DeepFake Detection based on Noise Residuals

Minh Thong DOI[1,2] Jan BUTORA[2] Vincent ITIER[1,2] Jérémie BOULANGER[2] Patrick BAS[2]

[1]IMT Nord Europe, Institut Mines-Télécom, Centre for Digital Systems, F-59000 Lille, France

[2]Centre de Recherche en Informatique, Signal et Automatique de Lille, Avenue Henri Poincaré, 59655 Villeneuve d'Ascq, France

Résumé – Les « deepfakes » posent des problèmes majeurs, notamment en ce qui concerne la fraude, la désinformation et la falsification de preuves. Comme les deepfakes deviennent de plus en plus répandus, il est essentiel de disposer de méthodes de détection efficaces. Nous présentons DJIN [5], un modèle de détection des deepfakes qui conserve les composantes de bruit en évitant les couches de mise en commun dans les étapes initiales. Pré-entraîné sur ImageNet pour la stéganographie en utilisant la version JIN, DJIN surpasse CoDE [2] et CLIP [7] et est le meilleur parmi tous les détecteurs mentionnés pour l'ensemble de données In-Distribution. DJIN est très efficace pour traiter des images de haute qualité et des images de différentes tailles. Étant donné que les générateurs de deepfake produisent généralement des résultats de haute qualité, une analyse d'explicabilité révèle que DJIN exploite le bruit de l'image en se concentrant sur les zones plus sombres dans les images réelles et sur les zones plus claires dans les images générées.

Abstract – Deepfakes pose major challenges, especially regarding fraud, misinformation, and evidence tampering. As deepfakes become more prevalent, effective detection methods are essential. We introduce DJIN [5], a deepfake detection model that retains noise components by avoiding pooling layers in the initial stages. Pre-trained on ImageNet for steganography using the JIN version, DJIN outperforms CoDE [2] and CLIP [7] and is the best among all detectors mentioned for the In-Distribution dataset. DJIN is highly effective in handling high-quality images and processing images of various sizes. Since deepfake generators typically produce high-quality outputs, an explainability analysis reveals that DJIN leverages image noise by focusing on darker areas in real images and brighter areas in generated ones.

1 Introduction

Deepfakes have been developing dramatically in recent years, from the generation of realistic human faces to the creation of convincing audio and video content. Since the introduction of GANs [8] (Generative Adversarial Networks) in 2014, the quality of the generated contents has been increasing exponentially, and the ability to detect fake content has become a real challenge. Although deepfakes provide a powerful tool for creating realistic contents, which can be used mainly for entertainment nad marketing purposes, malicious use of deepfakes can have serious consequences, such as the spread of fake news, the creation of fake evidence, or the impersonation of individuals.

While deepfakes are widely accessible to fit the needs of the general public, it raises the an important ethical question: how can we trust the content that we see and hear?

Generated images can be distinguished through 2 main components: the noise residual and the semantic features. Generators are trained to recreate images by learning patterns and features from the training data, rather than understanding the physical world. These drawbacks can still be surpassed with the advance of deepfakes models. However, any generator will leave a noticeable trace inherent by the generation process in the output image, even when it is invisible to the human eyes [6]. These traces are different from typical noise in images which allows us to detect generated images. In some cases, pre or post-processing techniques can have a negative impact on the noise residual as it removes some high-frequency components in the image which are essential for detection.

The recent benchmark from PEReN outlines 3 state-of-the-art models for deepfake detection: CoDE [2], Corvi *et al.* [6], CLIP [7]. Corvi *et al.* is a CNN based on ResNet50 and which has the particularity to avoid down-sampling in the first layer, as down-sampling can reduce the performance of the model [11]. CLIP features, used initially for computer vision tasks to provide an embedding where image textual descriptions and image contents are close using a contrastive procedure, are in [7] used as inputs of a classifier to detect generated images. CoDE, built on a Vision Transformer, analyzes relationships between different image regions to detect anomalies in generated images. A contrastive learning strategy is used in order to find an embedding which brings closer the class of generated images or the class of natural images, but also spread apart the two classes. Local and global features are used.

In this paper, we present the DJIN deepfake detector [5], a simple and efficient convolutional neural network that extracts the noise residual in the image. The most important question that we want to answer is: what are the advantages and drawbacks of noise-based detectors compared to semantic-based detectors? To answer that question, we compare DJIN's performance to other models using different datasets: our validation dataset and an out-of-distribution dataset. We show that if we include many different generators in our training data, the DJIN detector may be the best option.

The paper is organized as follow: in the next section, deepfake detectors and image datasets are described. Section 4 analyzes the results obtained on in-distribution and out-of-distribution datasets, and Section 5 highlights the advantages and drawbacks of noise-based detectors compared to semantic-based detectors.

2 The DJIN architecture

	Conv layers	Conv layers before subsampling	Params	Receptive Field before sub	Receptive Field	Pretrained database
DJIN	26	14	5.3M	31	179	ImageNet for steganography
Corvi et al.	17	4	23.5M	15	113	ImageNet for object classification

Table 1 – Comparison of DJIN and Corvi *et al.* architectures in terms of convolutional layers, parameters, receptive fields, and training datasets.

Steganalysis detectors based on deep learning have been extremely successful in detection of imperceptible changes in the noise component of digital images as it is the case for steganalysis.

We select the JIN detector, which is a specifically initialized SRNet [4], a CNN designed for steganalysis. One specificity of the SRNet is that the first 12 convolutional layers are not using any pooling, thus allowing the detector to focus on noise components. The acronym JIN (J-UNIWARD ImageNet) then means that it has been pre-trained on almost a million images embedded with steganography [1], which has been shown to provide a very good initialization strategy in tasks related to steganalysis and image forensics [5]. By using JIN to initialize our training procedure, we consequently enforce the analysis of noise residuals over the image semantic. As we can see in Section 4.3, our explainability analysis confirms this rationale. Note also that, as a CNN, this model can process during inference images of any size without resizing, preserving critical details and high-frequency components. It is important to avoid as resizing may erase high-frequency components in the image [6]. We can also see in Table 1, Corvi et al. has 4.5 times more parameters than DJIN, which may cause memory issues when training with large images on hardware with limited resources.

3 Experimental setup

3.1 Image datasets

We consider two datasets in this work, ID (in-distribution) dataset, used for training and testing the detectors, and OOD (out-of-distribution) dataset to test the detectors on images generated with previously unseen generators.

The real images in the ID datasets are QF 100 JPEG images of size at least 2048×2048 downloaded from Flickr. We randomly select 9000 images and crop them to size 2048×2048. Subsequently, we split them into training and testing sets of 7200, and 1800, respectively. Note that we do not use a validation set as we do not modify any hyperparameters of any of the detectors.

For the generated images of the ID dataset, we use the Synthbuster dataset [1], consisting of 9 image generators (DALL·E 2, DALL·E 3, Firefly, Glide, Midjourney-v5, and four Stable Diffusion versions: 1.3, 1.4, 2, XL) each providing 1000 images of variable sizes. We use 800 images of each generator for the training, and the rest for testing.

To evaluate the detectors on unseen generators, we manually generated images from 7 different generators to form the OOD dataset, namely Genmo, Bing Creator, Deep Dream, Google SGE, Meta AI, Runway, and Shutterstock. We generated roughly 100 images for each generator, with an exception at Deep Dream where we only used 41 images. The OOD images have variable size between 700×700 to 1600×1600.

3.2 Detectors

We will demonstrate in this work that steganalysis detectors can be also very efficient in the detection of generated images since we can assume that the noise component of such images has different properties than natural ones.

The DJIN, CoDE, Corvi *et al.*, and CLIP models are then fine-tuned on the ID dataset. The training process is conducted with a batch size of 64, an initial learning rate of 10^{-3}, a weight decay of 2×10^{-4} for 500 epochs, with early stopping when the learning rate drops below 2×10^{-7}. We are using ReduceLROnPlateau for the learning rate scheduler and Adamax optimizer.

For Corvi *et al.* [6], we use the ResNet50 with a dropout rate 0.5, a padding of 1, and a stride of 1 in the first convolutional layer, which avoids down-sampling in the first layer [9].

For CoDE [2] and CLIP [7], the required input size is fixed at 224×224 due to their architectural constraints. For a fair comparison, we train all the detectors at this image resolution. To increase the amount of training data, we therefore include several augmentations. First, we apply D4 transformations. Next, for DJIN and Corvi *et al.*, one of the following transformations is used (each with 25% probability): `RandomResizeCrop`, `RandomCrop`, `Resize`, and a composition of resizing (with a random factor between 0.5 and 2) followed by `RandomCrop`. For CLIP and CoDE, we aim to preserve the full content of the image; therefore, only resizing using Lanczos4 interpolation is applied. Furthermore, with a 50% probability, we apply JPEG compression with a random Quality Factor (QF) between 40 and 100. The final image is then normalized depending on the normalization of the pretrained model (DJIN divides the value by 255, Corvi *et al.* and CoDE use ImageNet normalization). Note that while the real images are already JPEG compressed with QF 100, these augmentations will, in many cases, destroy detectable traces of this compression, thus avoiding a possible bias between the real and generated images. However, in future work, we will explore datasets of uncompressed images.

Note that while Corvi *et al.* and DJIN can train on larger images, we do not use this strategy. However, we will show that testing these detectors on the images with their original resolutions can bring additional performance boosts.

4 Results

We now present the results of the detectors on the ID, and OOD datasets. We consider uncompressed images, and JPEG compression with three QFs: 40, 65, and 90. As we decrease the quality factor, we can expect more high-frequency components to be removed from the image. Consequently, we expect

1. The JIN initialization weights are available at `https://janbutora.github.io/assets/scripts/JIN_SRNet.zip`

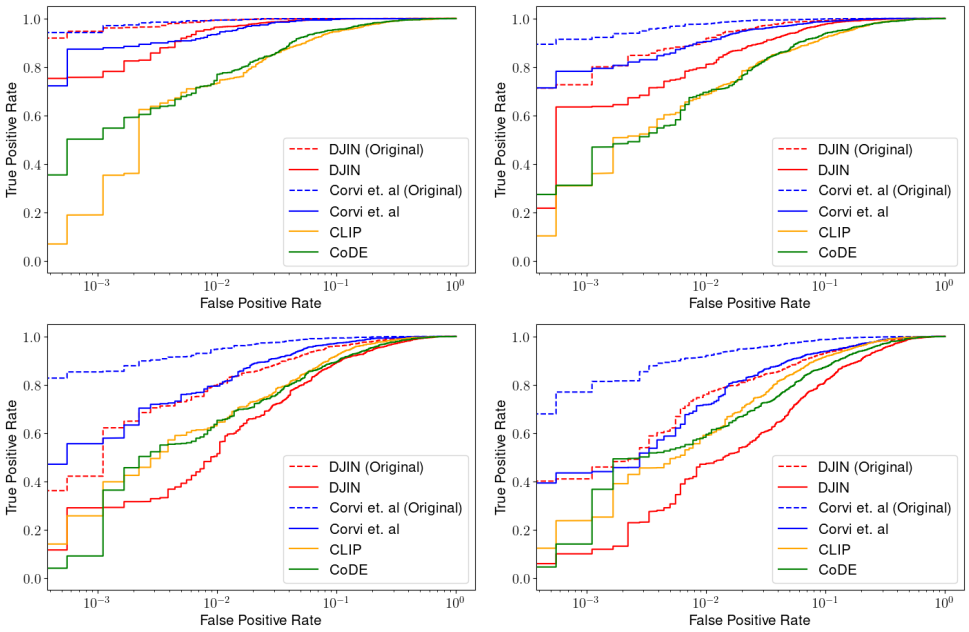

Figure 1 – ROC curves of DJIN, CoDE, Corvi *et. al.*, and CLIP on uncompressed ID dataset (top left), and compressed with QF 90 (top right), 65 (bottom left), and 40 (bottom right). We include DJIN and Corvi *et. al.* results on original resolutions.

the noise-based detectors (DJIN, Corvi *et al.*) to have more difficulties to detect the deepfake images than the semantic-based detector (CLIP and CoDE).

4.1 In-distribution dataset

Based on the ROC curves in Figure 1, we can observe that DJIN has the second best performance for uncompressed and QF 90 images. For the lower QFs, DJIN becomes less efficient than CLIP and CoDE, with bigger difference at QF 40. We note that Corvi *et al.* has consistently the best performance.

However, as mentioned before, the CNN-based detectors can test images of any size, without refining the detectors. We observe significant gains for both DJIN and Corvi *et al.* when testing on images with their original sizes. Notably, DJIN and Corvi *et al.* outperforms other detectors considered, demonstrating a big advantage of CNN-based detectors in terms of performance and flexibility.

4.2 Out-of-distribution dataset

With the OOD dataset, we test on 7 previously generators, and the results are shown in Table 2. CLIP and DJIN show the best performance among the models, achieving 89.9% and 89.12% accuracy respectively for uncompressed images, compared to 83.97% for Corvi *et al.*, and 81.8% for CoDE. However, as the quality factor decreases, DJIN and Corvi *et al.*'s accuracy drop more quickly than other models while CLIP maintains the accuracy among different quality factors. When we test DJIN with the image size of 512x512, the accuracy also improves but with original images, the accuracy drops significantly. It still needs further investigations to understand the reason behind this drop. However, the results suggests that for noise-based detectors, bigger images can provide more useful information about the noise component to detect the deepfake images.

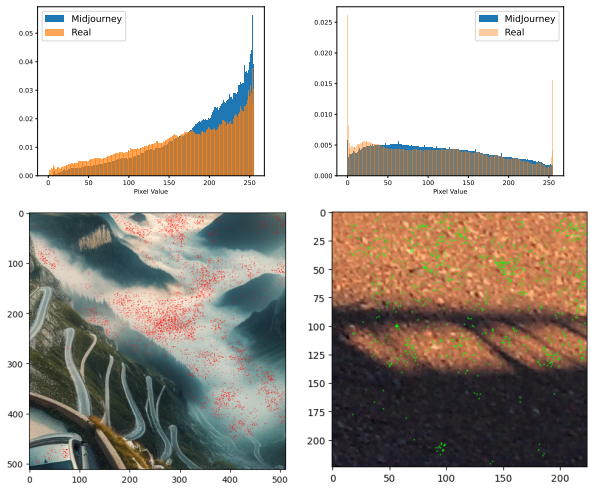

Figure 2 – Probability of the top 1% most influential pixels according to Integrated Gradients w.r.t the total pixels of each value (0-255) (top left). Histogram of pixels' values in real images and Midjourney (top right). Integrated Gradients of midjourney sample image (bottom left), real image (bottom right).

4.3 Explainability

We want to understand the behavior of DJIN when detecting the deepfake images. We use the Integrated Gradients (IG) method to compute the importance of each pixel in the image which explains the contribution of input's features to the prediction of the model [10]. We compute the IG for the real images and the Midjourney generator, and only choose 1% most influential pixels. The results are shown in figure 2. We can see that the higher pixel values are more important for the DJIN detector in making the decision as there is a linear relation between the pixel values and the noise variance [3]. For the real images, the model has more focus on dark areas

Out-Of-Distribution									
DJIN (train on 224x224)		alpha_genmo	bingcreator	deepdream	google_sge	metaAI	runway	shutterstock	Accuracy (224/512/original)
Uncompressed	Real	3	15	11	15	20	10	0	89.12% / **90.74%** / 87.5%
	Fake	97	85	30	89	80	125	100	
QF = 90	Real	14	16	9	14	42	10	3	84.12% / **86.18%** / 80.29%
	Fake	86	84	32	90	58	125	97	
QF = 65	Real	23	41	15	33	37	16	2	75.44% / **81.03%** / 72.35%
	Fake	77	59	26	71	63	119	98	
QF = 40	Real	28	16	17	44	32	17	3	76.91% / **78.24%** / 72.21%
	Fake	72	84	24	60	68	118	97	
Corvi *et al.* (train on 224x224)		alpha_genmo	bingcreator	deepdream	google_sge	metaAI	runway	shutterstock	Accuracy (224)
Uncompressed	Real	5	5	8	41	36	14	0	83.97%
	Fake	95	95	33	63	64	121	100	
QF = 90	Real	12	6	10	40	47	15	0	80.88%
	Fake	88	94	31	64	53	120	100	
QF = 65	Real	20	18	15	50	44	21	0	75.29%
	Fake	80	82	26	54	56	114	100	
QF = 40	Real	26	15	20	67	42	30	1	70.44%
	Fake	74	85	21	37	58	105	99	
CoDE (train on 224x224)		alpha_genmo	bingcreator	deepdream	google_sge	metaAI	runway	shutterstock	Accuracy (224)
Uncompressed	Real	12	15	6	49	24	15	3	81.8%
	Fake	88	85	35	55	76	120	97	
QF = 90	Real	17	13	7	51	22	19	4	80.4%
	Fake	83	87	34	53	78	116	96	
QF = 65	Real	17	13	5	47	15	19	1	82.8%
	Fake	83	87	36	57	85	116	99	
QF = 40	Real	31	20	12	50	22	29	1	75.7%
	Fake	69	80	29	54	78	106	99	
CLIP (train on 224x224)		alpha_genmo	bingcreator	deepdream	google_sge	metaAI	runway	shutterstock	Accuracy (224)
Uncompressed	Real	7	3	6	35	12	6	0	89.9%
	Fake	93	97	35	69	88	129	100	
QF = 90	Real	11	3	8	26	7	7	0	90.9%
	Fake	89	97	33	78	93	128	100	
QF = 65	Real	22	7	12	49	20	14	0	81.8%
	Fake	78	93	29	55	80	121	100	
QF = 40	Real	22	11	8	36	14	11	0	85.0%
	Fake	78	89	33	68	86	124	100	

Table 2 – Performance comparison of DJIN, Corvi *et al.*, CoDE, and CLIP on out-of-distribution datasets with different JPEG compression levels. All models are trained on 224x224 resolution. The accuracy is calculated on 224x224 images for all models, and also on 512x512 and original images for DJIN.

while the Midjourney generator has more focus on bright areas. This can be explained as the generator usually provides studio quality images so it will be brighter than the real images (top right of figure 2 where we can see a lot of dark areas in the real image). We notice also that the classifier focuses on image noise, not on the content.

5 Conclusion

This work presents a deepfake detector that is based on noise residuals. We compare the performance of DJIN with that of various state-of-the-art models across different scenarios. DJIN stands out as an optimal solution in terms of resource efficiency and flexibility, as it has a significantly lower number of parameters than Corvi *et al.* (5.3M vs. 23.5M parameters). It takes only 7 GB of VRAM to train it on a single GPU with an image size of 224 x 224. This model can perform better when testing with larger images. While experimental results vary across conditions, DJIN performs best with high-quality images. Meanwhile, semantic-based detectors like CLIP and CoDE offer better consistency across different JPEG compression levels due to their contextual approach. Our findings suggest that noise- and semantic-based detectors complement each other, providing a balanced solution that mitigates their respective limitations.

6 Acknowledgement

This work was also supported by a French government grant managed by the *Agence Nationale de la Recherche* under the France 2030 program, reference ANR-22-PECY0011 and the project ANR-23-IAS4-0004.

References

[1] Quentin Bammey. Synthbuster: Towards detection of diffusion model generated images. *IEEE Open Journal of Signal Processing*, 5:1–9, 2024.

[2] Lorenzo Baraldi, Federico Cocchi, Marcella Cornia, Lorenzo Baraldi, Alessandro Nicolosi, and Rita Cucchiara. Contrasting Deepfakes Diffusion via Contrastive Learning and Global-Local Similarities. In *Proceedings of the European Conference on Computer Vision*, 2024.

[3] Patrick Bas. Steganography via cover-source switching. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2016.

[4] Mehdi Boroumand, Mo Chen, and Jessica Fridrich. Deep residual network for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 14(5):1181–1193, 2019.

[5] Jan Butora, Yassine Yousfi, and Jessica Fridrich. How to pretrain for steganalysis. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, page 143–148, New York, NY, USA, 2021. Association for Computing Machinery.

[6] Riccardo Corvi, Davide Cozzolino, Giada Zingarini, Giovanni Poggi, Koki Nagano, and Luisa Verdoliva. On the detection of synthetic images generated by diffusion models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[7] Davide Cozzolino, Giovanni Poggi, Riccardo Corvi, Matthias Nießner, and Luisa Verdoliva. Raising the bar of ai-generated image detection with clip, 2023.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[9] Diego Gragnaniello, Davide Cozzolino, Francesco Marra, Giovanni Poggi, and Luisa Verdoliva. Are gan generated images easy to detect? a critical analysis of the state-of-the-art, 2021.

[10] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017.

[11] Yassine Yousfi, Jan Butora, Eugene Khvedchenya, and Jessica Fridrich. Imagenet pre-trained cnns for jpeg steganalysis. In *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2020.