



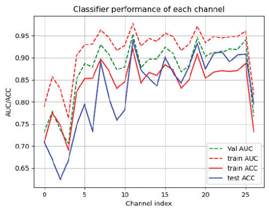
(c) Ukiyoe



(d) Hor2zeb



(e) Facade



(f) Ukiyoe

Figure 4: The detection performance on raw images of three exemplary categories, where the $x$-axis indicates the channel index. The $x$-value ranges from 0 to 11 in (a)–(c), in which a PixelHop of size $2 \times 2 \times 3$ is used. The $x$-value ranges from 0 to 26 in (d)–(f), in which a PixelHop of size $3 \times 3 \times 3$ is used. Each subfigure shows four performance curves: training AUC (red dashed line), training ACC (red line), validation AUC (green dashed line), and test ACC (blue line).

## 3.4 Image-level Decision Ensemble

Given block-level soft decisions from a single image in the third module, we develop an ensemble scheme to yield the final image-level decision in the last module. We first arrange the block-level soft decisions from smallest to largest in the unit interval, i.e., [0, 1]. The decision scores at the two ends are more informative than those in the middle range. Suppose that we plan to sample $p\%$ of blocks to train an ensemble classifier that will yield the image-level decision. Our sampling strategy is to choose $0.5p\%$ soft decisions from the two ends of the distribution as shown in Figure 5, where selected representative

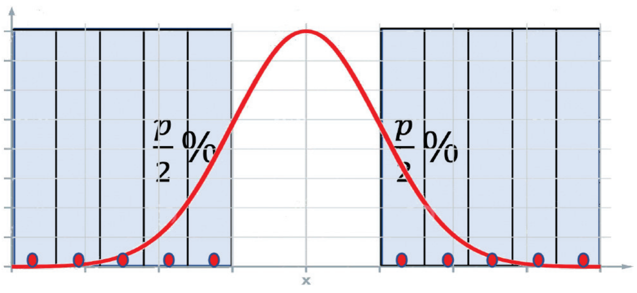


Figure 5: Illustration of the block sampling strategy for the image-level decision ensemble.

soft decisions are denoted by red dots. Hyperparameter p represents the percentage of samples selected from the two tail regions of the distribution of soft decision scores. We set $p = 10, 20, 30, 40$, and choose the one that yields the best performance on the validation set.

## 4 Experiments

### 4.1 CycleGAN Dataset

We evaluated our model on the CycleGAN dataset in [24]. It has 14 semantic categories: Apple, Orange, Horse, Zebra, Yosemite summer, Yosemite winter, Facades, CityScape Photo, Satellite Image, Ukiyo-e, Van Gogh, Cezanne, Monet and Photo. According to the image translation content, the dataset contains 10 subsets where each subset contains both real and translated images. For example, the *hor2zeb* subset includes real horse and zebra images for training CycleGAN and corresponding fake horse and zebra images generated from the trained model. In total, there are over 36k images in the CycleGAN dataset.

We conducted experiments using the Leave-One-Out setting, as was done in [24] and [20]. Namely, one semantic category will be set aside for testing and the remaining semantic categories will be used for training and validation. We use the ratio of 8:2 to split the data from the remaining nine categories into training and validation sets. In this case, our proposed method is not restricted to a specific semantic category and can generalize well to all CycleGAN images. We tested our model under 3 different image manipulation techniques: JPEG compression, image resizing and additive noise. For each type of manipulation, both the training and testing images will be processed with the same manipulation setting to avoid mismatch. First, we discuss the detection on raw image data as reference. Then, we examine the scenarios in which the various manipulations are applied. We noticed that there exists a few categories that are relatively sensitive to manipulations. In Section 5, we analyze the effect of manipulations on sensitive categories and demonstrate that, by including a small amount of images from the sensitive categories in the training stage, our RGGID method is robust to image manipulations for all semantic categories in the CycleGAN dataset.

### 4.2 Detection on Raw Images

Table 1 shows the test detection results on the raw CycleGAN dataset with only 10% training data. By 10% training data, we mean 10% of training images from each category while keeping the validation and test sets unchanged. The results show that RGGID can perform well even under extremely weak supervision, which is discussed in more detail in Section 5.3. We compare

Table 1: Test accuracy on raw images with 10% training data.

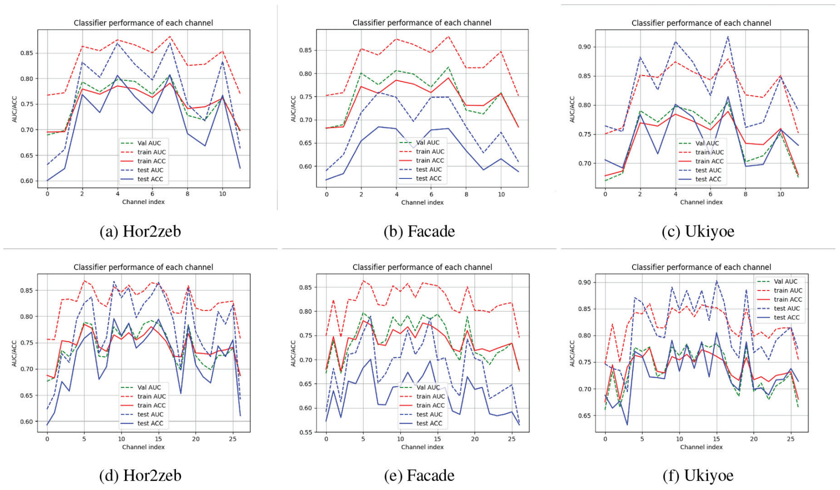
| Accuracy | ap2or | hor2zeb | win2sum | citysc. | facades | map2sat | Ukiyoe | Van Gogh | Cezanne | Monet | average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DenseNet | 79.1 | 95.8 | 67.7 | 93.8 | 99.0 | 78.3 | 99.5 | 97.7 | 99.9 | 89.8 | 89.2 |
| XceptionNet | 95.9 | 99.2 | 76.7 | 100.0 | 98.6 | 76.8 | 100.0 | 99.9 | 100.0 | 95.1 | 94.5 |
| InceptionNet | 85.0 | 94.8 | 58.8 | 99.4 | 94.0 | 70.5 | 99.8 | 98.8 | 99.9 | 89.9 | 89.1 |
| Cozzolino2017 | 99.9 | 99.9 | 61.2 | 99.9 | 97.3 | 99.6 | 100.0 | 99.9 | 100.0 | 99.2 | 95.1 |
| Auto-Spec | 98.3 | 98.4 | 93.3 | 100.0 | 100.0 | 78.6 | 99.9 | 97.5 | 99.2 | 99.7 | 97.2 |
| Nataraj2019 | 99.7 | 99.8 | 99.8 | 80.6 | 92.0 | 97.5 | 99.6 | 100.0 | 99.6 | 99.2 | 96.8 |
| RGGID (6 channels) | 99.2 | 99.8 | 100.0 | 94.4 | 100.0 | 94.1 | 100.0 | 100.0 | 100.0 | 99.4 | 98.7 |
| RGGID (9 channels) | 99.2 | 99.7 | 100.0 | 94.4 | 100.0 | 95.8 | 100.0 | 100.0 | 100.0 | 99.2 | 98.8 |
| RGGID (12 channels) | 99.2 | 99.9 | 100.0 | 95.9 | 100.0 | 95.8 | 100.0 | 100.0 | 100.0 | 99.1 | **99.0** |

the proposed RGGID method with six state-of-the-art models. The highest performance we obtained is 99.0% test accuracy acquired from the fusion of 12 channels, in which we select the 4 best channels from each of the three filter banks (i.e., the $2 \times 2 \times 3$, $3 \times 3 \times 3$, and $4 \times 4 \times 3$ filter banks). The second best is 98.8% from 9 channel fusion, in which we select the 3 best channels from each filter bank. The 6 channel fusion result is the same as the one presented in [25]. This is the case where we achieve equally good performance but with the smallest model size. This indicates that our PixelHop solution is very powerful even if only a few channels are selected in the ensemble process.

## *4.3 JPEG Compression Manipulation*

To assess the robustness of RGGID under realistic scenarios, we run experiments in which images are compressed using different quality factors. JPEG compression creates distortions such as blocking and ringing artifacts that interfere with the up-sampling artifact originating from generative models. We verified the assumption that high-frequency responses are more distinguishable than other frequencies for the raw data. However, when applying JPEG compression, the high-frequency components of real compressed images are severely distorted as well. As a result, the difference between real and fake images is less discernible. Figure 6 shows the soft classification performance for each spectral channel on compressed images with quality factor 85. We see that discriminant channels are shifted from high-frequency bands to mid-and-low frequency bands.

We chose three commonly-used quality factors, i.e., 75, 85, and 95, in the experiments. Table 2 shows the test accuracy of RGGID for JPEG compressed images. For each quality factor, we show results for individual filter banks as well as for ensemble settings. Results for individual filter banks are marked as $2 \times 2 \times 3$ *only*, $3 \times 3 \times 3$ *only*, and $4 \times 4 \times 3$ *only*. Results for ensemble schemes



(a) Hor2zeb (b) Facade (c) Ukiyoe

(d) Hor2zeb (e) Facade (f) Ukiyoe

Figure 6: Soft classification performance of exemplary categories on JPEG compressed images (QF = 85): For subfigures (a)–(c), the filter size is $2 \times 2 \times 3$ (i.e., the number of channels is 12), while for subfigures (d)–(f), the filter size is $3 \times 3 \times 3$ (i.e., the number of channels is 27). For each subfigure, we show five performance curves: train set AUC (red dashed line), train set ACC (red line), validation set AUC (green dashed line), test set AUC (blue dashed line), and test set ACC (blue line).

are marked as *ensemble*. For example, *ensemble (2&3)* means that we use only discriminant channels from the $2 \times 2 \times 3$ and $3 \times 3 \times 3$ filter banks. On the other hand, *ensemble (all)* is the case where we use discriminant channels from all filter banks. For each quality factor, we use **bold** to mark the setting with highest average test accuracy, and <u>underline</u> for the setting with the second highest accuracy. Generally speaking, $2 \times 2 \times 3$ *only* tends to have better performance than other settings. This could be attributed to the $8 \times 8$ block DCT transform used in JPEG. Also, the $2 \times 2 \times 3$ filter bank is more favorable than the $4 \times 4 \times 3$ filter bank. This is because we select the same number of channels from each filter bank. Feature maps of selected channels in the $2 \times 2 \times 3$ filter bank are more informative. This also explains the reason why ensemble schemes do not always give the best result. Also, we see that the accuracies for the *ap2or* and *map2sat* categories are significantly lower than other categories, which will be analyzed in Section 6.

Furthermore, we compare RGGID with other state-of-the-art methods in Table 3. Here, we present results from 8 other state-of-the-art methods whose performance scores are taken from [16]. In Table 3, the first 5 models are relatively shallow networks while the last three (DenseNet, InceptionNet, and XceptionNet) are deeper neural networks. Their performance scores are based on Twitter-like compression as explained in [16]. However, their compression quality factor is not explicitly provided. For fair comparison, we average our

Table 2: Test accuracy of RGGID on JPEG compressed images.

| Quality factor | Setting | ap2or | hor2zeb | win2sum | citysc. | facades | map2sat | Ukiyoe | Van Gogh | Cezanne | Monet | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QF = 75 | $2 \times 2 \times 3$ only | 67.58 | 89.21 | 90.31 | 93.78 | 89.50 | 70.98 | 90.53 | 98.83 | 98.97 | 81.40 | **87.11** |
| | $3 \times 3 \times 3$ only | 66.11 | 88.25 | 87.32 | 68.74 | 93.75 | 69.98 | 87.80 | 89.53 | 98.87 | 81.96 | 83.23 |
| | $4 \times 4 \times 3$ only | 63.43 | 89.09 | 86.16 | 89.06 | 93.38 | 60.40 | 83.53 | 80.47 | 97.50 | 81.03 | 82.40 |
| | ensemble (2&3) | 68.05 | 90.38 | 90.99 | 81.24 | 93.00 | 60.03 | 89.17 | 93.33 | 99.47 | 80.93 | 84.66 |
| | ensemble (2&4) | 67.08 | 88.28 | 90.58 | 91.29 | 93.13 | 59.35 | 86.83 | 93.37 | 97.70 | 81.36 | 84.90 |
| | ensemble (all) | 65.91 | 89.96 | 88.90 | 89.58 | 92.88 | 63.91 | 88.43 | 94.80 | 98.03 | 81.08 | <u>85.34</u> |
| QF = 85 | $2 \times 2 \times 3$ only | 70.66 | 91.36 | 90.13 | 97.19 | 91.63 | 50.23 | 93.07 | 98.73 | 99.83 | 81.22 | 86.36 |
| | $3 \times 3 \times 3$ only | 62.31 | 92.17 | 93.43 | 97.18 | 95.75 | 51.92 | 94.20 | 93.83 | 99.83 | 82.25 | 86.29 |
| | $4 \times 4 \times 3$ only | 63.28 | 92.00 | 92.80 | 89.83 | 93.88 | 69.66 | 94.20 | 91.10 | 98.70 | 80.95 | 86.64 |
| | ensemble (2&3) | 72.64 | 91.69 | 95.10 | 87.65 | 95.63 | 62.68 | 96.23 | 98.33 | 99.80 | 82.35 | 88.21 |
| | ensemble (2&4) | 73.58 | 92.17 | 94.92 | 95.90 | 94.50 | 60.58 | 95.03 | 96.57 | 99.50 | 81.82 | <u>88.46</u> |
| | ensemble (all) | 72.64 | 92.46 | 95.30 | 95.83 | 95.75 | 67.52 | 96.20 | 96.53 | 99.70 | 82.34 | **89.43** |
| QF = 95 | $2 \times 2 \times 3$ only | 66.91 | 92.88 | 97.65 | 96.52 | 95.88 | 50.00 | 98.80 | 99.73 | 99.97 | 89.87 | **88.82** |
| | $3 \times 3 \times 3$ only | 64.90 | 95.46 | 97.58 | 91.92 | 95.13 | 50.00 | 99.20 | 97.37 | 99.97 | 89.27 | 88.08 |
| | $4 \times 4 \times 3$ only | 63.51 | 95.81 | 96.90 | 88.72 | 96.38 | 53.47 | 98.57 | 92.70 | 99.53 | 85.56 | 87.16 |
| | ensemble (2&3) | 67.08 | 94.29 | 97.72 | 92.37 | 95.88 | 50.00 | 99.20 | 99.17 | 99.97 | 91.10 | <u>88.68</u> |
| | ensemble (2&4) | 68.42 | 94.25 | 97.49 | 93.85 | 95.63 | 51.41 | 98.93 | 97.67 | 99.83 | 88.92 | 88.64 |
| | ensemble (all) | 64.42 | 94.25 | 97.74 | 91.06 | 96.25 | 51.32 | 99.23 | 98.87 | 99.90 | 89.42 | 88.26 |

best result for each quality factor and present it in the last row of Table 3. In terms of the average test accuracy across all semantic categories, RGGID is very close to the two best models, DenseNet and XceptionNet, with only a 0.06% and 0.58% performance gap, respectively. Although XceptionNet is able to achieve marginally better performance in this particular experiment, it is important to point out that XceptionNet is a very deep neural network

Table 3: Test accuracy comparison of different detectors for JPEG compressed images.

| Method | ap2or | hor2zeb | win2sum | citysc. | facades | map2sat | Ukiyoe | Van Gogh | Cezanne | Monet | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Steganalysis feat. | 79.39 | 90.02 | 56.66 | 92.17 | 73.62 | 69.39 | 65.83 | 95.30 | 94.73 | 80.89 | 81.09 |
| GAN discr. | 63.29 | 91.08 | 51.90 | 53.14 | 88.75 | 79.35 | 76.56 | 80.32 | 96.41 | 81.83 | 73.33 |
| Cozzolino2017 | 79.57 | 89.82 | 53.74 | 86.81 | 62.88 | 89.64 | 67.67 | 98.80 | 99.93 | 87.33 | 82.62 |
| Bayar2016 | 54.64 | 95.34 | 50.27 | 54.00 | 90.63 | 52.69 | 58.90 | 74.27 | 99.77 | 78.60 | 69.17 |
| Rahmouni2017 | 84.96 | 98.35 | 54.30 | 57.60 | 91.88 | 54.93 | 96.83 | 99.63 | 99.77 | 89.72 | 80.97 |
| DenseNet | 78.27 | 93.44 | 66.94 | 97.83 | 98.19 | 80.45 | 97.54 | 98.53 | 99.57 | 83.95 | <u>88.51</u> |
| InceptionNet v3 | 78.60 | 95.23 | 64.54 | 96.09 | 90.14 | 63.84 | 99.53 | 96.31 | 100.00 | 86.21 | 87.37 |
| XceptionNet | 93.52 | 93.77 | 67.07 | 95.11 | 99.22 | 67.97 | 99.66 | 95.18 | 99.97 | 84.02 | **89.03** |
| RGGID | 69.04 | 91.52 | 94.42 | 95.38 | 93.71 | 62.83 | 95.18 | 98.36 | 99.55 | 84.54 | 88.45 |

with more than 22.9M trainable parameters, while RGGID has only 76.2K trainable parameters.

## *4.4 Image Resizing Manipulation*

Another common image manipulation is resizing. We focus on the scenario of resizing to lower spatial resolutions, referred to as down-sizing. Since the down-sizing operation interacts with artifacts arising from up-sampling in generative models and the differences between real and fake images becomes obscure, down-sized fake images are more challenging to detect.

There is little work on detecting resized real/fake images. Zhang *et al.* [24] chose 4 image sizes and randomly selected one as the target size. They trained a neural network with CycleGAN and Auto-GAN horse images, and tested it on other categories. Here, we consider 2 resizing factors (0.5 and 0.75) and conduct experiments under the “Leave-One-Out” setting for all categories.

Table 4 shows the results for individual filter banks as well as for ensemble settings. The proposed RGGID method can achieve a maximum accuracy of 95.45% and 92.84% for resize factors of 0.75 and 0.5, respectively. As compared with the 99% detection accuracy on raw images in Table 1, the accuracy degrades by 3.55% and 6.16% for resize factors of 0.75 and 0.5, respectively. Thus, RGGID is robust with respect to image resizing.

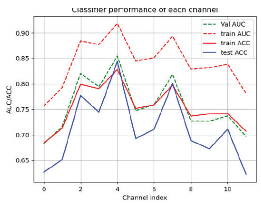
Table 4: Test accuracy of RGGID for resized images.

| Resize factor | Setting | ap2or | hor2zeb | win2sum | citysc. | facades | map2sat | Ukiyoe | Van Gogh | Cezanne | Monet | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $2 \times 2 \times 3$ only | 97.89 | 95.67 | 99.73 | 64.83 | 96.75 | 99.13 | 99.87 | 99.47 | 99.13 | 95.92 | 94.84 |
| | $3 \times 3 \times 3$ only | 95.38 | 95.84 | 99.84 | 79.80 | 93.13 | 95.03 | 99.60 | 98.97 | 99.60 | 97.32 | **95.45** |
| 0.75 | $4 \times 4 \times 3$ only | 98.14 | 98.40 | 99.47 | 53.16 | 99.63 | 63.77 | 99.30 | 97.50 | 99.83 | 97.88 | 90.71 |
| | ensemble (2&3) | 98.39 | 96.31 | 99.91 | 73.38 | 94.13 | 95.62 | 99.80 | 99.63 | 99.56 | 97.10 | 95.38 |
| | ensemble (all) | 98.36 | 97.06 | 99.89 | 73.79 | 98.88 | 88.50 | 99.90 | 99.33 | 99.80 | 97.55 | 95.31 |
| | $2 \times 2 \times 3$ only | 95.63 | 95.48 | 99.31 | 50.81 | 93.25 | 94.71 | 99.30 | 94.60 | 98.73 | 87.34 | 90.92 |
| | $3 \times 3 \times 3$ only | 91.88 | 96.79 | 97.99 | 55.92 | 95.88 | 96.35 | 98.70 | 94.70 | 94.00 | 83.48 | 90.57 |
| 0.5 | $4 \times 4 \times 3$ only | 94.51 | 96.08 | 97.27 | 75.39 | 88.50 | 90.37 | 98.50 | 93.00 | 97.93 | 85.65 | 91.72 |
| | ensemble (2&3) | 94.69 | 96.94 | 99.27 | 55.98 | 95.75 | 98.04 | 99.17 | 95.27 | 98.13 | 86.64 | 91.99 |
| | ensemble (all) | 96.67 | 96.33 | 98.84 | 70.03 | 92.75 | 92.61 | 99.50 | 95.30 | 98.47 | 87.93 | **92.84** |

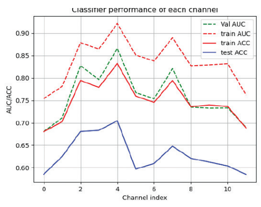
For a resize factor of 0.75, the $3\times3\times3$ filter bank yields the best performance while the ensemble of the $3 \times 3 \times 3$ and $2 \times 2 \times 3$ filter banks yields the second best performance. For a resize factor of 0.5, individual filter banks are less effective, and the ensemble of all three filter banks gives the best performance.

## *4.5 Additive Gaussian Noise Manipulation*

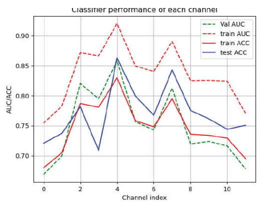
The third image manipulation tested is additive Gaussian noise. Although it may not be as common as JPEG compression and image resizing in social media, Gaussian noise could be used to cover up certain weaknesses in synthesized images. It is essential to demonstrate the robustness of RGGID against additive Gaussian noise. In our experiments, we normalize the pixel values of the raw image to $[0, 1]$ and use Gaussian noise with two noise levels (namely, $\sigma = 0.01$ and $0.02$) to simulate a realistic scenario in forensics. Because additive noise introduces additional high-frequency information to the raw image, the source differences between real and fake images in high-frequency regions are diminished. This phenomenon is observed in the soft classification performance shown in Figure 7, where $\sigma = 0.01$. Similar to Figure 6, we see that high-frequency channels are not as discriminant as those in the raw image dataset. Discriminant channels are shifted from high-frequency to mid-frequency bands.



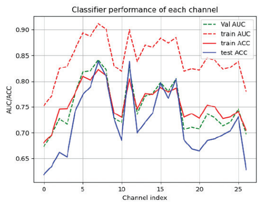
(a) Hor2zeb



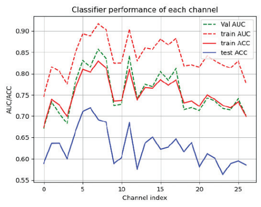
(b) Facade



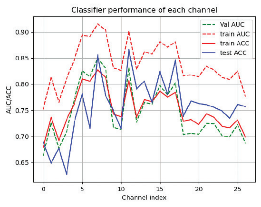
(c) Ukiyoe



(d) Hor2zeb



(e) Facade



(f) Ukiyoe

Figure 7: Soft classification performance of six exemplary semantic categories on noisy images ($\sigma = 0.01$), (a)–(c) uses the $2 \times 2 \times 3$ filter bank so that the $x$-axis has 12 channels, (d)–(f) uses the $3 \times 3 \times 3$ filter bank so that the $x$-axis has 27 channels. Each subfigure shows four performance curves: train set AUC (red dashed line), train set ACC (red line), validation set AUC (green dashed line), and test set ACC (blue line).

Table 5 shows the test accuracy for each semantic category under different noise levels. When $\sigma = 0.01$, RGGID can achieve a maximum average accuracy of 89.89%, which is approximately a 10% drop as compared to the accuracy for the raw image dataset. When the noise level is increased to $\sigma = 0.02$, noise in the smooth regions is visible to human eyes, and detection of fake images becomes more challenging. In this case, the maximum average accuracy of

Table 5: Test accuracy of RGGID for noisy images.

| $\sigma$ | Setting | ap2or | hor2zeb | win2sum | citysc. | facades | map2sat | Ukiyoe | Van Gogh | Cezanne | Monet | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma = 0.01$ | $2 \times 2 \times 3$ only | 64.42 | 91.71 | 95.96 | 92.52 | 95.5 | 52.24 | 93.07 | 99.63 | 99.83 | 99.49 | 88.44 |
| | $3 \times 3 \times 3$ only | 65.27 | 94.81 | 99.13 | 75.29 | 96.25 | 52.11 | 99.27 | 99.97 | 99.93 | 99.88 | 88.19 |
| | $4 \times 4 \times 3$ only | 59.51 | 96.13 | 98.70 | 53.31 | 95.62 | 72.39 | 98.77 | 99.90 | 99.23 | 98.64 | 87.22 |
| | ensemble (2&3) | 64.70 | 94.69 | 94.12 | 93.28 | 96.38 | 51.93 | 97.33 | 99.63 | 99.87 | 99.81 | 89.17 |
| | ensemble (3&4) | 64.50 | 93.73 | 98.77 | 92.84 | 96.38 | 53.81 | 99.17 | 99.83 | 99.97 | 99.88 | **89.89** |
| | ensemble (all) | 62.39 | 94.69 | 94.12 | 93.28 | 96.38 | 53.63 | 97.73 | 99.90 | 99.90 | 99.81 | <u>89.18</u> |
| $\sigma = 0.02$ | $2 \times 2 \times 3$ only | 68.20 | 90.52 | 94.80 | 51.65 | 90.00 | 63.96 | 97.67 | 98.80 | 98.90 | 96.65 | 85.12 |
| | $3 \times 3 \times 3$ only | 66.08 | 89.53 | 96.90 | 64.84 | 91.13 | 65.92 | 97.53 | 97.83 | 98.93 | 96.58 | **86.52** |
| | $4 \times 4 \times 3$ only | 69.76 | 93.17 | 94.19 | 51.83 | 89.13 | 76.94 | 95.70 | 97.83 | 97.20 | 97.86 | <u>86.36</u> |
| | ensemble (2&3) | 67.85 | 91.84 | 97.10 | 53.31 | 91.13 | 65.10 | 97.63 | 98.07 | 98.47 | 96.62 | 85.71 |
| | ensemble (3&4) | 65.47 | 92.50 | 93.14 | 64.17 | 87.50 | 67.35 | 96.63 | 98.87 | 99.47 | 98.17 | 86.32 |
| | ensemble (all) | 64.23 | 92.61 | 94.12 | 53.34 | 86.38 | 63.61 | 96.20 | 98.40 | 99.27 | 98.17 | 84.63 |

RGGID is 86.52% using the $3 \times 3 \times 3$ filter bank. Overall, RGGID can maintain good detection performance against additive Gaussian noise.

## 5 Analysis

For each of the three aforementioned image manipulations, there are certain categories for which the performance is significantly lower as compared to the remaining categories. They are referred to as challenging categories. They are *ap2or* and *map2sat* for JPEG compression, *citysc* for image resizing, and *ap2or*, *citysc* and *map2sat* for additive Gaussian noise. We first analyze the effect of image manipulations in Section 5.1. Next, we propose a new experimental setting called *Leave-None-Out* in Section 5.2. We conduct extensive experiments under the new setting and show that the performance for the challenging categories can be increased significantly.

### *5.1 Image Manipulation Analysis*

For JPEG compression, the two challenging categories are *ap2or* and *map2sat*. Figure 8 shows examples of original images, the corresponding JPEG compressed images, difference maps between the original and compressed images, and the spectra for both images. As revealed by the difference maps, we observe stronger distortion on the *ap2or* and *map2sat* images caused by JPEG compression as compared with *hor2zeb* and *Ukiyoe* images, which are considered easy categories. Furthermore, stronger high-frequency components of original images in the *ap2or* and *map2sat* categories are also revealed by their spectra (see the fourth column). On one hand, these high-frequency

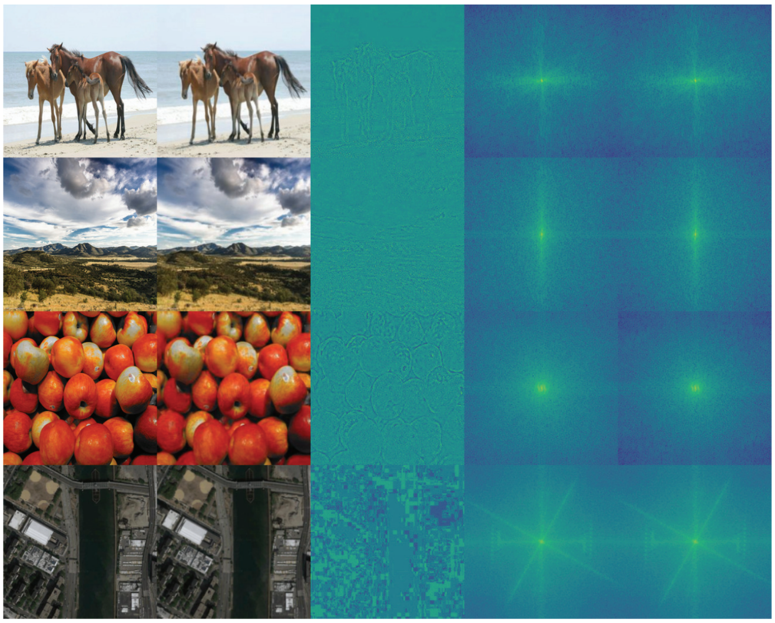


Figure 8: Frequency analysis on JPEG compression (QF = 75). We show exemplary images from *hor2zeb*, *Ukiyoe*, *ap2or*, and *map2sat* categories (in the 1st column), JPEG compressed images (in the 2nd column), difference maps between original and JPEG compressed images (in the 3rd column), spectra of original images (in the 4th column) and spectra of compressed images (in the 5th column).

components cannot be synthesized well in GAN-fake images. On the other hand, they are degraded by JPEG compression for both real and fake images as well. JPEG compression offers a masking effect on the generation artifact in fake image detection with respect to these two challenging categories.

Similarly, we conduct frequency analysis of image resizing in Figure 9. Images from the *citysc* category contain street views from car cameras and its content contains many vertical edges. As shown in the figure, resizing introduces stronger vertical distortion on images from the *citysc* category as compared to other categories. These vertical edges in raw images offer good cues for fake image detection. Since these cues are masked by resizing, it becomes more challenging to differentiate real and fake images.

The same phenomenon is observed for the *ap2or*, *citysc* and *map2sat* categories under the additive noise manipulation as shown in Figure 10. By comparing the spectra before and after additive noise, we see that *ap2or*, *citysc* and *map2sat* images are more affected by additive noise than *Cezanne* and *win2sum* images. It is worthwhile to point out that the difference in spectral image for the satellite map category is not as obvious as that for the *citysc*

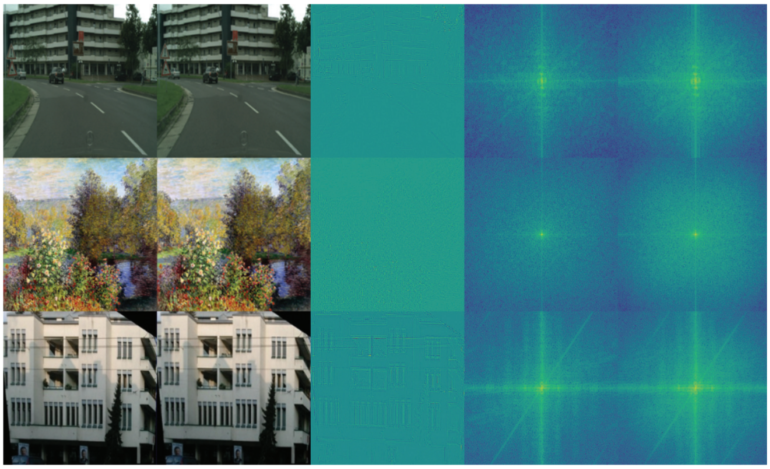


Figure 9: Frequency analysis on image resizing (Resize factor = 0.75). We show exemplary images from *citysc*, *Monet*, and *facades* categories (in the 1st column), resized images (in the 2nd column), difference maps (in the 3rd column), spectra of original images (in the 4th column) and spectra of resized images (in the 5th column). For ease of comparison, we resize original images to smaller size and resize them back to original size and compute the pixel-wise difference between the two to yield the difference map.

category. This is because the satellite map images are much larger in size and the scales of their spectral images are actually different.

### 5.2 Leave-None-Out Setting

From analysis in Section 5.1, we see that leaving a specific semantic category out during training can affect the performance for certain semantic categories under image manipulation scenarios. For example, if we leave the *ap2or* semantic category out in JPEG compression, its performance becomes much worse as shown in Table 2.

Actually, the Leave-One-Out setting is not practical in real-world forensics. It is reasonable to assume that we can have access to all semantic category images when we are confronted with fake image attacks. For this reason, we propose another experimental setting called **Leave-None-Out**, where all semantic categories in the CycleGAN dataset are employed in the training process. In this setting, we enlarge the training dataset by including 10% of test category images and use the other 90% of test category images in testing.

Since this setting only includes a small number of test category images in the training set, we can still examine the detection performance and robustness of our model with respect to a specific semantic category. We conduct experiments under the Leave-None-Out setting for each manipulation and show the results

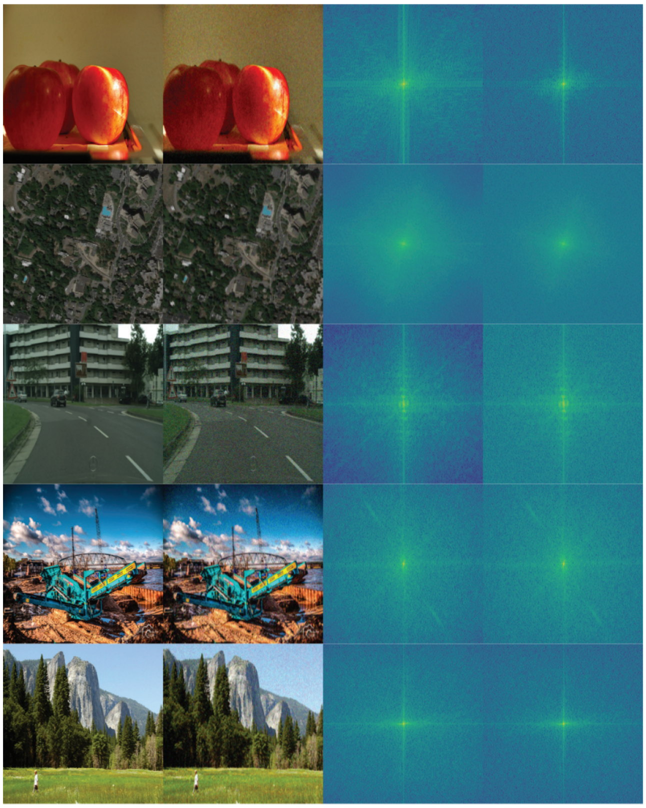


Figure 10: Frequency analysis on additive Gaussian noise with $\sigma = 0.02$. We show exemplary images from *ap2or*, *map2sat*, *citysc*, *Cezanne*, and *win2sum* categories (in the 1st column), noisy images (in the 2nd column), spectra of original images (in the 3rd column), and spectra of noisy images (in the 4th column).

in Tables 6, 7, and 8 for JPEG compression, resizing, and additive noise, respectively. We use * to denote the result under the Leave-None-Out setting. As compared with the Leave-One-Out setting, we observe a 3–20% test accuracy increase for the new setting. For example, for image resizing with a resize factor of 0.5, the test accuracy of RGGID improves from 70.03% to 93.09% for the *citysc* category with an ensemble of all three filter banks.

## 5.3 Model Size, Computational Complexity and Weak Supervision

The proposed RGGID method is a green solution since it has low computational and memory complexity and it can achieve high performance with weak supervision as discussed below.

Table 6: Test accuracy for JPEG-compressed images under the Leave-None-Out setting.

| | Setting | ap2or | map2sat | ap2or* | map2sat* |
|---|---|---|---|---|---|
| QF = 75 | $2 \times 2 \times 3$ only | 67.58 | 70.98 | 72.30 | 79.21 |
| | $3 \times 3 \times 3$ only | 66.11 | 69.98 | 68.74 | 66.18 |
| | $4 \times 4 \times 3$ only | 63.43 | 60.40 | 66.36 | 74.85 |
| | ensemble (2&3) | 68.05 | 60.03 | 72.19 | 69.62 |
| | ensemble (2&4) | 67.08 | 59.35 | 72.27 | 74.29 |
| | ensemble (all) | 65.91 | 63.91 | 72.16 | 74.85 |
| QF = 85 | $2 \times 2 \times 3$ only | 70.66 | 50.23 | 77.21 | 71.55 |
| | $3 \times 3 \times 3$ only | 62.31 | 51.92 | 72.46 | 73.63 |
| | $4 \times 4 \times 3$ only | 63.28 | 69.66 | 70.25 | 77.43 |
| | ensemble (2&3) | 72.64 | 62.68 | 76.35 | 72.16 |
| | ensemble (2&4) | 73.58 | 60.58 | 76.88 | 70.59 |
| | ensemble (all) | 72.64 | 67.52 | 76.38 | 70.59 |
| QF = 95 | $2 \times 2 \times 3$ only | 66.91 | 50.00 | 87.09 | 69.23 |
| | $3 \times 3 \times 3$ only | 64.90 | 50.00 | 84.08 | 75.11 |
| | $4 \times 4 \times 3$ only | 63.51 | 53.47 | 81.46 | 66.23 |
| | ensemble (2&3) | 67.08 | 50.00 | 86.78 | 76.19 |
| | ensemble (2&4) | 68.42 | 51.41 | 86.59 | 78.26 |
| | ensemble (all) | 64.42 | 51.32 | 86.70 | 77.01 |

Table 7: Test accuracy for resized images under the Leave-None-Out setting.

| | Setting | citysc. | citysc.* |
|---|---|---|---|
| Resize factor 0.75 | $2 \times 2 \times 3$ only | 64.83 | 97.48 |
| | $3 \times 3 \times 3$ only | 79.80 | 99.05 |
| | $4 \times 4 \times 3$ only | 53.16 | 98.58 |
| | ensemble (2&3) | 73.38 | 99.38 |
| | ensemble (all) | 73.79 | 99.85 |
| Resize factor 0.5 | $2 \times 2 \times 3$ only | 50.81 | 82.85 |
| | $3 \times 3 \times 3$ only | 55.92 | 67.59 |
| | $4 \times 4 \times 3$ only | 75.39 | 91.24 |
| | ensemble (2&3) | 55.98 | 83.69 |
| | ensemble (all) | 70.03 | 93.09 |

Table 8: Test accuracy for noisy images under the Leave-None-Out setting.

| | Setting | ap2or | citysc. | map2sat | ap2or* | citysc.* | map2sat* |
|---|---|---|---|---|---|---|---|
| $\sigma = 0.01$ | $2 \times 2 \times 3$ only | 64.42 | 92.52 | 52.24 | 73.58 | 81.18 | 88.63 |
| | $3 \times 3 \times 3$ only | 65.27 | 75.29 | 52.11 | 78.90 | 84.67 | 81.55 |
| | $4 \times 4 \times 3$ only | 59.51 | 53.31 | 72.39 | 69.37 | 96.39 | 90.77 |
| | ensemble (2&3) | 64.70 | 93.28 | 51.93 | 75.56 | 81.18 | 83.71 |
| | ensemble (3&4) | 64.50 | 92.84 | 53.81 | 76.39 | 84.67 | 79.33 |
| | ensemble (all) | 62.39 | 93.28 | 53.63 | 72.07 | 81.18 | 83.71 |
| $\sigma = 0.02$ | $2 \times 2 \times 3$ only | 68.20 | 51.65 | 63.96 | 72.76 | 85.15 | 72.47 |
| | $3 \times 3 \times 3$ only | 66.08 | 64.84 | 65.92 | 77.30 | 78.47 | 81.83 |
| | $4 \times 4 \times 3$ only | 69.76 | 51.83 | 76.94 | 73.84 | 94.75 | 86.42 |
| | ensemble (2&3) | 67.85 | 53.31 | 65.10 | 76.59 | 73.63 | 76.01 |
| | ensemble (3&4) | 65.47 | 64.17 | 67.35 | 77.21 | 78.47 | 88.42 |
| | ensemble (all) | 64.23 | 53.34 | 63.61 | 79.91 | 73.63 | 83.14 |

### *5.3.1 Model Size Comparison*

We compute the model size for each component in Table 9. The model parameters of RGGID include PixelHop filter parameters, soft classifier parameters, and ensemble classifier parameters. The number of soft classifier parameters is proportional to the number of selected channels. For example, for individual filter banks, if the filter size is $s \times s \times c$ and the selected channel number is $k$, the number of PixelHop filter parameters is $s^2 \times c \times k$. For ensemble schemes, to obtain the total number of PixelHop filter parameters, we sum across the filter banks in the ensemble. For the soft classifier parameters, we train each

Table 9: Model size breakdown.

| Ensemble scheme | No. of PixelHop filter parameters | No. of selected channels | No. of soft classifier parameters | No. of ensemble classifier parameters | Total No. of parameters |
|---|---|---|---|---|---|
| $2 \times 2 \times 3$ only | $12 \times 12$ | 4 | 76k | 40 | 76.2k |
| $3 \times 3 \times 3$ only | $27 \times 27$ | 4 | 76k | 40 | 76.8k |
| $4 \times 4 \times 3$ only | $48 \times 48$ | 4 | 76k | 40 | 78.3k |
| ensemble (2&3) | 873 | 8 | 152k | 40 | 152.9k |
| ensemble (3&4) | 3033 | 8 | 152k | 40 | 155k |
| ensemble (2&4) | 2448 | 8 | 152k | 40 | 154.4k |
| ensemble (all) | 3177 | 12 | 228k | 40 | 231.2k |

Table 10: Model size comparison.

| Method | DenseNet | XceptionNet | InceptionNet | Cozzolino | Auto-Spec | Nataraj (raw dataset) | RGGID |
|---|---|---|---|---|---|---|---|
| Number of parameters | 1.0M | 22.9M | 23.8M | 1k | 21.8M | 730k | 76.2k |

XGBoost classifier with 100 trees, and each tree has a maximum depth of 6. The total number of soft classifier parameters is equal to the number of channels multiplied by 19k. The ensemble classifier is a shallow XGBoost classifier with only 10 trees and each tree has a depth of 1. Thus, the number of ensemble classifier parameters is 40. For the various settings, we see that the model size ranges from 76.2k to 231.2k parameters.

The model sizes of other state-of-the-art fake image detection models on the raw CycleGAN dataset are given in Table 10. DNN models such as DenseNet, InceptionNet and XceptionNet have millions of parameters. A shallow CNN that has two convolutional layers and one fully connected layer was introduced by Cozzolino *et al.* [8]. Its model has only 1k parameters. Auto-Spec [24] uses ResNet-34 as a classification network and has 21.8M parameters. Nataraj *et al.* [18] used a neural network for feature extraction and classification, and its model size is 730k. In contrast, RGGID has a minimum of 76.2k parameters (with the $2 \times 2 \times 3$ filter bank) and a maximum of 231.2k parameters (with the ensemble of all three filter banks). Its model size is significantly smaller than those of DNNs.

### *5.3.2 Computational Complexity*

We measure the training time from scratch on CPU Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz. The average training time for each category is 1.9 hours, yielding a total training time of 19 hours for all 10 categories. Other existing models need GPU and they often rely on pre-trained models.

### *5.3.3 Weak Supervision*

As reported in Table 1, RGGID can achieve an accuracy of 99.0% on the raw image dataset based on 10% of training images from each training category.

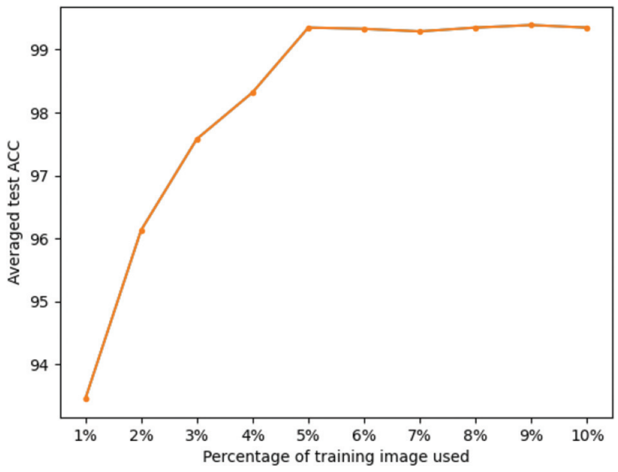


Figure 11: The test accuracy as a function of different percentages of the total training images.

We also conducted experiments using only 1%, 2%, $\cdots$ , 8% and 9% of training data from each semantic category and show the corresponding test accuracies in Figure 11 for the $2 \times 2 \times 3$ filter bank. Its test accuracy reaches 93% even with 1% of the original training images. It converges to 99% using only 5% of the original training images. This shows that RGGID can perform well even under extremely weak supervision.

## 6 Conclusion and Future Work

A green and robust GAN-fake image detector called RGGID was proposed in this work. It was developed under the assumption that GANs fail to generate high-frequency components of real images in high fidelity. Thus, it focuses on complex local regions that have high-frequency components and employs a set of local filters, called filter banks or PixelHops, to extract features. Discriminant channels were identified and their responses were used as features and fed into the XGBoost classifier for soft decision. Finally, various ensemble schemes were adopted to make RGGID adaptive to different semantic categories and robust with respect to compression, resize and additive noise manipulations. It was shown by experimental results that RGGID can maintain good detection performance against these manipulations.

The robustness of RGGID was conducted against the CycleGAN dataset in this work. As future extension, it is interesting to test the robustness of RGGID against different GANs in a cross-GAN setting. That is, we may train the RGGID model solely on real images and ProGAN fake images and then test it on images generated by other GANs.

## References

[1] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[2] H.-S. Chen, M. Rouhsedaghat, H. Ghani, S. Hu, S. You, and C.-C. J. Kuo, "Defakehop: A Light-weight High-performance Deepfake Detector," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2021, 1–6.

[3] H.-S. Chen, K. Zhang, S. Hu, S. You, and C.-C. J. Kuo, "Geo-defakehop: High-performance Geographic Fake Image Detection," *arXiv preprint arXiv:2110.09795*, 2021.

[4] T. Chen and C. Guestrin, "Xgboost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, 785–94.

[5] Y. Chen and C.-C. J. Kuo, "Pixelhop: A Successive Subspace Learning (SSL) Method for Object Recognition," *Journal of Visual Communication and Image Representation*, 70, 2020, 102749.

[6] Y. Chen, M. Rouhsedaghat, S. You, R. Rao, and C.-C. J. Kuo, "Pixelhop++: A Small Successive-Subspace-Learning-based (SSL-based) Model for Image Classification," in *2020 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2020, 3294–8.

[7] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, 8789–97.

[8] D. Cozzolino, G. Poggi, and L. Verdoliva, "Recasting Residual-based Local Descriptors as Convolutional Neural Networks: An Application to Image Forgery Detection," in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, 2017, 159–64.

[9] J. Fridrich and J. Kodovsky, "Rich Models for Steganalysis of Digital Images," *IEEE Transactions on Information Forensics and Security*, 7(3), 2012, 868–82.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," *Advances in Neural Information Processing Systems*, 27, 2014.

[11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image Translation with Conditional Adversarial Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, 1125–34.

[12] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs For Improved Quality, Stability, and Variation," *arXiv preprint arXiv:1710.10196*, 2017.

[13] T. Karras, S. Laine, and T. Aila, "A Style-based Generator Architecture for Generative Adversarial Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, 4401–10.

[14] C.-C. J. Kuo, M. Zhang, S. Li, J. Duan, and Y. Chen, "Interpretable Convolutional Neural Networks via Feedforward Design," *Journal of Visual Communication and Image Representation*, 60, 2019, 346–59.

[15] H. Li, B. Li, S. Tan, and J. Huang, "Detection of Deep Network Generated Images Using Disparities in Color Components. arxiv 2018," *arXiv preprint arXiv:1808.07276.*

[16] F. Marra, D. Gragnaniello, D. Cozzolino, and L. Verdoliva, "Detection of GAN-generated Fake Images Over Social Networks," in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, IEEE, 2018, 384–9.

[17] F. Matern, C. Riess, and M. Stamminger, "Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations," in *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, IEEE, 2019, 83–92.

[18] L. Nataraj, B. S. Mohammed Tajuddin Manhar andf Manjunath, S. Chandrasekaran, A. Flenner, J. H. Bappy, and A. K. Roy-Chowdhury, "Detecting GAN Generated Fake Images Using Co-occurrence Matrices," *Electronic Imaging*, 2019(5), 2019, 532–1.

[19] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "GauGAN: Semantic Image Synthesis with Spatially Adaptive Normalization," in *ACM SIGGRAPH 2019 Real-Time Live!* 2019, 1–1.

[20] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "Cnn-Generated Images are Surprisingly Easy to Spot . . . for Now," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, 8695–704.

[21] N. Yu, L. S. Davis, and M. Fritz, "Attributing Fake Images to GANs: Learning and Analyzing GAN Fingerprints," in *Proceedings of the IEEE/ CVF International Conference on Computer Vision*, 2019, 7556–66.

[22] M. Zhang, Y. Wang, P. Kadam, S. Liu, and C.-C. J. Kuo, "Pointhop++: A Lightweight Learning Model on Point Sets for 3D Classification," in *2020 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2020, 3319–23.

[23] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo, "Pointhop: An Explainable Machine Learning Method for Point Cloud Classification," *IEEE Transactions on Multimedia*, 22(7), 2020, 1744–55.

[24] X. Zhang, S. Karaman, and S.-F. Chang, "Detecting and Simulating Artifacts in GAN Fake Images," in *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, 2019, 1–6.

[25] Y. Zhu, X. Wang, H.-S. Chen, R. Salloum, and C.-C. J. Kuo, "A-pixelhop: A Green, Robust and Explainable Fake-image Detector," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, 8947–51.