# Detection of Diffusion-Generated Images Using Sparse Coding

Daishi TANAKA* and Michiharu NIIMI†
* Graduate School of Computer Science and Systems Engineering,
Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-shi, Fukuoka, 820-8502 Japan
tanaka.daishi864@mail.kyutech.jp
† Faculty of Computer Science and Systems Engineering,
Kyushu Institute of Technology, 680-4, Kawazu, Iizuka-shi, Fukuoka, 820-8502 Japan
niimi@ai.kyutech.ac.jp

***Abstract*—This paper proposes a method for detecting images generated by diffusion models using sparse coding. In the diffusion model, an image can be generated by removing noise from a noisy image. This different generation process from real images leads us to believe that there may be a statistical difference in pixels between the real and the generated images. Specifically, the image is divided into small patch regions, and all patch images are reconstructed using the basis image. In this process, sparse coefficients that contain many zeros are obtained using sparse coding, and features are calculated from the obtained coefficients. Then, a simple discriminator using the features as input is trained with a small number of data to discriminate the diffusion-generated images. In our experiments, we evaluated the proposed method on six datasets created using three diffusion models and two real image datasets. Experiments were also conducted to evaluate the robustness of the proposed method against JPEG compression. Experimental results show that our proposed method is sufficiently robust against JPEG compression with as few as 1800 training data.**

## I. Introduction

In recent years, it has become difficult even for the human eye to distinguish between generated and real images. In addition, as AI that generates images from text, such as Stable Diffusion, has become available to everyone, the number of cases of generated images containing false information being intentionally spread has been increasing. In fact, in Japan in 2022, a false image of typhoon flooding in Shizuoka Prefecture was generated using Stable Diffusion, and in the U.S. in 2023, images of former President Trump being arrested using Midjourney was spread on X (formerly Twitter), respectively.

This ability to generate high-quality images from text has been made possible by the emergence of powerful image generation techniques such as diffusion models. This rapid progress in image generation technology has led to growing concerns about malicious use of the generated images. For this reason, there has been a lot of recent research on diffusion-generated image detection. Bammey et al.[1] proposed a detection method focusing on artifacts in the frequency domain of diffusion-generated images. Wang et al.[2] proposed a detection method using the reconstruction error between the reconstructed image and the original image obtained from a learned diffusion model. As an improvement, Luo et al.[3] proposed a very fast reconstruction-based detection method in the latent space . On the other hand, several detection methods have been proposed that use a large-scale vision-language model CLIP (Contrastive Language-Image Pre-training) as a feature extraction model[4][5]. As evaluation methods for diffusion-generated image detection, Mingjian et al.[6] presented two evaluation methods for detection methods similar to real-world scenarios using a newly constructed large dataset GenImage. As one of the methods, they proposed a method to evaluate detectors on images degraded by JPEG compression and so on. For some of the detection methods[1][4][5] shown earlier, the impact of JPEG compression on image degradation classification has been evaluated, and it can be confirmed that the identification accuracy decreases due to JPEG compression. Similarly, DIRE[2] has also been tested for robustness against JPEG compression, and while it performed well, it requires a large amount of data for detector training, which is an issue. One remaining detection method[3] has not been evaluated for classification of image degradation.

Considering these previous works, we aim to propose a new method for detecting diffusion-generated images that is robust in real-world scenarios with less data and high robustness against JPEG compression. As an approach to our detection method, we focused on the process of generating diffusion-generated images. The diffusion process gradually adds Gaussian noise to the image $x_0$, and finally obtains pure Gaussian noise $x_T$. On the other hand, the inverse diffusion process is a process that, given a pure Gaussian noise $x_T$, gradually removes the noise from it to finally obtain a clear image $x_0$. From this fact, we considered that the method of steganalysis, which discriminates whether an image has been subjected to steganography that embeds invisible information into the image or not, might be effective. In fact, a GAN detector[7] based on pixel co-occurrence matrices has been proposed with reference to steganalysis methods. In this study, we propose a method for detecting diffusion-generated images by sparse coding based on our previous work[8] on steganalysis, especially on universal steganalysis independent of embedding method. The outline of the proposed method is as follows: First, the image is divided into small patch

regions, and all the patch regions are reconstructed using the basis image. During the reconstruction, sparse coding is used to obtain sparse coefficients that contain many zeros, and features are calculated. In our proposed method, we propose two different features. By using a discriminator with each of these features as input, we expect to discriminate between diffusion-generated and real images using a small number of data. We also evaluate the robustness of the discriminator against JPEG compression.

In the following sections of this paper, we describe the proposed method in Section 2, conduct experiments and discuss the results in Section 3, and summarize in Section 4.

## II. Proposed Method for Detecting Diffusion-Generated Images

In the proposed method for detecting diffusion-generated images, sparse coding is performed on patch images, features are calculated based on the calculated sparse coefficients, and a simple discriminator is used to detect diffusion-generated images. As a specific process of this method, the input image is a three-channel full-color RGB image, and the gray-scale image of the G component, which has the greatest influence on the gray-scale component, is targeted for processing, and is divided into $8 \times 8$ patch images by sliding the image one pixel at a time. For all the patch images, sparse coding is performed using 64 basis images and sparse coefficients are calculated. The features are computed based on these coefficients, and a discriminator using the features as input is used to discriminate between diffusion-generated images and natural images. We propose the following two different features as input to the discriminator.

- Sparse Coding Coefficient Statistics (SCCS)
- Sparse Coding Basis Statistics (SCBS)

For the discriminator, we used SVM, which is a popular binary classification model. The following subsections describe the detailed processing of the proposed method.

### A. Sparse Coding

Sparse coding is a method of representing input data as a linear sum of a small number of bases. Fig. 1 shows a schematic diagram of patch reconstruction for sparse coding. In Fig. 1, for a patch image to be processed, sparse coding coefficients are calculated for multiple basis images of the same patch size, and a reconstructed image that approximates the input patch image is computed by calculating a linear sum of the basis images and sparse coding coefficients. In this study, the patch size is $8 \times 8$, and the patch images are extracted by shifting the G component of the image one pixel at a time. For each patch image, sparse coding is performed using 64 basis images to obtain the coefficients.

### B. Sparse Coding Coefficient Statistics (SCCS)

SCCS features are computed from the coefficients based on the results of the two processes, the following six features are computed.

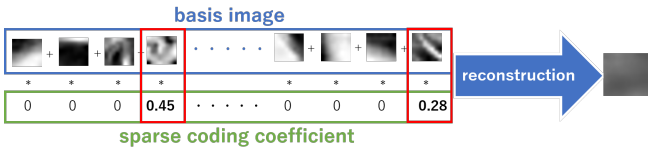


Fig. 1: Schematic of Sparse Coding

- Reconstruction error before and after sparse code shrinkage
- Average of noise percentage before and after sparse code shrinkage
- Kurtosis of the modified histogram
- Skewness of modified histogram
- Mean of modified histogram
- Standard deviation of modified histogram

In the following subsections, the above features are explained for each process.

*1) Feature Extraction Process Using Sparse Code Shrinkage:* Sparse code shrinkage is a method for removing Gaussian noise from non-Gaussian distributed data. The process of denoising by sparse code shrinkage is described below. First, let $c_{ij}$ $(i = 1, 2, \ldots, PN;\ j = 1, 2, \ldots, 64)$ denote the coefficient values arranged in one dimension, where the first argument $i$ is the patch image number, $PN$ is the number of patch images, and the second argument $j$ is the basis image number. The threshold value $val$ for removing noise is calculated as follows:

$$val = \frac{\sqrt{2} \cdot \sigma^2}{d_{scale}} \tag{1}$$

where the noise variance $\sigma$ is the absolute mean of the deviations of the coefficients and $d_{scale}$ is the scale parameter, computed as the square root of the mean square of the coefficients. The coefficients after sparse code shrinkage, denoted as $sc_{ij}$, are calculated based on the thresholded coefficients $c'_{ij}$, where $c'_{ij}$ are obtained by subtracting the threshold value from the absolute value of the coefficients $c_{ij}$. The calculation is defined as:

$$sc_{ij} = \begin{cases} \mathrm{sgn}(c_{ij}) \cdot c'_{ij} & \text{if } c'_{ij} > 0 \\ 0 & \text{if } c'_{ij} \leq 0 \end{cases} \tag{2}$$

where $\mathrm{sgn}(c_{ij})$ obtains the sign of the coefficients $c_{ij}$ before taking the absolute value. The coefficients $sc_{ij}$ after sparse code shrinkage computed in this way are used to compute the following two features:

- Reconstruction error before and after sparse code shrinkage
- Average of noise percentage before and after sparse code shrinkage

For the reconstruction error before and after sparse code shrinkage, we calculate the mean squared error (MSE) of pixel values between the shrinkage reconstruction image computed using $sc_{ij}$, and the original reconstruction image computed using $c_{ij}$. This MSE serves as the first feature. Furthermore,

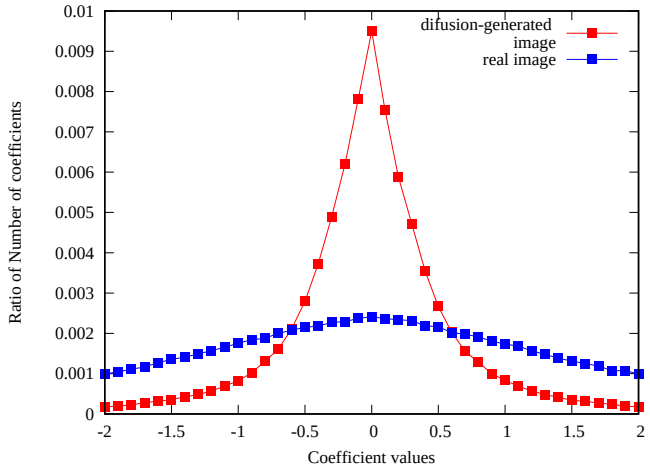


Fig. 2: Examples of modified histograms of diffusion-generated and real images

the second feature, the average noise percentage, is computed using the following formula:

$$\frac{\sum_{i=1}^{PN}\sum_{j=1}^{64}\left|\frac{sc_{ij}-c_{ij}}{c_{ij}}\right|}{N} \tag{3}$$

where $N$ is the number of the coefficients $c_{ij}$. These two features aim to extract characteristics of residual Gaussian noise on diffusion-generated images.

*2) Feature Extraction Process from Modified Histograms:* We introduce a modified histogram of sparse coding coefficients. In this paper, a modified histogram is calculated in the following steps. Firstly, lower and upper limits are set to -20 as the lower limit and +20 as the upper limit to count the number of coefficients with, so we can count the number of sparse coding coefficients between -20 and +20. This range is divided into 4000 bins having 0.01 interval. The bin value is the ratio of the number of coefficients in the bin range divided by the total number of coefficients. For example, a bin value between 0 and 0.01 represents the percentage of the number of coefficients whose values lie within that range. The value of the bin between 0 and 0.01 minus the number of zero coefficients is the new value of the bin between 0 and 0.01. This correction is made to take into account the fact that an extremely large number of zeros in the coefficients can adversely affect the difference in the feature values between the diffusion-generated image and the real images.

Fig. 2 shows examples of the modified histograms of the diffusion-generated image and the real image. In order to effectively capture the differences in the modified histograms in Fig. 2, we use the following four features: kurtosis, skewness, mean, and standard deviation.

Based on the computed modified histogram, for each $n$ bin value $x_1, x_2, ...., x_n$, the mean $\bar{x}$ and standard deviation $std$ are computed. The kurtosis $K$ and the skewness $S$ are also calculated. By using these features, we consider extracting differences in the overall statistics of the coefficients obtained by sparse coding in diffusion-generated images and real images.

### C. Sparse Coding Basis Statistics (SCBS)

In SCBS features, the feature value consists of 64 values computed for each basis image. Specifically, it is calculated using the following formula:

$$\frac{\sum_{i=1}^{PN}|c_{i,j}|}{PN} \quad (j = 1, 2, \ldots, 64) \tag{4}$$

This computation yields a feature value where each value represents the sum of absolute coefficients for each basis image. By using these features, we consider extracting the difference in the coefficient statistics for each basis image between diffusion-generated and real images.

## III. Experiments

The following experiments were conducted to evaluate the discrimination accuracy of SVM with SCCS and SCBS features as input features and their robustness to JPEG compression, respectively. First, we conducted experiments on the basis image and sparsity, which are considered to affect the discrimination accuracy of our method, and compared the accuracy of each. Then, we evaluated the robustness of the SVM trained on uncompressed data against JPEG compression by using test data with Q-values of 90 and 50, respectively.

### A. Dataset

In this experiment, we prepared eight datasets of six diffusion-generated and two real images. As diffusion models, the following three models downloaded from Hugging Face[9] were used to generate the images and create the datasets.

- DeepFloyd/IF-II-L-v1.0
  *DeepFloyd is licensed under the DeepFloyd License, Copyright (c) Stability AI Ltd. All Rights Reserved.*
- emirianJR/epiCRealism[10]
- stabilityai/stable-diffusion2-1[11]

Using these three models, we generated 1000 images each for the category “dog” and “non-dog” from the same prompts. We also collected 1000 real images of the category “dog” from the Open Image Dataset V7[12] using FiftyOne, and 1000 real images including various subjects taken in RAW mode. TABLE I lists the datasets used in this experiment. Fig. 3 shows examples of images generated by each diffusion-generated model using the same prompt as input. In the experiments, the aspect ratio was fixed and the diffusion-generated images were resized to $256\times256$ and the real images were resized to 256 on the longest side before processing.

### B. Experiments on Different Dictionaries

In this experiment, we evaluate the effect of using dictionaries consisting of different basis images on the accuracy of the proposed method. Specifically, we use dictionaries trained on two different datasets and compute the coefficients by sparse coding for each dataset. Based on the coefficients calculated here, SCCS and SCBS features are obtained, and each SVM is trained and evaluated by a 10-fold cross-validation. In the

TABLE I: Dataset List

| | Diffusion models | Size | Class | Number of data | Identifier |
|---|---|---|---|---|---|
| Diffusion-generated images | DeepFloyd IF | $256 \times 256$ | dog | 1000 | IFd |
| | | $256 \times 256$ | non-dog | 1000 | IFnd |
| | epiCRealism | $512 \times 512$ | dog | 1000 | ECd |
| | | $512 \times 512$ | non-dog | 1000 | ECnd |
| | Stable Diffusion V2-1 | $768 \times 768$ | dog | 1000 | SDd |
| | | $768 \times 768$ | non-dog | 1000 | SDnd |
| Real images | - | $3264 \times 2448$ or $2448 \times 3264$ | dog | 1000 | NLd |
| | - | long side: 1024 | non-dog | 1000 | NLnd |

Fig. 3: Images generated with "A vintage train chugs along a scenic route, nostalgic and scenic" as a prompt. From left, generated by DeepFloyd IF, epiCRealism, and Stable Diffusion V2-1.

following subsections, we describe the specific ways in which the experiment is performed.

*1) Dictionaries:* In this experiment, two dictionaries are prepared. Let us denote them as dictionary A (dicA) and dictionary B (dicB), respectively. DicA was trained using IFd, a dataset of diffusion-generated images. DicB is trained on NLnd, a dataset of real images. The method of creating the dictionaries is as follows. First, 100 images are randomly selected from the dataset. For each selected image, 2000 patch images of $8 \times 8$ are extracted at random locations. Thus, 64 basis images were trained from a total of 200,000 patch images. In the specific program, spams.trainDL in python were used, the parameter of the optimization relation was set to lambda1 = 0.015, and the number of training cycles was set to 100.

*2) Calculation of Sparse Coding Coefficients:* Sparse coding is performed using dicA and dicB, and the coefficients are computed for each. For the specific execution, spams.lasso in python was used and the parameter of the optimization relation was set to lambda1 = 0.015. Sparse coding was performed on each patch of the image by shifting the image by one pixel, and the coefficients were calculated.

*3) Discriminator Learning:* For each dataset, SCCS and SCBS features are computed from the coefficients obtained using each of dicA and dicB separately. For all 12 pairs of diffusion-generated images and real images, we trained SVMs using the standardized SCCS and SCBS features as inputs, respectively. The SVMs were trained and evaluated by 10-fold cross-validation. That is, the average accuracy of SVMs trained 10 times on a dataset of 2000 data sets, with 1800 training data and 200 test data randomly changed. TABLE II and III show the results for dicA and dicB using SVM, respectively.

TABLE II: Evaluation accuracy of discriminators of SCCS features in different dictionaries

| | dictionary | IFd | IFnd | ECd | ECnd | SDd | SDnd |
|---|---|---|---|---|---|---|---|
| NLd | dicA | **0.958** | **0.847** | **0.867** | **0.697** | **0.834** | **0.824** |
| | dicB | 0.956 | 0.834 | 0.850 | 0.672 | 0.819 | 0.819 |
| NLnd | dicA | **0.988** | **0.951** | **0.940** | **0.834** | **0.837** | **0.696** |
| | dicB | 0.984 | 0.937 | 0.932 | 0.809 | 0.830 | 0.692 |

TABLE III: Evaluation accuracy of discriminators of SCBS features in different dictionaries

| | dictionary | IFd | IFnd | ECd | ECnd | SDd | SDnd |
|---|---|---|---|---|---|---|---|
| NLd | dicA | **0.971** | **0.964** | **0.939** | **0.903** | **0.850** | **0.869** |
| | dicB | 0.964 | 0.955 | 0.912 | 0.877 | 0.842 | 0.856 |
| NLnd | dicA | **0.991** | **0.984** | **0.972** | **0.953** | **0.886** | **0.759** |
| | dicB | 0.985 | 0.979 | 0.964 | 0.920 | 0.878 | 0.719 |

Although there is no significant difference in accuracy between the different dictionaries, there is a slight improvement in accuracy using dicA trained with IFd for all datasets. Based on these results, we trained the dictionaries using IFd in the subsequent experiments.

### C. Experiments on Sparsity

In order to investigate the effect of sparsity of sparse coding coefficients on identification accuracy, we made the following changes compared to III-B.

- Increased number of training data for the dictionary
- Increased number of dictionary training iterations
- Change sparsity parameters

We newly used IFd and changed the parameter lambda1 = 0.2 during training to improve sparsity compared to III-B. The number of training cycles was increased to 2000 to learn dictionary C (dicC). The number of patches was increased fivefold to 1,000,000 patches from the 200,000 patches used in dicA. In addition, the calculation of the sparse coding coefficients was modified to improve the sparsity of the coefficients by setting lambda1 = 0.2. Table IV and table V show the results of training SVMs with SCCS and SCBS features as inputs, respectively. The accuracies shown in their tables were obtained by 10-fold cross-validation. Table IV shows that improving the sparsity of the coefficients in the SCCS features decreased the overall accuracy. Table V shows that improving the sparsity of the coefficients in the SCBS features improves the overall accuracy. This suggests that increasing the

TABLE IV: Evaluation accuracy of discriminators of SCCS features at different sparsity

| | lambda1 | IFd | IFnd | ECd | ECnd | SDd | SDnd |
|---|---|---|---|---|---|---|---|
| NLd | 0.015 | 0.958 | **0.847** | **0.867** | 0.697 | **0.834** | **0.824** |
| | 0.2 | **0.963** | 0.846 | **0.867** | **0.733** | 0.763 | 0.796 |
| NLnd | 0.015 | **0.988** | **0.951** | **0.940** | **0.834** | **0.837** | **0.696** |
| | 0.2 | 0.981 | 0.891 | 0.908 | 0.752 | 0.776 | 0.691 |

TABLE V: Evaluation accuracy of discriminators of SCBS features at different sparsity

| | lambda1 | IFd | IFnd | ECd | ECnd | SDd | SDnd |
|---|---|---|---|---|---|---|---|
| NLd | 0.015 | 0.971 | 0.964 | **0.939** | 0.903 | 0.850 | 0.869 |
| | 0.2 | **0.978** | **0.967** | 0.934 | **0.906** | **0.874** | **0.900** |
| NLnd | 0.015 | **0.991** | **0.984** | 0.972 | **0.953** | 0.886 | 0.759 |
| | 0.2 | 0.989 | 0.983 | **0.976** | 0.929 | **0.926** | **0.800** |

sparsity of the coefficients decreases the statistical difference of the overall coefficients between the diffusion-generated images and real images, while increasing the statistical difference of the coefficients for each basis image.

### *D. Evaluation Experiments on JPEG Compression*

In this experiment, we evaluated the robustness of JPEG compression in the proposed method. For both SCCS and SCBS features, we trained SVM by computing features using dicC under the same conditions as in III-C. Then, we applied JPEG compression to 200 pieces of test data used for evaluating the accuracy of SVM with Q-values of 90 and 50, and evaluated SVM using each JPEG compressed test data. In other words, we conducted a realistic evaluation experiment on unknown images with JPEG compression. In order to obtain the actual accuracy, we computed the average accuracy of 10 times while randomly changing the training and test data. The results are shown in TABLE VI and VII. The results in TABLE VI and VII show that both features are sufficiently robust to JPEG compression. In particular, the SCCS features maintain their accuracy even with higher JPEG compression.

### *E. Discussion*

As shown in Table VI, the SVM with SCCS features as input does not lose accuracy when comparing the results for the uncompressed test data and the JPEG-compressed test data with Q value = 50. Therefore, it is hypothesized that the distribution of coefficients obtained by sparse coding, i.e., the modified histogram, is less affected by high-frequency components.

To investigate the influence of high-frequency components in the distribution of coefficients obtained by sparse coding, we conducted comparison experiments between high-frequency images containing many high-frequency components and JPEG-compressed high-frequency images in which the information on high-frequency components was reduced by applying JPEG compression to the high-frequency images. Firstly, 100 high-frequency images of $256 \times 256$ were collected

TABLE VI: Evaluation accuracy of JPEG compression in discriminators of SCCS features

| | Q-value | IFd | IFnd | ECd | ECnd | SDd | SDnd |
|---|---|---|---|---|---|---|---|
| NLd | 100 | 0.962 | 0.842 | 0.858 | 0.752 | 0.766 | **0.797** |
| | 90 | 0.953 | 0.832 | 0.860 | 0.748 | 0.778 | 0.791 |
| | 50 | **0.967** | **0.897** | **0.887** | **0.801** | **0.782** | 0.788 |
| NLnd | 100 | 0.975 | 0.893 | **0.905** | 0.743 | 0.783 | 0.698 |
| | 90 | 0.972 | 0.899 | 0.902 | **0.749** | **0.788** | **0.703** |
| | 50 | **0.976** | **0.911** | 0.895 | 0.715 | 0.782 | 0.701 |

TABLE VII: Evaluation accuracy of JPEG compression in discriminators of SCBS features

| | Q-value | IFd | IFnd | ECd | ECnd | SDd | SDnd |
|---|---|---|---|---|---|---|---|
| NLd | 100 | **0.983** | **0.954** | **0.936** | **0.909** | **0.874** | **0.898** |
| | 90 | 0.972 | 0.947 | 0.932 | 0.908 | 0.867 | 0.895 |
| | 50 | 0.967 | 0.947 | 0.919 | 0.899 | 0.782 | 0.877 |
| NLnd | 100 | 0.989 | 0.989 | **0.979** | 0.918 | **0.931** | 0.807 |
| | 90 | **0.990** | **0.990** | **0.979** | **0.924** | 0.925 | **0.819** |
| | 50 | 0.989 | 0.985 | 0.976 | 0.918 | 0.876 | 0.778 |

for the NLnd dataset. Specifically, for all 1000 images in NLnd, a region of $256 \times 256$ was cut out by sliding the image by 256 pixels in height and width, and the portion of the image that was less than $256 \times 256$ at the edge was ignored. In other words, since the image size of NLnd is 3264px on the long side and 2448px on the short side, 108 images can be collected from a single image, or 108,000 $256 \times 256$ images in total. In addition, 100 JPEG-compressed images were created for these 100 images, with Q value = 50. The average of the mean square error of the pixel values of the uncompressed and JPEG-compressed images for all 100 images was calculated to be 58.4. This value indicates that considerable image degradation has occurred due to JPEG compression, i.e., the 100 uncompressed images collected are high-frequency images that contain many high-frequency components. Sparse coding was performed on these 200 images using dicC under the same conditions as in III-D to obtain the coefficients. Using these coefficients, a modified histogram was created to see the distribution of the coefficients. In this experiment, Bhattacharyya distance was used as a measure for comparing the similarity of histograms. Bhattacharyya distance $D_B$ is defined as

$$D_B(p,q) = -\ln\left(\sum_i \sqrt{p(i)q(i)}\right) \tag{5}$$

where $p(i)$ and $q(i)$ denote the probability values of the $i$-th bin of two histograms, respectively. This distance quantitatively evaluates the degree of overlap of the histograms; the greater the overlap, the smaller the value. In other words, a value closer to 0 indicates that the histograms are in agreement. For 100 pairs of uncompressed and JPEG-compressed high-frequency images, the mean Bhattacharyya distance of the histograms is $1.62 \times 10^{-2}$ and the variance is $1.41 \times 10^{-5}$. To show that the histograms are visually close, the histogram computed from the high-frequency image used in this experiment is shown

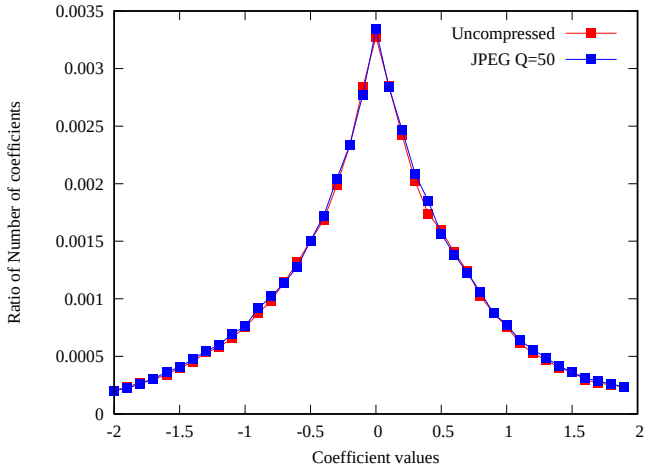


Fig. 4: Modified histograms of the uncompressed and JPEG-compressed high-frequency images

in Fig. 4. Based on these results, it can be said that the distribution of sparse coding coefficients is fairly insensitive to high-frequency components.

To confirm the same thing in the 6-dimensional feature space in the SCCS features, we conducted an experiment using SVM with SCCS features as input to discriminate between uncompressed images and JPEG compressed images for the 200 images mentioned earlier. The average accuracy obtained from the 10-fold cross-validation was 0.515, confirming that the SCCS features are difficult to discriminate. In other words, as with the distribution of coefficients in the modified histogram, the SCCS features are not easily affected by high-frequency components.

From the previous experiments, it is confirmed that the sensitivity of SVM with SCCS features as input is very low to changes in high-frequency components. This means that the method using SCCS features would be able to maintain the discrimination accuracy in the robustness experiments against JPEG compression.

## IV. Conclusion

In this paper, we proposed a new method for detecting diffusion-generated images from real images using features extracted by sparse coding. One of the most notable features of this method is that it is sufficiently robust to JPEG compression with a small amount of training data. In particular, high robustness was confirmed for the SCCS features. Based on these results, we conducted additional experiments and confirmed that the distribution of sparse coding coefficients was not easily affected by high-frequency components, and that high robustness with respect to JPEG compression was obtained by computing features using the distribution of sparse coding coefficients.

For future prospects, we believe that new features are needed that can achieve higher accuracy than the current ones on a wide range of data sets while maintaining high robustness against image degradation such as JPEG compression by using the distribution of sparse coding coefficients.

## References


[1] Q. Bammey, "Synthbuster: Towards Detection of Diffusion Model Generated Images," *IEEE Open Journal of Signal Processing*, 2023.

[2] Z. Wang, J. Bao, W. Zhou, *et al.*, *DIRE for Diffusion-Generated Image Detection*, 2023. arXiv: 2303.09295v1 [cs.CV].

[3] Y. Luo, J. Du, K. Yan, and S. Ding, "LaRE^2: Latent Reconstruction Error Based Method for Diffusion-Generated Image Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 17 006–17 015.

[4] D. Cozzolino, G. Poggi, R. Corvi, M. Nießner, and L. Verdoliva, *Raising the Bar of AI-generated Image Detection with CLIP*, 2024. arXiv: 2312.00195 [cs.CV].

[5] S. A. Khan and D.-T. Dang-Nguyen, "CLIPping the Deception: Adapting Vision-Language Models for Universal Deepfake Detection," in *Proceedings of the 2024 International Conference on Multimedia Retrieval*, 2024, pp. 1006–1015.

[6] M. Zhu, H. Chen, Q. Yan, *et al.*, *GenImage: A Million-Scale Benchmark for Detecting AI-Generated Image*, 2023. arXiv: 2306.08571 [cs.CV].

[7] L. Nataraj, T. M. Mohammed, S. Chandrasekaran, *et al.*, *Detecting GAN generated Fake Images using Co-occurrence Matrices*, 2019. arXiv: 1903 . 06836 [cs.CV].

[8] M. Niimi and H. Noda, "An application of Sparse Code Shrinkage to image steganalysis based on supervised learning," in *2011 18th IEEE International Conference on Image Processing*, IEEE, 2011, pp. 1941–1944.

[9] *Hugging Face*. [Online]. Available: https://huggingface.co/.

[10] C. O. RAIL-M, *CreativeML Open RAIL-M license*, Hugging Face. [Online]. Available: https://huggingface.co/spaces/CompVis/stable-diffusion-license.

[11] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis With Latent Diffusion Models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 10 684–10 695.

[12] *Open Images Dataset V7*. [Online]. Available: %5Curl%7Bhttps://storage.googleapis.com/openimages/web/index.html%7D.