Deepfake Network Architecture Attribution

Tianyun Yang[1,2*], **Ziyao Huang**[1,2*], **Juan Cao**[1,2†], **Lei Li**[1,2], **Xirong Li**[3]

[1] Key Lab of Intelligent Information Processing, Institute of Computing Technology, CAS, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Key Lab of Data Engineering and Knowledge Engineering, Renmin University of China
{yangtianyun19z,huangziyao19f,caojuan,lilei17b}@ict.ac.cn, xirong@ruc.edu.cn

Abstract

With the rapid progress of generation technology, it has become necessary to attribute the origin of fake images. Existing works on fake image attribution perform multi-class classification on several Generative Adversarial Network (GAN) models and obtain high accuracies. While encouraging, these works are restricted to model-level attribution, only capable of handling images generated by seen models with a specific seed, loss and dataset, which is limited in real-world scenarios when fake images may be generated by privately trained models. This motivates us to ask whether it is possible to attribute fake images to the source models' architectures even if they are finetuned or retrained under different configurations. In this work, we present the first study on *Deepfake Network Architecture Attribution* to attribute fake images on architecture-level. Based on an observation that GAN architecture is likely to leave globally consistent fingerprints while traces left by model weights vary in different regions, we provide a simple yet effective solution named DNA-Det for this problem. Extensive experiments on multiple cross-test setups and a large-scale dataset demonstrate the effectiveness of DNA-Det.

1 Introduction

The deepfake technology has raised big challenges to visual forensics. Dedicated research efforts are paid (Durall, Keuper, and Keuper 2020; Wang et al. 2020; Liu, Qi, and Torr 2020; Zhang, Karaman, and Chang 2019; Jeon et al. 2020; Nataraj et al. 2019; Chai et al. 2020; Frank et al. 2020; Zhao et al. 2021; Haliassos et al. 2021; Liu et al. 2021; Chandrasegaran, Tran, and Cheung 2021; Li et al. 2020) to detect generated images in recent years. However, only real/fake classification is not the end: On the one hand, for malicious and illegal content, law enforcers need to identify its owner. On the other hand, GAN models need experienced designers with laborious trial-and-error testings, some of which have high commercial value and should be protected. These motivate works on fake image attribution, i.e., attributing the origin of fake images. For fake image attribution, existing works (Marra et al. 2019; Yu, Davis, and Fritz 2019; Frank et al. 2020; Joslin and Hao 2020) perform attribution for multiple GAN models and obtain high classification accuracies. While encouraging, the problem of GAN attribution is far from studied and solved sufficiently.

From the perspective of understanding GAN fingerprints, previous works (Marra et al. 2019; Yu, Davis, and Fritz 2019; Frank et al. 2020; Joslin and Hao 2020) suggest that: 1) Models with different architectures have distinct fingerprints. 2) With architecture fixed, changing only the model's random initialization seed or training data also results in a distinct fingerprint. From 2), it can be deduced that model weights may influence GAN fingerprints. While from 1), it cannot be verified whether the GAN fingerprint is related to the architecture since weights also change as the architecture changes. This motivates us to investigate whether GAN architectures leave fingerprints. In other words, do different models with the same architecture share the same fingerprint? Answering this question may help us understand deeper into the generation of GAN fingerprints.

From the perspective of application, previous works on GAN attribution only perform model-level attribution, i.e., training and testing images come from the same model, which means for fake images, we can only handle those generated by seen models. However, this approach is limited in real-world scenarios. For malicious content supervision, the malicious producers would probably download a certain deepfake project to their own computers from code hosting platforms at first, and then use their personal collected training data to finetune or train from scratch instead of directly using the public available models. In such a situation, model-level attribution is no longer applicable since it is unfeasible to get the privately trained model. For intellectual property protection, if an attacker steals a copyrighted GAN, and modifies weights by finetuning, model-level attribution will fail too. These motivate us to solve fake image attribution under a more generic setting, i.e., attribute fake images to the source architecture instead of the specific model.

In this paper, we propose a novel task of **Deepfake Network Architecture Attribution**. Compared with model-level attribution, architecture-level attribution requires attributing fake images to their generators' architectures even if the models are fine-tuned or retrained with a different seed, loss or dataset. Although architecture-level attribution is more coarse-grained than model-level attribution, it is still

[*] Equal contribution.
[†] Corresponding author

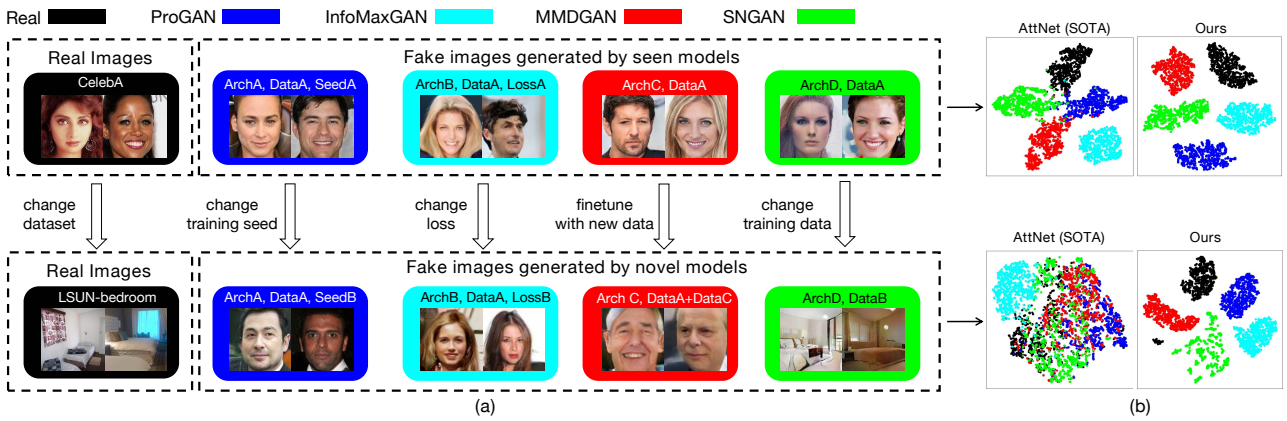


Figure 1: (a) The scenario for deepfake network architecture attribution. (b) The t-SNE visual comparison between our learned features and AttNet (Yu, Davis, and Fritz 2019). When testing on images from the same set of GAN models and real images used in training (above), AttNet and our method both extract distinct features. However, when testing on novel images from finetuned models or models with changed seed, loss or dataset (below), features extracted by AttNet are highly entangled, but our method can still extract well-separated feature.

Figure 2: Class activation maps from trained AttNet classifying {real, ProGAN, InfoMaxGAN, MMDGAN, SNGAN}.

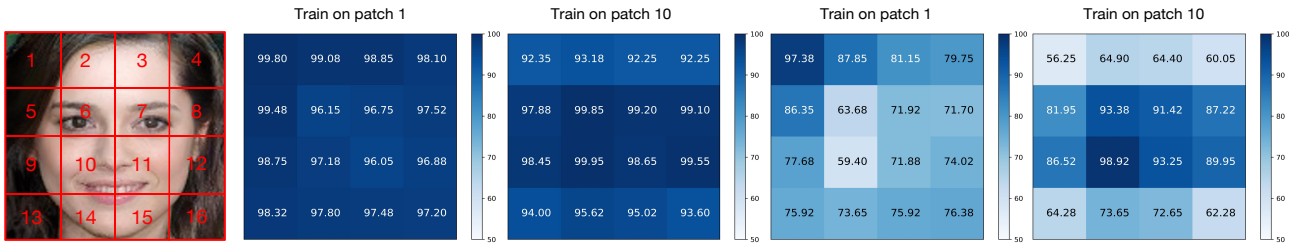
challenging. As Figure 1 shows, for seen GAN models with certain architectures (above), there may exist other versions of novel models different in training seed, loss or training data (below). If we train on real images and seen models, as the t-SNE plots show, although AttNet (Yu, Davis, and Fritz 2019) extracts distinct features when testing on images generated by seen models (above), features are highly entangled on images from novel models (below). To explain the drop, we visualize what regions the network focuses on for attribution in Figure 2. We notice that the network tends to focus on local regions closely related to image semantics such as eyes and mouth. However, for architecture attribution, it is problematic to concentrate on semantic-related local regions.

In this work, we observe that: GAN architecture is likely to leave fingerprints, which are globally consistent among the full image instead of gathered in local regions. Besides, traces left by weights varies in different regions. This observation is based on an empirical study in Section 3. Specifically, we divide GAN images into patches of equal size and conduct model weight classification and architecture classification on patches. We train on patches from a single position and then test on patches from every position respectively. We can observe that: 1) In weight classification, the testing accuracy is high on patches with the same position as patches used in training, but drops largely on patches from other positions. This result indicates that traces left by model weights are likely associated with the position. 2) In architecture classification, testing accuracies on patches from all positions are higher than 90%, even though we trained solely on patches from a single position. This suggests that there exist globally consistent traces on GAN images, which are distinct for models of different architectures. This globally consistent distinction is probably caused by the architecture under the prior observation from 1) that weight traces vary in different regions.

Motivated by the observation above, it is foreseeable that if we concentrate on globally consistent traces, architecture traces would play a primary role in decision, which generalize better when testing on unseen models. Thus we design a method for GAN architecture attribution, which we call DNA-Det: Deepfake Network Architecture Detector. DNA-Det explores globally consistent features that are invariant to semantics to represent GAN architectures by two techniques, i.e., pre-training on image transformation classification and patchwise contrastive learning. The former helps the network to focus on architecture-related traces, and the latter strengthen the global consistency of extracted features.

To summarize, the contributions of this work include:

- We propose a novel task of *Deepfake Network Architecture Attribution* to attribute fake images to the source architectures even if models generating them are finetuned or retrained with a different seed, loss function or dataset.
- We develop a simple yet effective approach named DNA-Det to extract architecture traces, which adopts pre-training on image transformation classification and patchwise contrastive learning to capture globally consistent features that are invariant to semantics.
- The evaluations on multiple cross-test setups and a large-scale dataset verify the effectiveness of DNA-Det. DNA-Det maintains a significantly higher accuracy than existing methods in cross-seed, cross-loss, cross-finetune and cross-dataset settings.



(a) Numbered patches (b) Testing accuracies for architecture classification (c) Testing accuracies for weight classification

Figure 3: Empirical study on GAN fingerprint in architecture and weight classification. We only train on patches from a fixed position, and test on patches from all positions respectively. We show the results training on patch 1 and 10.

2 Related Work

Fake image attribution can be classified into *positive attribution* (Kim, Ren, and Yang 2020; Yu et al. 2020a,b) and *passive attribution* (Marra et al. 2019; Yu, Davis, and Fritz 2019; Frank et al. 2020; Joslin and Hao 2020; Xuan et al. 2019). This paper focuses on passive attribution. Works on positive attribution insert artificial fingerprint (Yu et al. 2020a,b) or key (Kim, Ren, and Yang 2020) directly into the generative model. Then when tracing the source model, the fingerprint or key can be decoupled from generated images. Positive attribution requires "white-box" model training and thus is limited in "black-box" scenario when only generated images are available. Passive attribution aims at finding the intrinsic differences between different types of generated images without getting access to the generative model, which is more efficient and challenging compared with positive attribution. The work in (Marra et al. 2019) finds averaged noise residual can represent GAN fingerprint. The work in (Yu, Davis, and Fritz 2019) decouples GAN fingerprint into model fingerprint and image fingerprint. Specifically, this work takes the final classifier features and reconstruction residual as the image fingerprint and the corresponding classifier parameters in the last layer as the model fingerprint. The work in (Frank et al. 2020) observes the discrepant DCT frequency spectrums exhibited by images generated from different GAN architectures, and then sends the DCT frequency spectrum into classifiers for source identification. The work in (Joslin and Hao 2020) derive a similarity metric on the frequency domain for GAN attribution. Above works on passive fake image attribution all conduct experiments on multiple GAN models and achieve high accuracy. While encouraging, these works are restricted to model-level attribution (i.e., training and testing images come from the same set of models), which is limited in the real scenario. In this paper, we propose to solve fake image attribution on architecture-level, which can attribute generative models to their architectures even if they are modified by fine-tuning or retraining.

Pre-taining on image transformations was previously used in image manipulation detection (Wu, AbdAlmageed, and Natarajan 2019; Huh et al. 2018) based on the assumption that there may exist post-processing discontinuity between the tampered region and its surrounding. Our approach is inspired by these works but driven by a different motivation: Many traditional image transformation functions are similar to image generation operations, and pretraining on classifying different image transformations can help the network focus on architecture-related globally consistent traces.

3 Empirical Study on GAN Fingerprint

Reviewing fake images' generation process, the network components (e.g. convolution, upsampling, activation, normalization and so on) all operate on feature maps spatially equal, thus traces left by these components is likely to be identical on every patch. Intuitively, we hypothesize that if GAN architecture leaves traces, they would be globally consistent among patches. To verify this hypothesis, we design an empirical study as follows: We conduct two attribution experiments: 1) Architecture classification. We do four-class classification classifying images from four GAN models with different architectures, including ProGAN (Karras et al. 2017), MMDGAN (Bińkowski et al. 2018), SNGAN (Miyato et al. 2018) and InfoMaxGAN (Lee, Tran, and Cheung 2021), all of which are trained on celebA dataset (Liu et al. 2015). 2) Weight classification. Another four-class classification classifying images from four ProGAN models trained on celebA but with different training seed. It is foreseeable that in weight classification, the network will depend on weight traces for classification. In architecture classification, the network may depend on architecture or weight traces, or both. Implementation details are provided in the Appendix.

In detail, as Figure 3(a) shows, we divide each image into 4×4 grid of patches and number them from 1 to 16 according to the position. We only train on patches from a fixed position and test on patches from all positions respectively, getting 16 testing accuracies for 16 positions. Fig. 3(b)(c) shows the testing accuracies in architecture and weight classification when train solely on patches from position 1 and position 10. From the experiment results, we have two observations: 1) Weight traces is likely associated with the position, a semantic association perhaps. Since in weight classification, when the network is trained on patches from a single position, the testing accuracy is high on this position, but drops a lot on patches from other positions. 2) Architecture is likely to leave fingerprints, which are globally consistent

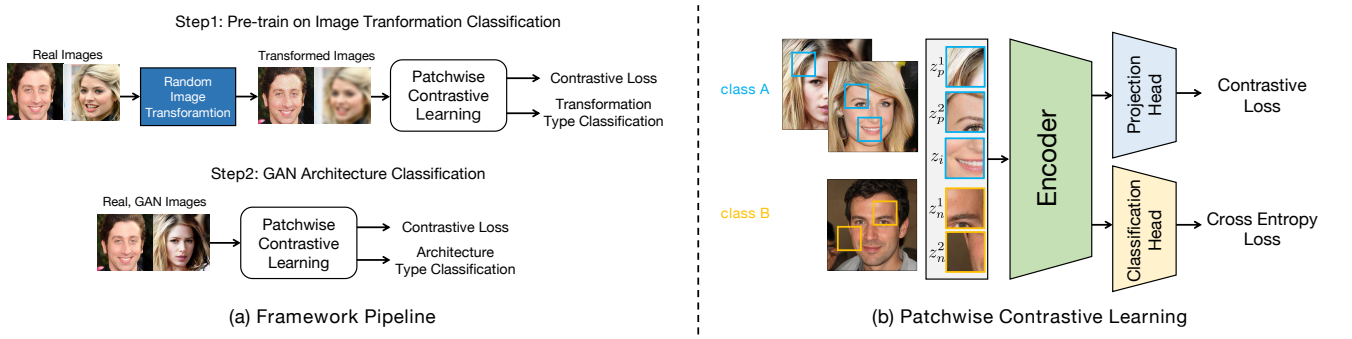


Figure 4: **Overview of DNA-Det's learning pipeline.** (a) Framework pipeline. In the first step, we use image-transformation classification as a self supervision task to make the network focus on architecture-related traces. In the second step, we use the weights learned in the former step as the initial weight and conduct GAN architecture classification. In the two steps, the network is trained by a patchwise contrastive learning mechanism. (b) Patchwise contrastive learning used in the two steps in (a), which force patches of the same class (image transformation type or GAN architecture type) close-by, and patches of different classes far apart.

among the full image. In architecture classification, testing accuracies on all positions are higher than 90% even though only patches from one single position are used for training. Given the prior observation from 1) that weight traces varies in different regions, this globally consistent distinction is probably caused by the architecture.

Our goal is to get a stable architecture representation regardless of weights. Given the empirical observations above, an intuitive approach is to restrict the global consistency of extracted features, such that architecture traces would play a decisive role in conducting architecture attribution.

4 Proposed Approach

Problem definition. We set deepfake network architecture attribution as a multi-class classification problem. Given an image x^y with source $y \in \mathbb{Y} = \{real, G_1, G_2, \ldots, G_N\}$, where $G_1, \ldots, G_N$ are different architectures. Our goal is to learn a mapping $D(x^y) \rightarrow y$. Note that the architecture in this paper refers to the architecture of the generator. Loss functions and discriminator are not considered as part of the architecture , since they only influence the generator's weights indirectly by gradient back propagation, while our goal is to attribute fake images to the source architecture regardless of model weights.

Framework overview. Figure 4 overviews the learning pipeline for DNA-Det. We train our network by two steps. In the first step, we use image transformation classification as a pre-train task to make the network focus on architecture-related traces. In the second step, we use the weights learned in former step as the initial weight and conduct GAN architecture classification. In the two steps, we use patchwise contrastive learning to force patches of the same class (image transformation type or GAN architecture type) close-by, and patches of different classes would be pushed far apart.

4.1 Pre-train on Image Transformations

Given a certain number of GAN images with architecture labels, the obvious idea is to use these labels to train a classifier using a supervised objective such as cross-entropy loss. However, directly using features learned by supervised training is problematic. This would make the classifier harvest any useful features to help classification, which may include semantic-related information as shown in Figure 2. Inspired by works in (Huh et al. 2018; Wu, AbdAlmageed, and Natarajan 2019), we use image transformation classification as a pre-train task motivated two reasons: 1) We found that some traditional image transformation operations are similar to the generator's components. For example, blurring and noising with kernels resembles convolution computation, and the resampling operation is similar to the upsampling layer. Thus traces left by traditional image transformations share similar properties with architecture traces. 2) Traditional image transformations and the generator's components both conduct on the images spatially equal. Thus pre-training on image transformation classification could aid the network to focus on globally consistent traces.

In detail, four image transformation families are considered: compression, blurring, resampling and noising. We randomly choose the parameters for each operation from a discrete set of numbers. Each operation with a specific parameter is taken as a unique type and we finally get 170 types of image transformations. In training, we apply these transformations on a natural image dataset containing LSUN-bedroom (Yu et al. 2015) and CelebA. Then we conduct patchwise contrastive learning (described in Section 4.2) to force patches with the same image transformation close-by and different image transformations far apart. We use this pre-trained model to initialize model weights.

4.2 Patchwise Contrastive Learning

We adopt a contrastive learning mechanism on patches to strengthen the global consistency of extracted features. Details are shown in Figure 4(b). Instead of training on whole images, randomly cropped patches are used as input samples. These patches are fed into an encoder followed by a projection head and a classification head. The projection

head consists of a two-layer MLP network, which maps representations to the space where a supervised contrastive loss (Khosla et al. 2020) is calculated. For an anchor patch, patches with the same class are positives, and patches with different classes are negatives. The contrastive loss forces patches from the same class closer in the representation space, and pushes patches from different classes farther away. Specifically, the contrastive loss is calculated as follows:

$$L_{con} = \sum_{i \in I} \frac{-1}{P(i)} \sum_{p \in P(i)} \log \frac{\exp\left(z_i \cdot z_p / \tau\right)}{\sum_{a \in A(i)} \exp\left(z_i \cdot z_a / \tau\right)} \quad (1)$$

Here, $i \in I$ is the index of an arbitrary training patch. $P(i)$ is the set of all positive pairs for the patch i. $A(i) \equiv I \backslash \{i\}$, which includes all positive and negative pairs for patch i. z_i is the feature vector for patch i. z_a is the feature vector for patches in $A(i)$. z_p and z_n (shown in Figure 4(b)) refer to the feature vector for positive and negative pairs respectively.

The classification head maps the representation from the encoder to the label space, in which we calculate a cross-entropy loss L_{ce}. Overall, the objective for patchwise contrastive learning is formulated as:

$$L = w_1 \cdot L_{con} + w_2 \cdot L_{ce} \quad (2)$$

where ω_1 and ω_2 are non-negative weights. Automatic weighted learning mechanism (Kendall, Gal, and Cipolla 2018) is used to adaptively optimize the objective.

5 Experiments

5.1 Experimental Setup

Compared Methods. We compare our method with several representative methods for fake image attribution as follows: 1) PRNU (Marra et al. 2019): a method using photo-response non-uniformity (PRNU) patterns as the fingerprint for fake image attribution. 2) AttNet (Yu, Davis, and Fritz 2019): a PatchGAN-like classifier for fake image attribution. 3) LeveFreq (Frank et al. 2020): a frequency-based method that uses Discrete Cosine Transform (DCT) images for fake image attribution and detection.

Implementation Details. For the network architecture, we use a shallow 8-layer CNN network as the encoder. The output channel numbers for convolution layers are 64,64,128,128,256,256,512 and 512. A Global Average Pooling is added after the convolution layers. For patchwise contrastive learning, we firstly resize all images to 128px (the lowest resolution in the dataset), and then resize them to 512px to magnify GAN traces, on which we randomly crop 16 patches of 64px as inputs. For inference, we test on the full image instead of patches. For optimization, we choose Adam optimizer. For the celebA experiment in section 5.2, the initial learning rate is set to 10^{-4} and is multiplied by 0.9 for every 500 iterations. For the LSUN-bedroom experiment in section 5.2 and the experiment in section 5.3, the initial learning rate is set to 10^{-3} and is multiplied by 0.9 for every 2500 iterations. The batch size is $32 \times$ #classes in Section 5.2 and $16 \times$ #classes in Section 5.3 with a class balance strategy. For the GradCAM maps shown in this paper, we visualize on layer-4. More details of the experiments could be found in the appendix material.

Real, GAN	train-set	cross-seed	cross-loss	cross-finetune	cross-dataset
celebA					
Real	celebA	-	-	-	bedroom
ProGAN	celebA(seed0)	seed 1-9	-	FFHQ-elders	bedroom
MMDGAN	celebA	-	CramerGAN	FFHQ-elders	bedroom
SNGAN	celebA	-	-	FFHQ-elders	bedroom
InfoMaxGAN	celebA	-	SSGAN	FFHQ-elders	bedroom
LSUN-bedroom					
Real	bedroom	-	-	-	celebA
ProGAN	bedroom(seed0)	seed 1-9	-	LSUN-sofa	celebA
MMDGAN	bedroom	-	CramerGAN	LSUN-sofa	celebA
SNGAN	bedroom	-	-	LSUN-sofa	celebA
InfoMaxGAN	bedroom	-	SSGAN	LSUN-sofa	celebA

Table 1: Dataset split for cross seed, loss, finetune and dataset evaluation. The evalution consists of two groups: celebaA and LSUN-bedroom.

5.2 Evaluation on Multiple Cross-Test Setups

Datasets. This experiment is conducted on 5 classes: real, ProGAN, MMDGAN, SNGAN, InfoMaxGAN. Details of the dataset split are shown in Table 1. As the table shows, the experiment is composed of two groups named by ***celebA*** and ***LSUN-bedroom***, depending on the training dataset of the GAN models and real images in the train-set. For each group, we conduct cross-seed, cross-loss, cross-finetune, and cross-dataset testings to evaluate the generalization of architecture attribution on unseen models with different random seed, loss function and dataset from the models in the train-set. Specifically, for cross-seed testing, the ProGAN model in the train-set is trained with seed 0, but we test on ProGAN models with seed 1-9. For cross-loss testing, we test on CramerGAN (Bellemare et al. 2017) and SSGAN (Chen et al. 2019) models, which have the same generator architecture as MMDGAN and InfoMaxGAN respectively but with different loss functions. Note that in cross-seed and cross-loss testing, models are trained on the same dataset as models in the train-set to control the dataset variable. For cross-finetune testing, we test on models finetuned on the models in the train-set. We finetune with FFHQ-elders and LSUN-sofa respectively in the celebA and LSUN-bedroom experiment. For cross-dataset testing, we test on models trained on different datasets, e.g., in the celebA experiment, the models in the train-set are all trained on celebA, but we test on models all trained on LSUN-bedroom. All of the GAN images and real images in this dataset are 128px.

Results. The results are shown in Table 2, which are measured by accuracy. Compare DNA-Det (the last row) with existing methods (first three rows), we have several findings: 1) In closed-set testing, nearly all methods achieve relatively good performance, suggesting that features captured by these methods are sufficient for model-level attribution. 2) In cross-testings, the performance degrades across all methods with different degrees. Among these cross-testings, the performance drops the most in cross-finetune and cross-dataset testing, showing that attribution methods are likely to learn content-relevant features, which is harmful for architecture attribution. 3) Compared with existing methods, DNA-Det achieves superior performance in closed-set and all cross-testings, especially gaining large improvement in cross-finetune and cross-dataset testing (from $\sim 30\%$ accu-

Method	*celebA*					*LSUN-bedroom*				
	closed-set	cross-seed	cross-loss	cross-finetune	cross-dataset	closed-set	cross-seed	cross-loss	cross-finetune	cross-dataset
PRNU(MIPR2019)	90.64	20.69	29.33	48.88	21.27	66.31	30.50	35.23	53.58	24.13
LeveFreq(ICML20)	99.50	86.02	92.50	52.19	47.07	99.53	82.76	76.30	74.59	53.42
AttNet(ICCV19)	98.88	83.50	87.10	35.21	38.54	99.25	88.73	89.28	35.21	21.88
AttNet+PT	99.50	93.46	82.65	36.89	44.71	98.77	98.19	97.60	73.29	49.82
AttNet+PCL	**100.00**	99.72	99.03	53.38	90.51	**100.00**	**100.00**	**100.00**	95.69	81.06
AttNet+PT+PCL	**100.00**	99.81	**99.80**	57.48	93.76	**100.00**	98.69	99.95	93.81	79.47
Base	99.88	94.17	62.93	45.83	33.02	**100.00**	72.73	74.83	38.45	21.66
Base+PT	**100.00**	99.36	95.83	54.80	53.00	**100.00**	98.74	97.78	61.74	49.73
Base+PCL	**100.00**	99.96	98.30	77.89	89.93	**100.00**	**100.00**	99.25	94.19	81.09
Base+PT+PCL(DNA-Det)	**100.00**	**99.99**	99.53	**97.65**	**94.95**	**100.00**	99.99	99.90	**97.50**	**83.45**

Table 2: Evaluation on multiple cross-test setups for Section 5.2 and ablation study for Section 5.4 measured by accuracy. "PT" means pre-train on image transformations. "PCL" means patchwise contrastive learning.

GAN	Block Type	Skip Connect	Upsample	Norm
ProGAN	DCGAN	-	Nearest	PN
MMDGAN	ResNet	Upsample+Conv	Depth2Space	BN
SNGAN	ResNet	Upsample+Conv	Nearest	BN
InfoMaxGAN	ResNet	Upsample+Conv	Bilinear	BN

Table 3: Structure components of four GANs.

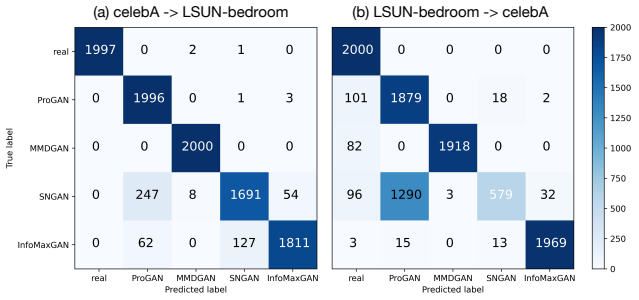


Figure 5: Confusion matrices of two cross-dataset testings.

racy to over 80%). As a result, DNA-Det is qualified for deepfake network architecture attribution.

Further Analysis. We show the confusion matrices on the two cross-dataset testings in Figure 5. From the confusion matrices, we find that SNGAN and InfoMaxGAN, SNGAN and ProGAN tend to be confused. To explore the reason, we check the details of the architecture components in Table 3. We notice that ProGAN and SNGAN both use Nearest upsampling layer but with different block structures (block type and skip connection). SNGAN and InfoMaxGAN share the same block structure but use different upsampling layers. From the relationship between the confusion matrices and architecture components, we have the following findings: 1) The successfully classified samples on the diagonal reflect that different block structures and upsampling types leave distinct traces, such that ProGAN and SNGAN, SNGAN and InfoMaxGAN can be distinguished in cross-dataset testing. 2) The misclassified samples suggest that the network doesn't capture the overall architecture traces on several samples, which causes the confusion between architectures whose components are partly the same.

5.3 Evaluation on GANs in the Wild

Datasets. In the real-world scenario, the collected data for different architectures may be more complex. The models may generate diverse contents and don not overlap among architectures. The content bias will mislead the network to focus on useless semantics. Thus we simulate the challenging real-world scenario and construct a large-scale dataset containing multiple public-released GANs with diverse contents as shown in Table 6. The dataset includes 58 GAN models from 10 architectures with 3 resolutions. Apart from the GANs used in Section 5.2, we further include CycleGAN (Zhu et al. 2017), StackGAN2 (Zhang et al. 2019), StyleGAN (Karras, Laine, and Aila 2019) and StyleGAN2 (Karras et al. 2020). Note that we take the different resolution versions of the same algorithm as different architectures, because they are different in the number of layers. The performance is measured by accuracy and macro-averaged F1-score over all classes.

Results. From the results in Table 4, we can observe that: 1) With more GANs added, the experiment becomes more difficult as the accuracies of compared methods are all below 90% in closed-set; 2) Our method outperforms other methods, not only in closed-set but also in cross-dataset testing, showing the effectiveness of our method in distinguishing different architectures and the generalization ability in real-world fake image architecture attribution.

5.4 Ablation Study

Quantitative Analysis. The results in Table 2 and Table 4 validate the effectiveness of pre-train on image transformations (PT) and patchwise contrastive learning (PCL). Removing any of them on DNA-Det causes the performance to drop in nearly all settings. PCL is by far the most important one. In the hardest cross-dataset evaluation, removing it results in a dramatic drop of 41.95, 33.72 and 32.65 points. This shows that the global consistency assumption makes sense and plays an important role in our method. Without PT, the performance drops by a modest 5.02, 2.36 and 3.28 points, respectively. But when PT is added to the base network, it improves 19.98, 28.07 and 26.09 points, which

Method	closed-set		cross-dataset	
	Acc.	F1	Acc.	F1
PRNU(MIPR2019)	66.77	63.76	20.31	12.58
LeveFreq(ICML2020)	70.53	73.71	38.96	22.73
AttNet(ICCV2019)	84.57	86.48	53.21	33.14
Base	88.11	90.27	47.95	25.04
Base+PT	95.79	97.09	73.02	50.82
Base+PCL	**99.99**	**99.99**	92.60	80.54
Base+PT+PCL(DNA-Det)	99.96	99.98	**92.94**	**83.54**

Table 4: Evalution on GANs in the wild for Section 5.3

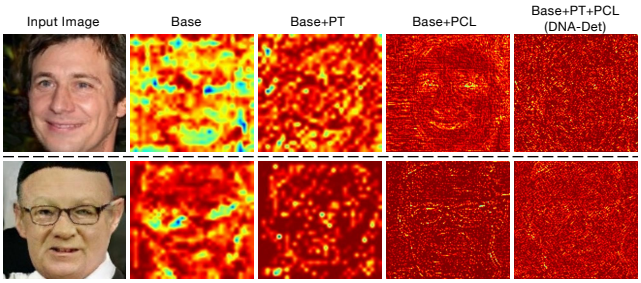


Figure 6: Qualitative Analysis. Comparison of GradCAM heatmaps. Bluer color indicates a higher response for better visualization.

means features extracted by image transformation classification is related to architecture traces in some aspects. We also apply PCL and PT to the compared method AttNet, both results in a large improvement. The former improves 51.97 and 58.18 points, and the latter improves 6.17 and 27.94 in cross-dataset evaluation as shown in Table 2.

Qualitative Analysis. We show in Figure 6 the GradCAM (Selvaraju et al. 2017) heatmaps to visualize how focused regions change with PCL and PT added. The two input images are from ProGAN and SNGAN respectively. The base network tends to concentrate on semantic-related local regions such as eyes and facial outline, which is untransferable for architecture attribution. With the PT added, the areas of concern are no longer locally focused. Adding PCL to the base network makes the feature extractor rely on more global and fine-grained traces, yet some salient regions such as eyes and face boundary can still be observed. PT plus PCL promote the network to only focus on globally consistent traces all around the image, and semantic-related regions nearly disappear in the heatmap.

5.5 Robustness Analysis

We consider five types of attacks that perturb test images: noise, blur, cropping, JPEG compression, relighting and random combination of them. Detailed parameters of these attacks are the same with the work in (Yu, Davis, and Fritz 2019). Table 5 reports the closed-set and cross-dataset testing accuracy in the ***celebA*** experiment under these attacks, which are included as a data augmentation in training for all methods. From the results, in closed-set testing, our method

Method	Crop	Blur	JPEG	Noise	Relight	Combination
Closed-Set						
PRNU	49.55	71.00	64.57	72.77	72.09	40.71
LeveFreq	85.96	77.58	71.00	85.07	66.41	45.37
AttNet	89.99	89.76	85.26	90.65	80.39	73.40
DNA-Det	**100.00**	**100.00**	**97.68**	**100.00**	**96.39**	**80.16**
Cross-Dataset						
PRNU	20.29	20.1	20.58	19.55	19.09	20.17
LeveFreq	32.18	30.67	29.14	31.70	30.32	23.83
AttNet	24.01	22.65	23.89	24.67	23.70	22.65
DNA-Det	**82.48**	**82.69**	**76.43**	**81.72**	**78.90**	**59.53**

Table 5: Robustness analysis against common attacks.

Resolution	Real, GAN	Train Content	Test Content
128	Real	celebA	bedroom
	ProGAN	celebA	bedroom
	MMDGAN	celebA	bedroom
	SNGAN	celebA	bedroom
	InfoMaxGAN	celebA	bedroom
256	Real	cat, airplane	boat, horse, sofa, cow, dog, train, bicycle, bottle, diningtable, motorbike, sheep, tvmonitor, bird, bus, chair, person, pottedplant, car
	ProGAN	cat, airplane	boat, horse, sofa, cow, dog, train, bicycle, bottle, diningtable, motorbike, sheep, tvmonitor, bird, bus, chair, person, pottedplant, car
	StackGAN2	cat, church	bird, bedroom, dog
	CycleGAN	winter, orange	apple, horse, summer, zebra
	StyleGAN2	cat, church	horse
1024	Real	FFHQ	celebA-HQ
	StyleGAN	FFHQ	celebA-HQ, Yellow, Model, Asian Star, kid, elder, adult, glass, male, female, smile
	StyleGAN2	FFHQ	Yellow, Wanghong, Asian Star, kid

Table 6: The dataset for Section 5.3. For cross-dataset testing, the split makes sure training and testing models of each architecture don't overlap in content.

overcomes all attacks when any single attack is applied, and over other methods. The performance drops the most on combination attacks due to its complexity, but we can still get an acceptable 80% accuracy. In cross-dataset testing, our method can get almost 80% accuracy under any of these attacks and a 59.53% accuracy under combination attack, much superior to compared methods.

6 Conclusions

In this work, we present the first study on deepfake network architecture attribution. Our empirical study verifies the existence of GAN architecture fingerprints, which are globally consistent on GAN images. Based on the study, we develop a simple yet effective approach named by DNA-Det to capture architecture traces by adopting pre-training on image transformations and patchwise contrastive learning. We evaluate DNA-Det on multiple cross-test setups and a large-scale dataset including 59 models derived from 10 architectures, verifying DNA-Det's effectiveness.

7 Acknowledgements

The corresponding author is Juan Cao. The authors thank Qiang Sheng, Xiaoyue Mi, Yongchun Zhu and anonymous reviewers for their insightful comments. This work was supported by the Project of Chinese Academy of Sciences (E141020), the Project of Institute of Computing Technology, Chinese Academy of Sciences (E161020), Zhejiang Provincial Key Research and Development Program of China (No. 2021C01164), and the National Natural Science Foundation of China (No. 62172420).

References

Bellemare, M. G.; Danihelka, I.; Dabney, W.; Mohamed, S.; Lakshminarayanan, B.; Hoyer, S.; and Munos, R. 2017. The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*.

Bińkowski, M.; Sutherland, D. J.; Arbel, M.; and Gretton, A. 2018. Demystifying MMD GANs. In *ICLR*.

Chai, L.; Bau, D.; Lim, S.-N.; and Isola, P. 2020. What makes fake images detectable? Understanding properties that generalize. In *ECCV*, 103–120. Springer.

Chandrasegaran, K.; Tran, N.-T.; and Cheung, N.-M. 2021. A Closer Look at Fourier Spectrum Discrepancies for CNN-generated Images Detection. In *CVPR*, 7200–7209.

Chen, T.; Zhai, X.; Ritter, M.; Lucic, M.; and Houlsby, N. 2019. Self-supervised gans via auxiliary rotation loss. In *CVPR*, 12154–12163.

Durall, R.; Keuper, M.; and Keuper, J. 2020. Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. In *CVPR*, 7890–7899.

Frank, J.; Eisenhofer, T.; Schönherr, L.; Fischer, A.; Kolossa, D.; and Holz, T. 2020. Leveraging frequency analysis for deep fake image recognition. In *ICML*, 3247–3258. PMLR.

Haliassos, A.; Vougioukas, K.; Petridis, S.; and Pantic, M. 2021. Lips Don't Lie: A Generalisable and Robust Approach To Face Forgery Detection. In *CVPR*, 5039–5049.

Huh, M.; Liu, A.; Owens, A.; and Efros, A. A. 2018. Fighting fake news: Image splice detection via learned self-consistency. In *ECCV*, 101–117.

Jeon, H.; Bang, Y. O.; Kim, J.; and Woo, S. 2020. T-GD: Transferable GAN-generated Images Detection Framework. In *ICML*, 4746–4761. PMLR.

Joslin, M.; and Hao, S. 2020. Attributing and Detecting Fake Images Generated by Known GANs. In *2020 IEEE Security and Privacy Workshops (SPW)*, 8–14. IEEE.

Karras, T.; Aila, T.; Laine, S.; and Lehtinen, J. 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Karras, T.; Laine, S.; and Aila, T. 2019. A style-based generator architecture for generative adversarial networks. In *CVPR*, 4401–4410.

Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2020. Analyzing and improving the image quality of stylegan. In *CVPR*, 8110–8119.

Kendall, A.; Gal, Y.; and Cipolla, R. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 7482–7491.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

Kim, C.; Ren, Y.; and Yang, Y. 2020. Decentralized Attribution of Generative Models. *arXiv preprint arXiv:2010.13974*.

Lee, K. S.; Tran, N.-T.; and Cheung, N.-M. 2021. InfoMax-GAN: Improved adversarial image generation via information maximization and contrastive learning. In *WACV*, 3942–3952.

Li, H.; Li, B.; Tan, S.; and Huang, J. 2020. Identification of deep network generated images using disparities in color components. *Signal Processing*, 174: 107616.

Liu, H.; Li, X.; Zhou, W.; Chen, Y.; He, Y.; Xue, H.; Zhang, W.; and Yu, N. 2021. Spatial-phase shallow learning: rethinking face forgery detection in frequency domain. In *CVPR*, 772–781.

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep learning face attributes in the wild. In *ICCV*, 3730–3738.

Liu, Z.; Qi, X.; and Torr, P. H. 2020. Global texture enhancement for fake face detection in the wild. In *CVPR*, 8060–8069.

Marra, F.; Gragnaniello, D.; Verdoliva, L.; and Poggi, G. 2019. Do gans leave artificial fingerprints? In *MIPR*, 506–511. IEEE.

Miyato, T.; Kataoka, T.; Koyama, M.; and Yoshida, Y. 2018. Spectral Normalization for Generative Adversarial Networks. In *ICLR*.

Nataraj, L.; Mohammed, T. M.; Manjunath, B.; Chandrasekaran, S.; Flenner, A.; Bappy, J. H.; and Roy-Chowdhury, A. K. 2019. Detecting GAN generated fake images using co-occurrence matrices. *Electronic Imaging*, 2019(5): 532–1.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 618–626.

Wang, S.-Y.; Wang, O.; Zhang, R.; Owens, A.; and Efros, A. A. 2020. CNN-generated images are surprisingly easy to spot... for now. In *CVPR*, 8695–8704.

Wu, Y.; AbdAlmageed, W.; and Natarajan, P. 2019. Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In *CVPR*, 9543–9552.

Xuan, X.; Peng, B.; Wang, W.; and Dong, J. 2019. Scalable fine-grained generated image classification based on deep metric learning. *arXiv preprint arXiv:1912.11082*.

Yu, F.; Seff, A.; Zhang, Y.; Song, S.; Funkhouser, T.; and Xiao, J. 2015. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

Yu, N.; Davis, L. S.; and Fritz, M. 2019. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *ICCV*, 7556–7566.

Yu, N.; Skripniuk, V.; Abdelnabi, S.; and Fritz, M. 2020a. Artificial GAN Fingerprints: Rooting Deepfake Attribution in Training Data. *arXiv e-prints*, arXiv–2007.

Yu, N.; Skripniuk, V.; Chen, D.; Davis, L.; and Fritz, M. 2020b. Responsible Disclosure of Generative Models Using Scalable Fingerprinting. *arXiv preprint arXiv:2012.08726*.

Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Wang, X.; Huang, X.; and Metaxas, D. N. 2019. StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks. *TPAMI*, 41(8): 1947–1962.

Zhang, X.; Karaman, S.; and Chang, S.-F. 2019. Detecting and simulating artifacts in gan fake images. In *WIFS*, 1–6. IEEE.

Zhao, H.; Zhou, W.; Chen, D.; Wei, T.; Zhang, W.; and Yu, N. 2021. Multi-attentional deepfake detection. In *CVPR*, 2185–2194.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2223–2232.