All Roads Lead to CLIP: Mapping the Landscape of CLIP-Based Image Forensics

Anonymous authors
Paper under double-blind review

Abstract

The increasing adoption of generative models for image generation, including Generative Adversarial Networks and Diffusion Models, has led to the dissemination of synthetic content online. These techniques enable even non-expert users to create highly realistic content, raising concerns regarding authenticity and potential misuse. In response, the multimedia forensics community has developed numerous techniques to determine whether an image is synthetically generated and to identify which specific generative model produced it.

Recent studies have shown that features extracted by large pre-trained models, most notably CLIP, provide a highly effective representation for deepfake detection when combined with simple classifiers. While several different CLIP-based approaches have been proposed, the majority of them adhere to a largely standard pipeline and differ primarily in implementation choices, such as the selection of the CLIP backbone, classifier architecture, image preprocessing strategy, and evaluation protocol. The impact of these choices is frequently overlooked, and the rationale behind them is rarely justified.

In this paper, we systematically analyze CLIP-based synthetic image detectors, studying how design choices affect detection and attribution performance. Moreover, we investigate the scalability of the considered CLIP backbones across different training set sizes. We identify the best configuration, referred to as DejaVu, and evaluate its generalization capabilities and robustness to common post-processing operations. Our results show that larger and more expressive backbones produce more discriminative representations, particularly when paired with deeper MLP classifiers, achieving strong performance even with limited training data. Finally, we show that DejaVu exhibits robustness to common real-world post-processing, especially for diffusion-based images, and that generative models cluster meaningfully, largely due to shared architectural characteristics, enabling near-optimal performance with fewer training generators.

1 Introduction

The detection and attribution of fake images have become an increasingly important task, as modern generative models, such as Generative Adversarial Networks (GANs) and, more recently, Diffusion Models (DMs), have significantly facilitated the creation and dissemination of highly realistic synthetic images. The widespread accessibility of these tools, even to non-expert users, raises significant concerns about image authenticity and the potential misuse of generated content, thereby making detection and attribution methods a central challenge in image forensics. Moreover, as synthetic content continue to spread across online platforms, often undergoing multiple post-processing operations, there is a growing need for image forensic methods capable of operating reliably under realistic, real-world conditions.

Extensive research efforts have focused on discriminating real images from synthetic content and, in some cases, on attributing images to the generative model responsible for their creation, by exploiting characteristic artifacts or artificial fingerprints left by generative models Yu et al. (2019); Marra et al. (2019); Corvi et al. (2023a). Although these methods usually perform well when training and test images are produced by the same generative models, their effectiveness typically declines in more realistic scenarios involving unseen

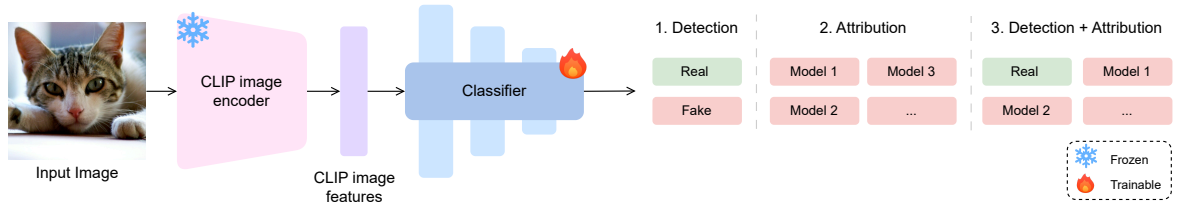

Figure 1: CLIP-based synthetic image detection common pipeline. The common paradigm consists of extracting image features from the frozen CLIP image encoder and then training a classifier to perform detection or attribution tasks.

generative models, different model families, or post-processing operations Wang et al. (2020); Corvi et al. (2023b).

In this context, large pre-trained vision–language models have recently been explored as effective tools for addressing detection and attribution challenges under realistic conditions. In particular, CLIP Radford et al. (2021) has been shown to produce highly transferable image representations that capture semantic information beyond low-level generation artifacts. As a result, several works Ojha et al. (2023); Sha et al. (2023); Cozzolino et al. (2024); Cioni et al. (2024); Amoroso et al. (2024); Cocchi et al. (2023); Koutlis & Papadopoulos (2024) have adopted CLIP as a feature extractor for synthetic image detection and attribution, obtaining promising results. However, most of existing CLIP-based approaches largely follow the same underlying strategy, as depicted in Figure 1, and differ mainly in implementation details, such as the choice of CLIP backbone, the classifier architecture, the image preprocessing procedure, and the evaluation protocol. To the best of our knowledge, the impact of these design choices on detection and attribution performance has not yet been systematically investigated. Consequently, it remains unclear which components are truly responsible for the observed gains and how CLIP-based detectors should be configured to better operate in challenging real-world scenarios.

In this paper, we conduct a systematic analysis of CLIP-based fake image detectors, investigating how different design choices affect detection and attribution performance. Specifically, we examine multiple CLIP backbones and classifier architectures, as well as different image preprocessing strategies. We further analyze how varying training set sizes affect performance across all evaluated backbones. We identify an effective configuration, referred to as **DejaVu**, and evaluate its generalization capabilities, as well as its robustness to common post-processing operations including JPEG compression, Gaussian blurring, resizing, and cropping. Through extensive experiments, our study offers practical insights into the strengths and limitations of CLIP-based forensic detectors and provides guidelines for their effective deployment.

In summary, the key contributions of this work are as follows:

- We analyze the impact of key design choices, as backbone selection, classifier architecture, and image preprocessing, in the common CLIP-based pipeline.
- We evaluate CLIP backbones across different training set sizes, highlighting their scalability and behavior in low-data regimes.
- We identify the best configuration, referred to as DejaVu, and compare its performance against existing state-of-the-art approaches.
- We evaluate DejaVu generalization capabilities and robustness to common real-world post-processing operations, such as JPEG compression, blurring, resizing, and cropping.

2 Related Work

Distinguishing real images from synthetic ones and identifying their source has become a critical challenge for image forensics. Accordingly, there is an extensive literature addressing fake image detection and source

attribution problems. Early works mainly focused on images generated by Generative Adversarial Networks, while more recent studies focused on Diffusion Models.

Regarding the detection problem, initial approaches Farid (2022a;b); Matern et al. (2019) relied on identifying visible artifacts (asymmetries, inconsistencies in perspectives, anomalous shadows, etc.) to recognize an image as fake. However, as image generation techniques have rapidly evolved, such artifacts have become less frequent and less reliable, motivating a shift towards methods that exploit more subtle forensic traces that are not usually perceptible to the human eye. In this direction, some approaches have been implemented, leveraging reconstruction errors Wang et al. (2023); Ma et al. (2023); Sha et al. (2024), diffusion noise Zhang & Xu (2023); Cazenavette et al. (2024), or features extracted from a CNN Wang et al. (2020); Guarnera et al. (2024); Coccomini et al. (2024).

Regarding the source identification problem, forensic attribution methods aim to identify the source generator by exploiting characteristic traces left in the generated images, often in the form of model-specific fingerprints observable in the frequency or residual domain Corvi et al. (2023a); Frank et al. (2020); Marra et al. (2019). These approaches have shown promising results for GAN-generated images; however, they do not readily extend to diffusion-based models because of the different generation processes and different traces left in the produced images Corvi et al. (2023b). To address this limitation, several works have been proposed, either leveraging reconstruction errors Bindini et al. (2024), or operating in a white-box setting that assumes access to the generator's architecture and weights Albright et al. (2019); Hirofumi et al. (2022); Ricker et al. (2024); such an assumption, however, rarely holds in practical scenarios.

Recently, large pre-trained models, such as CLIP Radford et al. (2021), have emerged as a powerful tool for fully synthetic image detection and attribution. C2P-CLIP Tan et al. (2025) injects manually defined category-level semantic concepts into the CLIP visual representation via a category-common prompt. RINE Koutlis & Papadopoulos (2024) exploits intermediate CLIP features, projecting them through a lightweight network into a learnable forgery-aware embedding space. GenDet Zhu et al. (2023a) employs a teacher-student discrepancy-aware learning framework using CLIP as feature extractor. Finally, FatFormer Liu et al. (2024) constructs a forgery-aware adaptive transformer that aligns image and text prompt representations, extracted by CLIP encoders, through contrastive objectives.

Additionally, several CLIP-based approaches have been proposed, all adhering to a common underlying paradigm. As depicted in Figure 1, the standard pipeline consists of extracting visual embeddings from the frozen CLIP image encoder and training a classifier to either discriminate between real and synthetic images or identify the generative model responsible for a given image. Several methods have adopted this framework, including De-Fake Sha et al. (2023), UnivFD Ojha et al. (2023), UnivAttr Cioni et al. (2024), as well as CLIPDet Cozzolino et al. (2024), the method proposed by Amoroso et al. Amoroso et al. (2024) and the one by Cocchi et al. Cocchi et al. (2023)[1]. Although these methods share a common high-level pipeline, they differ in several key design choices, as summarized in Table 1. In particular, they vary in the CLIP backbone employed, in the classifier architecture adopted for downstream tasks, and in the image processing procedure. Regarding the visual backbone, some approaches rely on the ViT-L/14 variant Ojha et al. (2023); Cozzolino et al. (2024), others employ ViT-B/32 Sha et al. (2023); Amoroso et al. (2024), while ViT-B/16 is used in Cioni et al. (2024); Cocchi et al. (2023). Similarly, different classifier models have been explored, including Multi-layer perceptrons (MLPs) Sha et al. (2023); Ojha et al. (2023), Support Vector Machines (SVM) Cozzolino et al. (2024), Logistic Regression (LR) Cioni et al. (2024); Cocchi et al. (2023); Amoroso et al. (2024), k-Nearest Neighbors (kNN) Cioni et al. (2024); Ojha et al. (2023); Cocchi et al. (2023), for both detection and attribution tasks. Finally, they also differ in the image preprocessing strategy, as some methods do not adopt the official CLIP preprocessing strategy, but instead employ a customized variant. These architectural choices can significantly affect both performance and generalization, motivating a systematic comparison of CLIP backbones and classifier designs in our experimental evaluation.

[1]Training code for CLIPDet, Amoroso et al., and Cocchi et al. is not publicly available.

Table 1: Key design choices in state-of-the-art CLIP-based approaches for synthetic image detection and attribution. Preprocessing is labeled 'Unknown' in case of lack of information in the original paper and unavailability of the training code.

Method	Preprocessing	Backbone	Classifier	Task	Evaluation Metrics
De-Fake (Sha et al., 2023)	Custom	ViT-B/32	3-MLP	Det., Attr.	Acc., F1, AUC
UnivFD (Ojha et al., 2023)	Custom	ViT-L/14	1-MLP	Det.	Acc., AP
CLIPDet (Cozzolino et al., 2024)	Unknown	ViT-L/14	SVM	Det.	Acc., AP, AUC
UnivAttr (Cioni et al., 2024)	CLIP official	ViT-B/16	LR, kNN	Det.+Attr.	Acc., AUC, OSCR
Amoroso et al. (Amoroso et al., 2024)	Unknown	ViT-B/32, RN50	LR	Det.	Acc., AUC
Cocchi et al. (Cocchi et al., 2023)	Unknown	ViT-B/16	LR, kNN	Det.	Acc.
DejaVu (ours)	CLIP official	RN50×64	3-MLP	Det., Attr., Det.+Attr.	Acc., F1, AP, AUC

3 Preliminaries

Since its release, CLIP Radford et al. (2021) has emerged as a powerful and versatile feature extractor for a wide range of computer vision tasks. It is built on the joint training of two neural encoders: an image encoder and a text encoder, mapping respectively visual inputs and natural language descriptions into a shared embedding space. Training is performed using a contrastive objective that encourages alignment of matching image–text pairs while pushing apart mismatched ones. As a result, the learned representations capture high-level semantic information that is consistent across both visual and textual modalities, placing semantically related image–text pairs close together in the embedding space.

Although this training objective is explicitly semantic, several works have hypothesized and empirically shown that CLIP representations also encode lower-level perceptual cues Koutlis & Papadopoulos (2024); Amoroso et al. (2024). This implicit information can be exploited for tasks beyond direct semantic understanding, such as discriminating between real and generated images Sha et al. (2023); Ojha et al. (2023); Cioni et al. (2024); Cozzolino et al. (2024); Amoroso et al. (2024); Cocchi et al. (2023); Tan et al. (2025); Koutlis & Papadopoulos (2024). Further analyses suggest that, when used for detection, CLIP features progressively downplay explicit semantic content in favor of more abstract similarity relationships, suggesting that detection emerges from concept matching rather than true/false semantics Tan et al. (2025).

This behavior contrasts sharply with classical end-to-end detectors, which often rely on low-level, model-specific artifacts. Such detectors tend to learn decision boundaries tightly coupled to the fake generators observed during training, leading to poor generalization to unseen models, with the real class effectively acting as a catch-all for samples lacking known fake fingerprints Ojha et al. (2023). In contrast, CLIP provides a feature space learned from large-scale, diverse image-text supervision rather than explicit real/fake discrimination Radford et al. (2021). This reduces bias toward specific generative artifacts while yielding consistent embeddings across heterogeneous domains and out-of-distribution samples.

The following subsections present the design choices for the backbone and classifier in CLIP-based pipelines employed by state-of-the-art methods, together with a description of the datasets and evaluation metrics used in our study.

3.1 CLIP Backbones

State-of-the-art detectors typically rely on ViT-L/14 Ojha et al. (2023); Cozzolino et al. (2024), ViT-B/32 Sha et al. (2023); Amoroso et al. (2024), or ViT-B/16 Cioni et al. (2024); Cocchi et al. (2023) as the CLIP backbone for feature extraction. However, CLIP provides a broader set of architectures that can be employed for this task. In our experiments, we therefore evaluated all available CLIP backbones[2], including **ViT-B/32**, **ViT-B/16**, **ViT-L/14**, **ViT-L/14@336px**, **RN50**, **RN101**, **RN50×4**, **RN50×16**, and **RN50×64**. Table 2 summarizes the main characteristics of each backbone.

[2]We used the CLIP implementation released by OpenAI.

Table 2: CLIP backbones details. For ResNet-based backbones, N× scaling means that architecture uses approximately N× the compute of ResNet-50.

Backbone	Image Encoder Arch.	Key Characteristic	Features dim.	Parameters
ViT-B/32	ViT	Patch size 32×32	512	151M
ViT-B/16	ViT	Patch size 16×16	512	150M
ViT-L/14	ViT	Patch size 14×14	768	428M
ViT-L/14@336px	ViT	Patch size 14×14 (input 336px)	768	428M
RN50	ResNet	50 layers	1024	102M
RN101	ResNet	101 layers	512	120M
RN50×4	ResNet	50 layers, 4× scaling	640	178M
RN50×16	ResNet	50 layers, 16× scaling	768	291M
RN50×64	ResNet	50 layers, 64× scaling	1024	623M

As shown in the table, CLIP image encoders fall into two main categories: Vision Transformers (ViT) and ResNets.

The ViT models primarily differ in patch size and input resolution, resulting in different levels of spatial details and number of parameters. The '-B' variants represent the smallest backbones, with approximately 150M parameters, whereas the '-L' variants are roughly 2.8× larger.

ResNet-based backbones, instead, scale either through depth (RN50 vs. RN101) or via width multipliers (RN50×4, RN50×16, RN50×64), substantially increasing model dimension. Among the evaluated backbones, RN50 is the smallest (102M parameters), while RN50×64 is the largest (623M parameters), offering the widest representational capacity.

3.2 Classifiers

State-of-the-art approaches typically rely on standard machine learning classifiers, such as Multi-layer perceptrons (MLPs) Sha et al. (2023); Ojha et al. (2023), Support Vector Machines (SVM) Cozzolino et al. (2024), Logistic Regression (LR) Cioni et al. (2024); Cocchi et al. (2023); Amoroso et al. (2024), k-Nearest Neighbors (kNN) Cioni et al. (2024); Ojha et al. (2023); Cocchi et al. (2023), for both detection and attribution tasks. Therefore, we evaluated these classifiers in our experiments.

Multi-layer perceptrons. MLP is a feed-forward neural network capable of capturing complex, non-linear relationships between features. In our study, we considered three MLP variants, 1-MLP, 2-MLP, and 3-MLP, trained on the extracted CLIP features. The architectures differ in depth: the 1-MLP consists of a single fully connected layer, the 2-MLP includes one hidden layer with 256 units, and the 3-MLP uses two hidden layers with 512 and 256 units, respectively. All the classifiers were optimized using the Adam optimizer[3] with a learning rate of 3×10^{-4}. Moreover, to mitigate class imbalance, class-weighted cross-entropy loss was employed. Each configuration was trained for 10 epochs and repeated over 10 independent runs to ensure robustness and account for variability due to random initialization.

Support Vector Machines. SVM aim to find the optimal hyperplane that maximizes the separation margin between classes and perform well in high-dimensional feature spaces. We employed *scikit-learn* implementation of linear SVM (LinearSVC) with a maximum of 10,000 iterations, tuning the regularization parameter C over the range $[0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100]$.

Logistic Regression. LR provides a linear and interpretable decision boundary by directly modeling class probabilities. Similarly to SVM, we used the *scikit-learn* LogisticRegression implementation with a maximum of 10,000 iterations and we tuned the regularization parameter C over the same range.

k-Nearest Neighbors. kNN is a simple yet effective non-parametric method that classifies samples based on the majority label among their k nearest neighbors in the feature space. In our experiments, we used

[3]We primarily follow the training protocol proposed in De-Fake.

cosine distance as the similarity metric and optimized both the number of neighbors $k \in [1, 3, 5, 7]$ and weighting strategy (uniform or distance-based) via grid search.

SVM, LR, and kNN were implemented using the standard *scikit-learn* library with hyperparameter optimization performed via 5-fold cross-validation. For all classifiers, the best-performing configuration was selected and evaluated on the test set.

3.3 Dataset

For this study, we considered **GenImage** Zhu et al. (2023b) dataset, a large-scale benchmark dataset designed to evaluate AI-generated image detectors. The dataset contains over one million of real images derived from the ImageNet Deng et al. (2009) dataset and employs 8 generative models (7 diffusion models and 1 GAN) to produce fully synthetic images for each of the 1,000 ImageNet classes. In particular, the generative models include ADM Dhariwal & Nichol (2021), GLIDE Nichol et al. (2022), Midjourney mid (2023), Stable Diffusion v1.4 Rombach et al. (2022) (SDv4), Stable Diffusion v1.5 Rombach et al. (2022) (SDv5), VQDM Gu et al. (2022), Wukong wuk (2022), and BigGAN Brock et al. (2019). Each synthetic image is generated using the prompt "*a photo of {class}*", where *{class}* corresponds to one of the ImageNet labels.

Specifically, our experiments were conducted using the GenImage test set, composed by 6,000 synthetic images and 6,000 real images for each model, with the exception of SDv5, which comprises 8,000 synthetic and 8,000 real images. Since the real images from ImageNet within the GenImage test set do not include the class in their filenames, we replaced them with images from the ImageNet validation set, which explicitly provides class labels. In this way, we were able to split the dataset into 80% training and 20% testing, with a strict constraint preventing any prompt (i.e., class) from appearing in both subsets. Following this procedure, the training set consists of 40,000 synthetic and 39,950 real images (79,950 in total) corresponding to 800 distinct prompts, while the test set comprises 10,000 synthetic and 10,050 real images (20,050 in total) corresponding to additional 200 prompts not used during training.

Finally, to mitigate potential compression-related biases Grommelt et al. (2024); Ricker et al. (2024), synthetic images originally stored as lossless PNG files were converted to JPEG format to match the compression characteristics of real images. Specifically, we used a JPEG quality factor (QF) of 96, consistent with the average compression quality of the ImageNet dataset.

3.4 Evaluation Metrics

Following state-of-the-art approaches Ojha et al. (2023); Sha et al. (2023); Cozzolino et al. (2024); Cioni et al. (2024) for fully synthetic image detection and attribution, we considered four metrics for experiment evaluation: Balanced Accuracy (Acc.) with a fixed threshold of 0.5, F1 score (F1), Average Precision score (AP) and Area Under ROC Curve (AUC). These metrics capture different aspects of classification performance: Acc. measures the proportion of correctly classified samples while mitigating class imbalance, F1 captures the precision–recall trade-off, while AP and AUC rovide threshold-independent evaluations of the ranking capability of the model. Together, these metrics offer a comprehensive and reliable characterization of model performance across all tasks.

4 Experiments and Results

In this work, we considered three main tasks: *detection*, i.e., classifying an image as real (naturally captured by a camera) or synthetic (produced by a generative model); *attribution*, i.e., identifying the source model that generated a synthetic image; and *joint detection and attribution*, a combined task that simultaneously performs both.

To systematically evaluate different CLIP backbones and classifiers and identify the best configuration for the investigated tasks, we first extracted features from the GenImage dataset (detailed in Section 3.3) using CLIP's official preprocessing and all the backbones described in Section 3.1. We then trained each classifier introduced in Section 3.2 on these features for every considered task.

Table 3: Detection results in terms of **Acc. / F1** scores. Bold values indicate the best-performing backbone for each classifier. Bold values with a gray background indicate the the best-performing classifier for each backbone. Bold and underlined values denote the best overall result across all backbones and classifiers.

	1-MLP	**2-MLP**	**3-MLP**	**LR**	**SVM**	**kNN**	**AVG**
ViT-B/32	0.896 / 0.896	0.946 / 0.946	**0.951 / 0.951**	0.914 / 0.914	0.915 / 0.915	0.776 / 0.774	0.900 / 0.899
ViT-B/16	0.926 / 0.926	0.959 / 0.959	**0.964 / 0.964**	0.943 / 0.943	0.946 / 0.946	0.814 / 0.812	0.925 / 0.925
ViT-L/14	0.946 / 0.946	0.965 / 0.965	**0.968 / 0.968**	0.955 / 0.955	0.960 / 0.960	0.813 / 0.812	0.934 / 0.934
ViT-L/14@336px	0.968 / 0.968	0.982 / 0.982	**0.983 / 0.983**	0.975 / 0.975	0.978 / 0.978	0.869 / 0.868	0.959 / 0.959
RN50	0.938 / 0.938	0.967 / 0.967	**0.969 / 0.969**	0.950 / 0.950	0.956 / 0.956	0.810 / 0.809	0.932 / 0.931
RN101	0.910 / 0.910	0.954 / 0.954	**0.962 / 0.962**	0.935 / 0.935	0.936 / 0.936	0.789 / 0.788	0.914 / 0.914
RN50×4	0.945 / 0.945	0.971 / 0.971	**0.974 / 0.974**	0.958 / 0.958	0.958 / 0.958	0.806 / 0.805	0.935 / 0.935
RN50×16	0.976 / 0.976	0.987 / 0.987	**0.989 / 0.989**	0.982 / 0.982	0.983 / 0.983	0.899 / 0.899	0.969 / 0.969
RN50×64	**0.987 / 0.987**	**0.993 / 0.993**	**0.994 / 0.994**	**0.991 / 0.991**	**0.991 / 0.991**	**0.924 / 0.924**	**0.980 / 0.980**
AVG	0.944 / 0.944	0.969 / 0.969	**0.973 / 0.973**	0.956 / 0.956	0.958 / 0.958	0.833 / 0.832	—

Table 4: Detection results in terms of **AP / AUC** scores. Bold values indicate the best-performing backbone for each classifier. Bold values with a gray background indicate the the best-performing classifier for each backbone. Bold and underlined values denote the best overall result across all backbones and classifiers.

	1-MLP	**2-MLP**	**3-MLP**	**LR**	**SVM**	**kNN**	**AVG**
ViT-B/32	0.959 / 0.961	0.988 / 0.988	**0.989 / 0.989**	0.973 / 0.972	0.973 / 0.972	0.800 / 0.841	0.947 / 0.954
ViT-B/16	0.980 / 0.979	0.993 / 0.992	**0.993 / 0.993**	0.987 / 0.986	0.988 / 0.987	0.841 / 0.879	0.964 / 0.969
ViT-L/14	0.988 / 0.988	0.995 / 0.995	**0.995 / 0.995**	0.991 / 0.991	0.993 / 0.993	0.837 / 0.876	0.966 / 0.973
ViT-L/14@336px	0.996 / 0.996	0.998 / 0.998	**0.998 / 0.999**	0.997 / 0.997	0.998 / 0.998	0.894 / 0.926	0.980 / 0.986
RN50	0.982 / 0.983	0.995 / 0.995	**0.995 / 0.995**	0.988 / 0.989	0.991 / 0.991	0.844 / 0.876	0.966 / 0.971
RN101	0.967 / 0.969	0.991 / 0.991	**0.993 / 0.993**	0.981 / 0.982	0.981 / 0.982	0.823 / 0.860	0.956 / 0.963
RN50×4	0.984 / 0.986	0.996 / 0.996	**0.996 / 0.996**	0.991 / 0.991	0.991 / 0.992	0.841 / 0.876	0.966 / 0.973
RN50×16	0.997 / 0.997	0.999 / 0.999	**0.999 / 0.999**	0.998 / 0.998	0.998 / 0.998	0.929 / 0.948	0.987 / 0.990
RN50×64	**0.999 / 0.999**	**1.000 / 1.000**	**1.000 / 1.000**	**1.000 / 1.000**	**1.000 / 1.000**	**0.948 / 0.964**	**0.991 / 0.994**
AVG	0.984 / 0.984	0.995 / 0.995	**0.995 / 0.995**	0.990 / 0.990	0.990 / 0.990	0.862 / 0.894	—

The remainder of this section is organized as follows. We first report results on the detection task in Section 4.1, followed by attribution in Section 4.2, and then the joint detection and attribution task in Section 4.3. In Section 4.4 we evaluate the influence of training data size across the investigated backbones on performance. Finally, we compare the best-performing configuration with the state-of-the-art methods in Section 4.5, and we evaluate its generalization and robustness capabilities (Sections 4.6 and 4.7).

4.1 Fully Synthetic Image Detection

Tables 3 and 4 report the detection performance of various CLIP backbones paired with different classifiers. Overall, results indicate that performance increases quite consistently with model size. Larger and more expressive CLIP variants tend to provide more discriminative feature representations, while deeper classifiers are better equipped to exploit these features, leading to further gains on the detection task.

Considering Table 3, among the tested CLIP backbones, RN50×64 achieves the highest average accuracy (0.980), followed closely by RN50×16 (0.969) and ViT-L/14@336px (0.959). Specifically, these backbones are the largest among considered ones.

Regarding the tested classifiers, deeper MLPs, i.e., 3-MLP (0.973) and 2-MLP (0.969), yield the best overall performance, outperforming all other methods. Conversely, kNN shows the weakest overall performance (0.833), also exhibiting the strongest dependence on the underlying feature quality. In particular, its accuracy rises dramatically from 0.776 with ViT-B/32 backbone to 0.924 with RN50×64 (+0.148).

Overall, the best performance (0.994) is achieved with the **RN50×64+3-MLP combination**.

This behaviour is consistent across all evaluation metrics, including F1, as well as AP and AUC reported in Table 4, which likewise show clear gains with larger backbones and deeper MLP architectures.

Figure 2 show the Receiver Operating Characteristic (ROC) curves for all evaluated CLIP backbones, using the 3-MLP as classifier. All backbones exhibit excellent detection capability, with ROC curves that

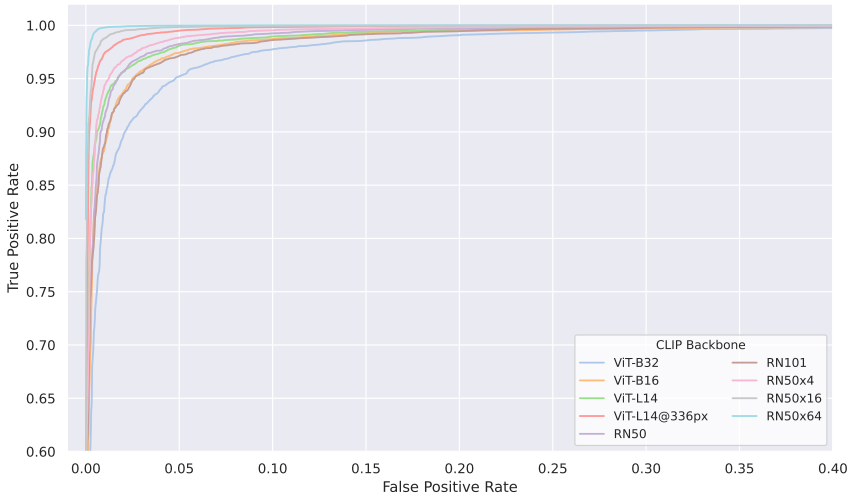

Figure 2: ROC curves for all evaluated backbones, using 3-MLP as classifier. Each curve shows the trade-off between the True Positive Rate and False Positive Rate across varying decision thresholds.

Table 5: Attribution results in terms of **Acc. / F1** scores. Bold values indicate the best-performing backbone for each classifier. Bold values with a gray background indicate the the best-performing classifier for each backbone. Bold and underlined values denote the best overall result across all backbones and classifiers.

	1-MLP	**2-MLP**	**3-MLP**	**LR**	**SVM**	**kNN**	**AVG**
ViT-B/32	0.725 / 0.695	0.783 / 0.746	**0.795** / 0.759	0.784 / **0.766**	0.779 / 0.752	0.512 / 0.505	0.730 / 0.704
ViT-B/16	0.726 / 0.699	0.794 / 0.761	**0.804** / 0.752	0.791 / **0.773**	0.790 / 0.766	0.544 / 0.535	0.741 / 0.714
ViT-L/14	0.748 / 0.718	0.803 / 0.772	**0.809** / 0.775	0.806 / **0.788**	0.804 / 0.780	0.533 / 0.526	0.750 / 0.726
ViT-L/14@336px	0.765 / 0.740	0.812 / 0.773	**0.818** / 0.782	0.816 / **0.798**	0.814 / 0.790	0.560 / 0.550	0.764 / 0.739
RN50	0.768 / 0.744	0.802 / 0.777	**0.808** / 0.768	0.801 / **0.786**	0.800 / 0.778	0.603 / 0.593	0.764 / 0.741
RN101	0.753 / 0.727	0.795 / 0.760	**0.804** / 0.771	0.798 / **0.782**	0.796 / 0.773	0.563 / 0.553	0.752 / 0.728
RN50×4	0.779 / 0.758	0.812 / 0.770	**0.821** / 0.782	0.812 / **0.797**	0.813 / 0.791	0.592 / 0.580	0.772 / 0.746
RN50×16	0.797 / 0.776	0.821 / 0.794	**0.825** / 0.785	0.819 / **0.804**	0.819 / 0.797	0.635 / 0.626	0.786 / 0.764
RN50×64	**0.811 / 0.792**	**0.831 / 0.810**	**0.833 / 0.809**	**0.829 / 0.813**	**0.831 / 0.810**	**0.627 / 0.620**	**0.794 / 0.776**
AVG	0.764 / 0.739	0.806 / 0.774	**0.813** / 0.776	0.806 / **0.790**	0.805 / 0.782	0.574 / 0.565	—

quickly rise toward the top-left corner. Larger and more expressive architectures, such as ResNet-50 variants (RN50×64, RN50×16) and ViT-L14 variants, perform slightly better, reaching near-perfect true positive rates even at very low false positive rates. Conversely, smaller backbones, like ViT-B/32, have a less steep curve, reflecting lower discriminative power, yet still maintain strong overall performance. Overall, the plot confirms that, when paired with the 3-MLP classifier, all CLIP backbones provide highly effective features for the detection task.

4.2 Source Attribution

For the attribution task, we employed the same CLIP-based features extracted for detection, changing only the number of output classes in each classifier to 8, corresponding to the generative models included in the dataset (real images are not considered in this task)[4].

Tables 5 and 6 report the obtained results. As in the detection task, performance increases quite consistently with the model size, confirming the benefit of larger and more expressive encoders combined with deep networks.

From Table 5, among the tested backbones, RN50×64 achieves again the highest average accuracy (0.794), followed closely by RN50×16 (0.786) and RN50×4 (0.772).

[4]Note that for the attribution task, random guessing corresponds to 1/8=0.125.

Table 6: Attribution results in terms of **AP / AUC** scores. Bold values indicate the best-performing backbone for each classifier. Bold values with a gray background indicate the the best-performing classifier for each backbone. Bold and underlined values denote the best overall result across all backbones and classifiers.

	1-MLP	**2-MLP**	**3-MLP**	**LR**	**SVM**	**kNN**	**AVG**
ViT-B/32	0.767 / 0.947	0.821 / 0.964	**0.832 / 0.967**	0.817 / 0.963	0.812 / 0.960	0.540 / 0.833	0.765 / 0.939
ViT-B/16	0.733 / 0.950	0.829 / 0.967	**0.838 / 0.969**	0.824 / 0.965	0.821 / 0.964	0.529 / 0.821	0.769 / 0.939
ViT-L/14	0.787 / 0.954	0.834 / 0.968	**0.840 / 0.970**	0.833 / 0.967	0.829 / 0.966	0.521 / 0.811	0.774 / 0.939
ViT-L/14@336px	0.806 / 0.960	0.842 / 0.970	**0.847 / 0.972**	0.840 / 0.970	0.837 / 0.969	0.558 / 0.833	0.788 / 0.946
RN50	0.805 / 0.959	0.833 / 0.968	**0.840 / 0.970**	0.828 / 0.967	0.826 / 0.966	0.643 / 0.885	0.796 / 0.952
RN101	0.790 / 0.954	0.832 / 0.967	**0.838 / 0.969**	0.831 / 0.967	0.826 / 0.964	0.592 / 0.858	0.785 / 0.946
RN50×4	0.811 / 0.961	0.842 / 0.970	**0.848 / 0.972**	0.839 / 0.969	0.839 / 0.969	0.589 / 0.846	0.795 / 0.948
RN50×16	0.824 / 0.965	0.846 / 0.972	**0.850 / 0.973**	0.842 / 0.970	0.841 / 0.970	**0.667 / 0.893**	**0.812 / 0.957**
RN50×64	**0.835 / 0.968**	**0.852 / 0.973**	**0.855 / 0.974**	**0.851 / 0.972**	**0.850 / 0.972**	0.622 / 0.864	0.811 / 0.954
AVG	0.800 / 0.958	0.837 / 0.969	**0.843 / 0.971**	0.834 / 0.968	0.831 / 0.967	0.585 / 0.849	—

Figure 3: Confusion matrix of Acc. obtained from RN50×64+3-MLP configuration for attribution task.

Regarding the classifiers, the 3-MLP achieves the best balanced accuracy (0.813), while 2-MLP, LR and SVM yield comparable performance. When considering the F1 score as evaluation metric, LR yields a slightly higher value than the MLP variants, although wuth a negligible gain (0.014). Conversely, kNN shows the weakest overall performance (0.574 of accuracy), also exhibiting the strongest dependence on the underlying feature quality.

The best overall performance, by considering also results in terms of AP and AUC in Table 6, is achieved with the **RN50×64+3-MLP combination**, same as detection task.

Figure 3 shows the confusion matrix for RN50×64+3-MLP combination. As we can notice, this configuration achieves nearly perfect performance for most of the generative models within the dataset. The lowest accuracies are observed for SDv4 (0.53), SDv5 (0.30), and Wukong (0.86). However, this outcome is not unexpected, as it reflects that architecturally similar models tend to be confused with one another. Specifically, SDv4 and SDv5 share the same base architecture, while Wukong is a diffusion model derived from Stable

Table 7: Joint detection and attribution results in terms of **Acc. / F1** scores. Bold values indicate the best-performing backbone for each classifier. Bold values with a gray background indicate the the best-performing classifier for each backbone. Bold and underlined values denote the best overall result across all backbones and classifiers.

	1-MLP	**2-MLP**	**3-MLP**	**LR**	**SVM**	**kNN**	**AVG**
ViT-B/32	0.725 / 0.753	0.785 / 0.811	**0.797 / 0.830**	0.763 / 0.835	0.742 / 0.818	0.475 / 0.617	0.715 / 0.777
ViT-B/16	0.734 / 0. 783	0.796 / 0.834	**0.810 / 0.850**	0.778 / 0.850	0.766 / 0.841	0.532 / 0.656	0.736 / 0.802
ViT-L/14	0.760 / 0.813	0.810 / 0.848	**0.817 / 0.857**	0.798 / 0.866	0.791 / 0.862	0.518 / 0.649	0.749 / 0.816
ViT-L/14@336px	0.785 / 0.846	0.824 / 0.868	**0.830 / 0.875**	0.818 / 0.883	0.818 / 0.882	0.575 / 0.700	0.776 / 0.843
RN50	0.770 / 0.809	0.809 / 0.848	**0.818 / 0.860**	0.794 / 0.864	0.789 / 0.860	0.558 / 0.676	0.756 / 0.819
RN101	0.752 / 0.781	0.798 / 0.831	**0.812 / 0.853**	0.782 / 0.852	0.767 / 0.839	0.523 / 0.646	0.738 / 0.798
RN50×4	0.785 / 0.827	0.822 / 0.859	**0.833 / 0.874**	0.809 / 0.874	0.803 / 0.868	0.574 / 0.681	0.771 / 0.829
RN50×16	0.814 / 0.866	0.836 / 0.880	**0.839 / 0.887**	0.830 / 0.892	0.823 / 0.8875	0.627 / 0.745	0.795 / 0.860
RN50×64	**0.830 / 0.884**	**0.848 / 0.899**	**0.848 / 0.898**	**0.842 / 0.901**	**0.840 / 0.899**	**0.642 / 0.764**	**0.808 / 0.875**
AVG	0.773 / 0.818	0.814 / 0.853	**0.823 / 0.865**	0.802 / 0.869	0.793 / 0.862	0.558 / 0.682	—

Table 8: Joint detection and attribution results in terms of **AP / AUC** scores. Bold values indicate the best-performing backbone for each classifier. Bold values with a gray background indicate the the best-performing classifier for each backbone. Bold and underlined values denote the best overall result across all backbones and classifiers.

	1-MLP	**2-MLP**	**3-MLP**	**LR**	**SVM**	**kNN**	**AVG**
ViT-B/32	0.739 / 0.960	0.804 / 0.976	**0.824 / 0.980**	0.805 / 0.976	0.798 / 0.974	0.499 / 0.846	0.745 / 0.952
ViT-B/16	0.759 / 0.965	0.819 / 0.980	**0.834 / 0.983**	0.819 / 0.979	0.817 / 0.978	0.540 / 0.853	0.765 / 0.956
ViT-L/14	0.785 / 0.972	0.832 / 0.983	**0.841 / 0.984**	0.831 / 0.983	0.832 / 0.982	0.529 / 0.844	0.775 / 0.958
ViT-L/14@336px	0.819 /0 .981	0.850 / 0.987	**0.856 / 0.987**	0.848 / 0.986	0.847 / 0.986	0.585 / 0.876	0.801 / 0.967
RN50	0.796 / 0.974	0.833 / 0.983	**0.843 / 0.985**	0.831 / 0.983	0.832 / 0.982	0.593 / 0.881	0.788 / 0.965
RN101	0.767 / 0.967	0.824 / 0.981	**0.837 / 0.984**	0.824 / 0.981	0.819 / 0.979	0.553 / 0.868	0.771 / 0.960
RN50×4	0.805 / 0.975	0.845 / 0.985	**0.855 / 0.986**	0.844 / 0.985	0.841 / 0.983	0.599 / 0.885	0.798 / 0.966
RN50×16	0.837 / 0.984	0.859 / 0.988	**0.863 / 0.988**	0.855 / 0.987	0.852 / 0.987	0.650 / 0.903	0.819 / 0.973
RN50×64	**0.853 / 0.987**	**0.867 / 0.989**	**0.869 / 0.989**	**0.865 / 0.988**	**0.863 / 0.988**	**0.661 / 0.913**	**0.830 / 0.970**
AVG	0.796 / 0.974	0.837 / 0.984	**0.847 / 0.985**	0.836 / 0.983	0.833 / 0.982	0.579 / 0.874	—

Diffusion (SD) and trained on a Chinese multimodal dataset that integrates a dedicated text encoder with the diffusion backbone.

4.3 Joint Detection and Attribution

We can perform detection and attribution simultaneously by including the real class as an additional category[5].

Tables 7 and 8 report the results obtained combining different CLIP backbones with various classifiers. As already observed for the individual detection and attribution tasks, performance consistently improves with larger CLIP backbones and more expressive classifiers.

From Table 7, among the tested backbones, RN50×64 achieves the highest average accuracy (0.808), followed closely by RN50×16 (0.795) and ViT-L/14@336px (0.776).

Across the tested classifiers, the 3-MLP outperforms all the others (average 0.823), followed by 2-MLP (0.814) and LR (0.802). As in the previous experiments, kNN shows the lowest overall performance (0.558), yet it exhibits the strongest dependence on the underlying feature quality.

This behaviour is again consistent across all evaluation metrics, including F1, as well as AP and AUC reported in Table 8, which likewise show clear gains with larger backbones and deeper MLP architectures.

The best overall performance (0.848) is still achieved with the **RN50×64+3-MLP combination**.

Figure 4 shows the confusion matrix obtained from the best configuration. Again, this approach achieves nearly perfect performance across all the evaluated architectures, for both real and synthetic images. The

[5]Note that for the joint detection and attribution task, random guessing corresponds to $1/9 = 0.111$.

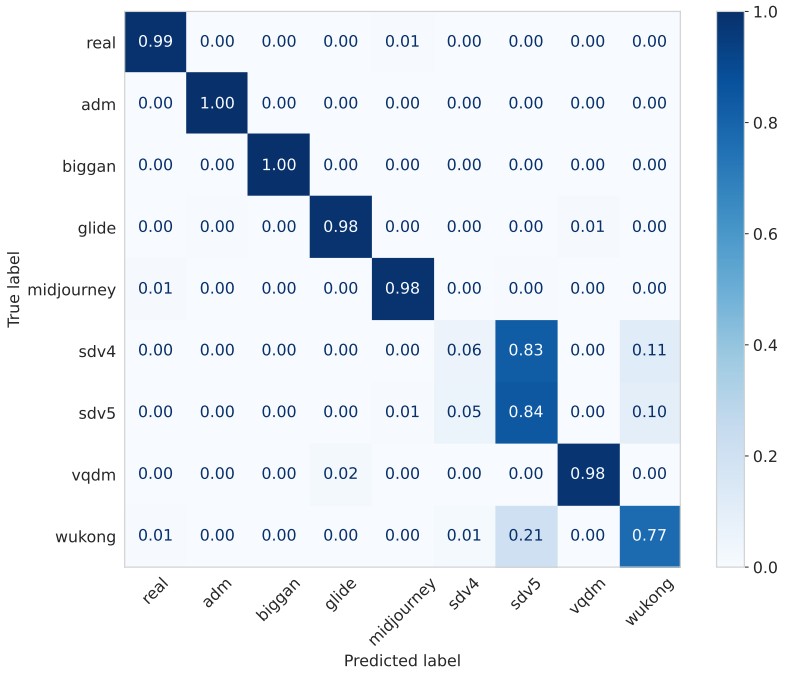

Figure 4: Confusion matrix of Acc. obtained from RN50×64+3-MLP configuration for joint detection and attribution task.

lowest accuracies are again observed for SDv4 (0.06), SDv5 (0.84), and Wukong (0.77). However, as mentioned for the attribution task, this outcome is not unexpected, as it reflects that architecturally similar models tend to be confused with one another.

4.4 Evaluation of Training Data Size

In this section, we evaluate how performance scales with training data. Across all considered tasks, we employed a 3-MLP classifier and, for each backbone, we varied the size of the training set by changing the number of prompts included in training. In particular, we adopted 1, 5, 10, 50, 100, 300, 500 and 800 prompts (where each prompt results in 100 training images). To ensure a fair comparison, the test set is kept fixed across all experiments.

Figure 5 illustrates the detection task performance in terms of accuracy. The results vary considerably across backbones as the number of training images changes. For instance, when using a single prompt in training, the RN50 and ViT-B/32 backbones achieve an accuracy slightly above 0.65, whereas the RN50×64 and RN50×16 backbones already exceed 0.90. When increasing to 100 training prompts (10k training images), all backbones surpass 0.90. Beyond this point, performance stabilizes, with only marginal improvements observed as additional training data are introduced. In particular, the RN50×64 backbone exhibits an accuracy gain of only 0.008 when increasing the number of training prompts from 100 to 800, while consistently remaining the best-performing model across all training set sizes.

A similar trend is also observed for both the attribution and the joint detection and attribution tasks. More details about these experiments can be found in Appendix A.1.

From this point onward, we will refer to the best-performing configuration across all experiments (RN50×64+3-MLP) as **DejaVu**.

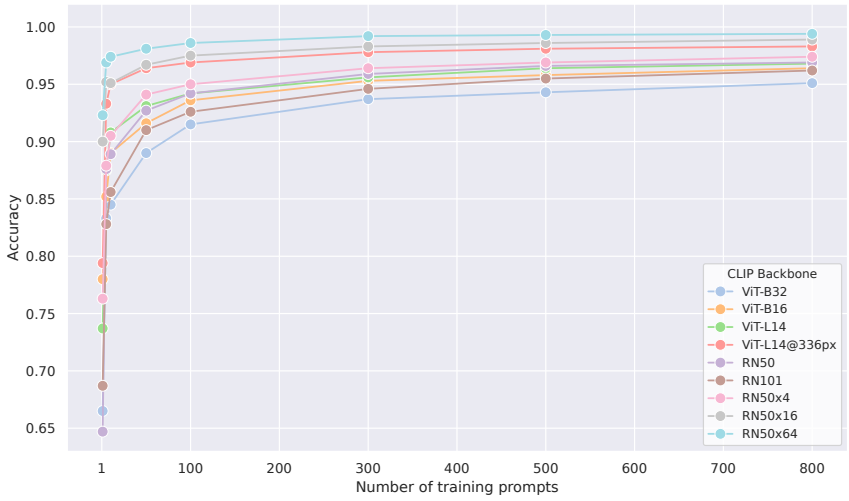

Figure 5: Detection task. Accuracies for CLIP backbones with a 3-MLP classifier, varying the number of training prompts.

Table 9: Comparison of DejaVu vs. state-of-the-art CLIP-based methods for detection (top) and attribution (bottom) tasks. Reported values are averaged on ten independent runs.

	Method	Acc.	F1	AP	AUC
Det.	De-Fake*	0.930	0.930	0.979	0.978
	UnivFD	0.922	0.922	0.976	0.977
	DejaVu	**0.994**	**0.994**	**1.000**	**1.000**
Attr.	De-Fake*	0.758	0.742	0.792	0.955
	UnivFD$^{\diamond}$	0.436	0.392	0.444	0.813
	UnivAttr†	0.829	**0.816**	0.848	0.972
	DejaVu	**0.833**	0.809	**0.855**	**0.974**

4.5 Comparison with State-of-the-Art Approaches

To evaluate the effectiveness of the proposed approach, we compare DejaVu against recent state-of-the-art CLIP-based detection and attribution methods.

For a fair comparison, all methods were re-trained on the same training set used for DejaVu and evaluated on the same test data, using the official code provided by the original authors. However, some methods required additional adjustments to align with our evaluation protocol. Specifically, De-Fake Sha et al. (2023) assumes access to the original prompts for real images during training, which is rarely feasible in practice. Since real images come from ImageNet, which does not provide prompts, we generated prompts for both real and fake images using BLIP Li et al. (2022), following the same procedure adopted by the authors at inference time when prompts are not available. This adjustment (indicated with an $*$ in the table) prevents potential biases, as generating prompts only for real images could inadvertently cue "realness", as prompt generation is also used at test time. Although prompts have limited impact on performance in this setup, this approach ensures the most faithful reproduction of De-Fake's original evaluation conditions.

UnivFD Ojha et al. (2023) was originally designed for detection only; however, by simply adjusting the number of neurons in the final layer and replacing the Binary Cross-Entropy Loss with a Cross-Entropy Loss, it can also operate in an attribution setting (marked with a $\diamond$ in the table).

Table 10: DejaVu performance under different preprocessing schemes.

Preprocessing	Acc.	F1	AP	AUC
De-Fake	0.987	0.987	0.999	0.999
UnivFD	0.874	0.873	0.961	0.962
CLIP	**0.994**	**0.994**	**1.000**	**1.000**

Table 11: DejaVu performance when extracting features from different layers for detection and attribution tasks.

	Feat. Layer	Acc.	F1	AP	AUC
Det.	Layer 1	**0.996**	**0.996**	0.992	0.992
	Layer 2	0.984	0.984	0.996	0.998
	Layer 3	0.974	0.974	0.987	0.993
	Layer 4	**0.996**	**0.996**	**1.000**	**1.000**
	Final Layer	0.994	0.994	**1.000**	**1.000**
Attr.	Layer 1	0.707	0.665	0.758	0.952
	Layer 2	0.734	0.685	0.771	0.951
	Layer 3	0.488	0.418	0.513	0.885
	Layer 4	**0.840**	0.807	**0.863**	**0.976**
	Final Layer	0.833	**0.809**	0.855	0.974

Similarly, UnivAttr Cioni et al. (2024), which was originally developed for an open-set scenario, is here evaluated under a close-set setting to ensure a fair comparison (indicated with a †).

Results of these comparisons are summarized in Table 9. Overall, DejaVu consistently outperforms the compared state-of-the-art approaches in both detection and attribution tasks, particularly in terms of accuracy and F1 score, confirming the effectiveness of the identified configuration.

A closer inspection of the results highlights several method-specific nuances. In particular, De-Fake and UnivFD, despite being strong baselines, obtain slightly lower results when compared to their corresponding backbone and classifier combinations. For instance, when considering the detection task in Tables 3 and 4, the corresponding configuration for De-Fake (ViT-B/32+3-MLP) achieves 0.951 of accuracy against 0.930 with the original method, while UnivFD (ViT-L/14+1-MLP) yield 0.946 of accuracy against 0.922 with the original approach. We found out that this performance gap is due to differences in preprocessing. Both De-Fake and UnivFD do not employ the official CLIP preprocessing as in DejaVu, but customize versions of it. More details about the preprocessing procedures can be found in Appendix A.2.

Table 10 reports the results obtained by applying De-Fake and UnivFD preprocessing to DejaVu, while keeping all other settings unchanged. Specifically, we observed that DejaVu experiences a negligible drop in accuracy when using De-Fake preprocessing (-0.007), whereas UnivFD preprocessing leads to a substantial decrease (-0.12). The same pattern is observed across all other evaluation metrics.

Interestingly, UnivAttr achieves higher performance by extracting features from an intermediate CLIP layer rather than the final one, using a ViT-B/16 backbone. To investigate this, we also experimented with using intermediate-layer features in DejaVu. As reported in Table 11, for the considered backbone, the differences were minimal, indicating that the final-layer features from our ResNet-based backbone are already highly informative for both detection and attribution tasks.

4.6 Cross-model Generalization

To further investigate the generalization capabilities of DejaVu, we evaluated its performance when trained on a single generative model and tested on the remaining unseen ones.

Table 12: Cross-model generalization of DejaVu for detection (**Acc. / F1**). Rows represent training models, while columns correspond to unseen test models. Row blocks correspond to the model clusters identified in the feature space. The last row reports results obtained by jointly training on best representative models from both clusters (VQDM and Wukong).

Model	ADM	BigGAN	GLIDE	Midjourney	SDv4	SDv5	VQDM	Wukong	AVG
ADM	**1.000 / 1.000**	0.690 / 0.920	0.958 / 0.991	0.500 / 0.843	0.501 / 0.843	0.500 / 0.799	0.979 / 0.995	0.503 / 0.844	0.696 / 0.665
BigGAN	0.888 / 0.974	**1.000 / 0.999**	0.961 / 0.991	0.503 / 0.844	0.502 / 0.844	0.502 / 0.801	0.940 / 0.986	0.504 / 0.845	0.716 / 0.692
GLIDE	0.943 / 0.987	0.847 / 0.965	**0.999 / 1.000**	0.507 / 0.847	0.501 / 0.844	0.500 / 0.799	0.963 / 0.992	0.502 / 0.844	0.712 / 0.686
VQDM	0.994 / 0.998	0.889 / 0.975	0.990 / 0.998	0.501 / 0.844	0.513 / 0.850	0.512 / 0.807	**1.000 / 1.000**	0.533 / 0.859	**0.732 / 0.712**
Midjourney	0.571 / 0.872	0.499 / 0.841	0.700 / 0.918	**0.990 / 0.991**	0.807 / 0.948	0.808 / 0.936	0.514 / 0.848	0.662 / 0.905	0.698 / 0.671
SDv4	0.528 / 0.857	0.510 / 0.848	0.525 / 0.855	0.659 / 0.909	**0.997 / 0.999**	**0.997 / 0.999**	0.606 / 0.890	0.985 / 0.996	0.737 / 0.718
SDv5	0.521 / 0.853	0.525 / 0.855	0.522 / 0.854	0.612 / 0.892	**0.997 / 0.998**	0.996 / 0.998	0.602 / 0.888	0.981 / 0.995	0.730 / 0.710
Wukong	0.565 / 0.873	0.556 / 0.869	0.530 / 0.857	0.615 / 0.893	**0.996 / 0.997**	0.995 / 0.996	0.721 / 0.928	0.995 / 0.997	**0.757 / 0.742**
VQDM+Wukong	0.994 / 0.996	0.973 / 0.991	0.990 / 0.995	0.620 / 0.894	0.995 / 0.996	0.995 / 0.996	**0.998 / 0.996**	0.995 / 0.996	**0.947 / 0.947**

Table 13: Cross-model generalization of DejaVu for detection (**AP / AUC**). Rows represent training models, while columns correspond to unseen test models. Row blocks correspond to the model clusters identified in the feature space. The last row reports results obtained by jointly training on best representative models from both clusters (VQDM and Wukong).

Model	ADM	BigGAN	GLIDE	Midjourney	SDv4	SDv5	VQDM	Wukong	AVG
ADM	**1.000 / 1.000**	0.984 / 0.998	0.997 / 1.000	0.276 / 0.798	0.338 / 0.843	0.391 / 0.841	0.999 / 1.000	0.452 / 0.881	0.924 / 0.917
BigGAN	0.992 / 0.999	**1.000 / 1.000**	0.997 / 1.000	0.547 / 0.911	0.616 / 0.943	0.674 / 0.943	0.995 / 0.999	0.559 / 0.924	0.965 / 0.964
GLIDE	0.998 / 1.000	0.996 / 1.000	**1.000 / 1.000**	0.614 / 0.907	0.561 / 0.914	0.617 / 0.915	0.999 / 1.000	0.512 / 0.888	0.956 / 0.951
VQDM	1.000 / 1.000	0.993 / 0.999	1.000 / 1.000	0.345 / 0.834	0.665 / 0.942	0.716 / 0.944	**1.000 / 1.000**	0.713 / 0.943	**0.961 / 0.957**
Midjourney	0.528 / 0.917	0.259 / 0.826	0.777 / 0.971	**0.996 / 0.999**	0.872 / 0.981	0.902 / 0.981	0.346 / 0.861	0.680 / 0.934	0.934 / 0.936
SDv4	0.675 / 0.953	0.681 / 0.962	0.701 / 0.963	0.853 / 0.972	**0.999 / 1.000**	**0.999 / 1.000**	0.862 / 0.982	0.997 / 1.000	0.979 / 0.980
SDv5	0.644 / 0.947	0.801 / 0.978	0.695 / 0.961	0.841 / 0.973	**1.000 / 1.000**	**1.000 / 1.000**	0.878 / 0.985	0.998 / 1.000	0.980 / 0.981
Wukong	0.712 / 0.957	0.738 / 0.965	0.616 / 0.941	0.770 / 0.957	**0.998 / 1.000**	**0.998 / 1.000**	0.894 / 0.985	**0.998 / 1.000**	**0.976 / 0.977**
VQDM+Wukong	0.998 / 1.000	0.992 / 0.999	0.997 / 1.000	0.728 / 0.951	0.999 / 1.000	0.999 / 1.000	**1.000 / 1.000**	0.999 / 1.000	**0.995 / 0.994**

The results reported in Tables 12 and 13 reveal a clear and consistent generalization pattern: while training and testing on the same model yields near-perfect performance, cross-model generalization is highly non-uniform and strongly dependent on the generative model used for training. In particular, models trained on certain generators generalize well to a subset of unseen models, whereas performance degrades substantially when testing on others, often approaching random-guess behavior.

This behavior is further supported by the plot illustrated in Figure 6, which shows a t-SNE Maaten & Hinton (2008) visualization of feature embeddings extracted using the RN50×64 backbone for a single shared prompt. We observe a well-defined cluster of real images and two distinct clusters of fake images. The first synthetic cluster groups ADM, BigGAN, GLIDE, and VQDM, while the second includes Midjourney, SDv4, SDv5, and Wukong.

Consistently with this structure, when DejaVu is trained on a model belonging to one cluster, it tends to achieve improved performance on the other models within the same cluster, suggesting the presence of shared underlying characteristics in their generated images. Conversely, generalization across clusters is significantly weaker, highlighting the impact of inter-model differences on detection performance. Motivated by this observation, we further train DejaVu using one representative model from each cluster. Specifically, we selected VQDM and Wukong, as they yield the highest average balanced accuracy when used individually for training (0.732 and 0.757, respectively). As reported in the last row of Tables 12 and 13, jointly training on these two complementary models substantially improves generalization, achieving an average balanced accuracy of approximately 0.95. Compared to training on VQDM or Wukong alone, this joint setting provides an absolute gain of roughly +0.22 and +0.19 balanced accuracy, respectively, further highlighting the benefit of jointly modeling diverse generative families. Despite the overall improvement, Midjourney remains the most challenging generator, exhibiting comparatively lower detection performance even under the joint-training setting (0.62 accuracy). Owing to its proprietary nature, the underlying cause of this behavior cannot be directly assessed. Nonetheless, a plausible hypothesis is that Midjourney employs architectural choices or prompt preprocessing strategies that differ substantially from those adopted by other models, thereby reducing detection effectiveness.

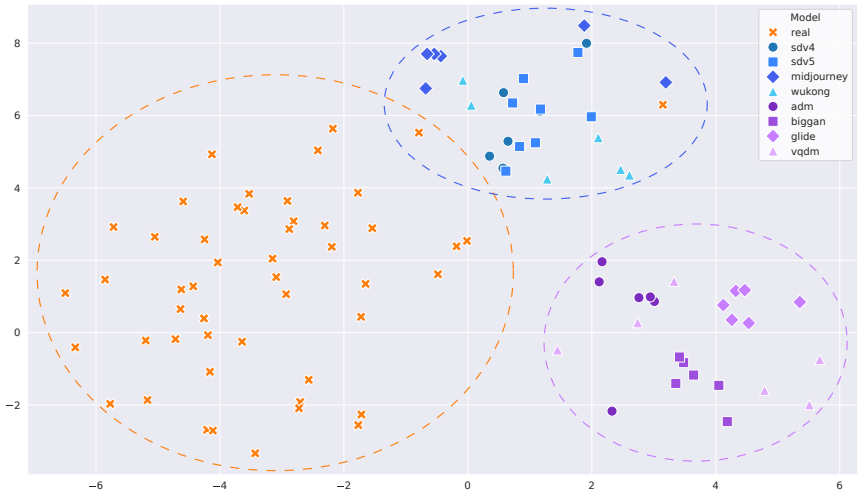

Figure 6: t-SNE visualization of feature embeddings from RN50×64 backbone for real and synthetic images, all corresponding to the same textual prompt.

Table 14: Jpeg compression impact on DejaVu detection. Baseline refers to uncompressed images.

QF	Acc.	F1	AP	AUC
Baseline	**0.994**	**0.994**	**1.000**	**1.000**
95	0.986	0.986	0.999	0.999
90	0.987	0.987	0.999	0.999
85	0.993	0.993	**1.000**	**1.000**
80	0.971	0.971	0.999	0.999
75	0.985	0.985	0.999	0.999
70	0.967	0.967	0.999	0.999
65	0.850	0.847	0.996	0.997
60	0.799	0.791	0.996	0.997

4.7 Robustness Evaluation

Finally, we measured the robustness of DejaVu to a range of real-world post-processing operations commonly applied to images, such as those introduced by social media sharing, image editing software, automatic beautification tools, and online compression pipelines. In particular, we considered JPEG compression, blurring, resizing, and cropping. Due to space and logistical constrains, the robustness is evaluated only on the detection task.

JPEG Compression JPEG compression is a lossy process that reduces image quality by discarding information; the quality factor (QF) controls the amount of compression, with lower QF values corresponding to stronger compression. We evaluated the impact of JPEG compression on DejaVu by varying the quality factor from 60 to 95. Both real and synthetic images were compressed to the target quality factor to assess how varying compression levels affect detection performance. Results are presented in Table 14. As expected, higher compression levels (corresponding to lower QF values) lead to reduced performance, likely due to the loss of forensic traces in the images. In particular, for QF ≤ 65, the performance degradation becomes more pronounced, with an accuracy drop of 0.144 compared to the baseline.

However, this degradation is not uniform across models: as shown in Figure 7, the observed drop is primarily driven by a subset of models (SDv4, SDv5, Wukong, and BigGAN). Notably, these correspond to the only GAN-based model in the dataset (BigGAN) and three Stable Diffusion variants, whereas the remaining models exhibit largely stable performance across compression levels, maintaining accuracy above 90%.

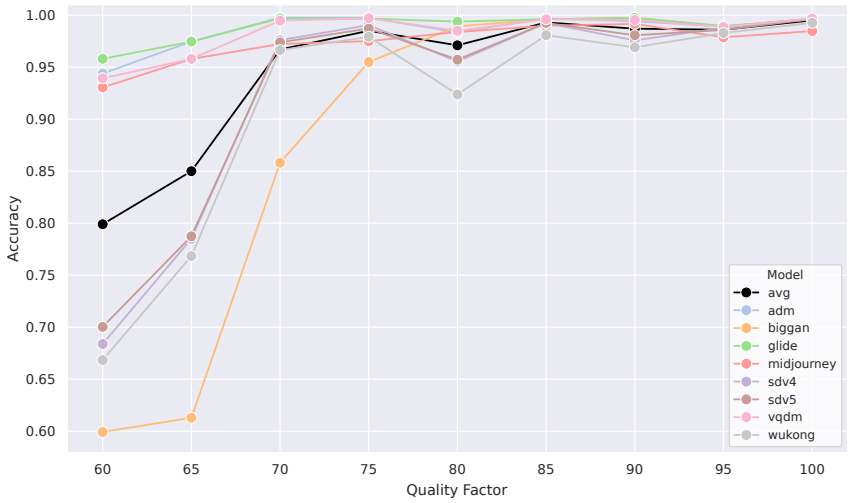

Figure 7: Detection accuracies of DejaVu across different models at varying JPEG quality factors (QF). QF=100 represents the baseline.

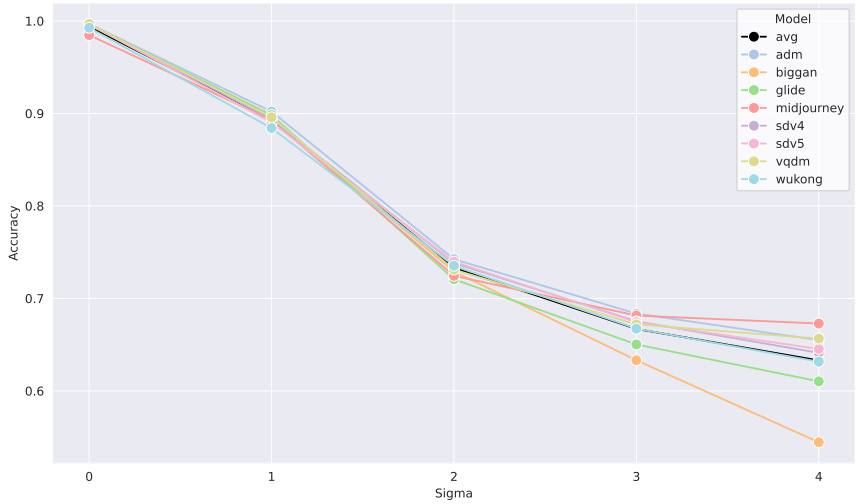

Figure 8: Detection accuracies of DejaVu across different models at varying standard deviation sigmas for Gaussian blurring. σ=0 represents the baseline.

Table 15: Gaussian Blur impact on DejaVu detection. Baseline refers to not blurred images.

σ	Acc.	F1	AP	AUC
Baseline	**0.994**	**0.994**	**1.000**	**1.000**
1	0.894	0.893	0.979	0.980
2	0.732	0.715	0.902	0.909
3	0.667	0.638	0.805	0.820
4	0.633	0.604	0.733	0.744

Table 16: Resizing impact on DejaVu detection. Baseline refers to not resized images.

r	Acc.	F1	AP	AUC
0.5	0.890	0.890	0.956	0.952
0.7	0.948	0.948	0.898	0.989
0.9	0.983	0.983	0.998	0.998
Baseline	**0.994**	**0.994**	**1.000**	**1.000**
1.1	0.988	0.988	0.999	0.999
1.3	0.988	0.988	0.999	0.999
1.5	0.991	0.991	**1.000**	**1.000**

Gaussian Blur The Gaussian blurring filter smooths an image by applying a sequence of extended box filters approximating a Gaussian kernel. We evaluated DejaVu's performance under different levels of blurring, controlled by the standard deviation σ of the Gaussian kernel, considering $\sigma \in \{1, 2, 3, 4\}$.

Results are presented in Table 15. We observe that higher σ values lead to more pronounced performance degradation, reflecting the loss of forensic traces. The drop is already noticeable at $\sigma = 1$, where accuracy decreases from 0.994 to 0.894, and becomes substantial at $\sigma = 4$ (accuracy 0.633), at which point, however, the image is heavily blurred. As shown in Figure 8, up to $\sigma = 2$ all models exhibit comparable degradation, while for larger σ values the drop is driven primarily by BigGAN and GLIDE.

Resize Image resizing is a common operation that changes the spatial resolution of an image, potentially affecting forensic traces. We measured DejaVu's robustness to different resizing by scaling images with factors $r \in \{0.5, 0.7, 0.9, 1.1, 1.3, 1.5\}$. Each factor r multiplies the original image dimensions; for instance, a factor $r = 1.5$ increases both width and height by 50%, while a factor of $r = 0.5$ reduces them by half.

Table 16 present the results of this experiment. While resizing the original image to higher resolutions does not significantly affect DejaVu's performance, reducing the resolution has a more noticeable impact. In particular, while resizing to 0.9 or 0.7 slightly decreases performance, halving the original resolution results in a substantial drop (-0.104 of accuracy). However, as shown in Figure 9, this degradation is largely driven by the GAN-based model (BigGAN). Diffusion-based models are much less affected: even at a resize factor of 0.5, their accuracy remains between 0.90 and 0.95, whereas BigGAN is the only model whose performance drops substantially, reaching 0.70.

Crop Finally, to assess the impact of spatial content reduction on detection, we evaluated DejaVu's robustness to cropping operations. Specifically, we applied different cropping factors $c \in \{0.3, 0.5, 0.7, 0.9\}$. Each factor c represents the proportion of the original image area that is retained; for example, $c = 0.9$ preserves 90% of the image. Both random and center cropping strategies were considered.

Results are presented in Table 17. Overall, the model is robust to cropping, with only minor decreases in performance. As expected, more severe cropping results in a more pronounced performance drop, likely due to the loss of critical forensic traces. Interestingly, random cropping slightly reduces performance more than center cropping for smaller retained areas, suggesting that preserving the central content of the image helps retain key forensic traces. However, despite the reductions, all accuracies remain above 0.92, indicating strong resilience to cropping. As shown in Figures 10 and 11, this moderate degradation is not uniform across models. In particular, Midjourney and BigGAN are the main contributors to the performance drop,

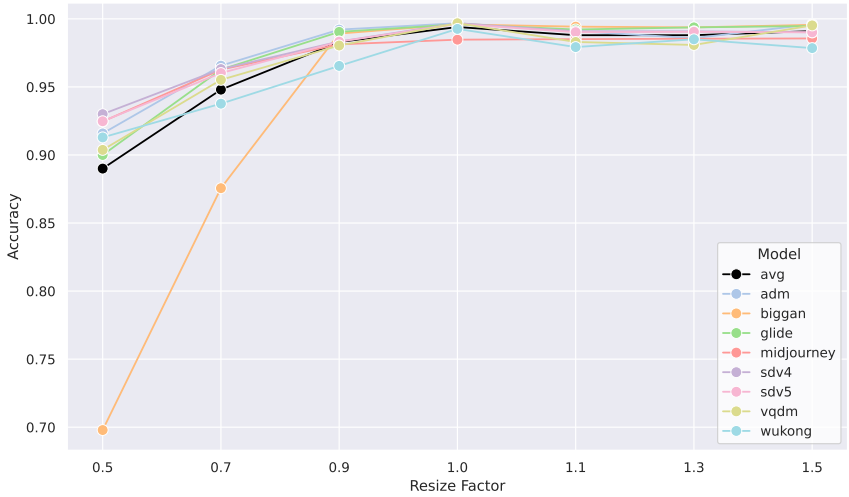

Figure 9: Detection accuracies of DejaVu across different models at varying resizing factors. Resize factor equals to 1.0 represents the baseline.

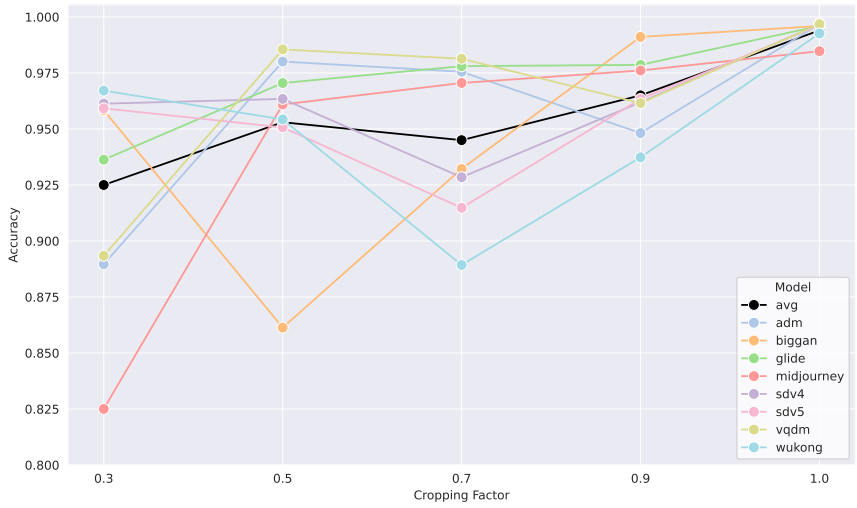

Figure 10: Detection accuracies of DejaVu across different models at varying random cropping factors. Cropping factor equals to 1.0 represents the baseline.

Table 17: Cropping impact on DejaVu detection. Baseline refers to not cropped images.

c	Random Crop				Center Crop			
	ACC	F1	AP	AUC	ACC	F1	AP	AUC
Baseline	**0.994**	**0.994**	**1.000**	**1.000**	**0.994**	**0.994**	**1.000**	**1.000**
0.9	0.965	0.965	0.998	0.998	0.958	0.958	0.997	0.997
0.7	0.945	0.945	0.996	0.997	0.944	0.944	0.995	0.996
0.5	0.953	0.953	0.993	0.994	0.969	0.969	0.995	0.995
0.3	0.925	0.925	0.980	0.981	0.943	0.943	0.986	0.988

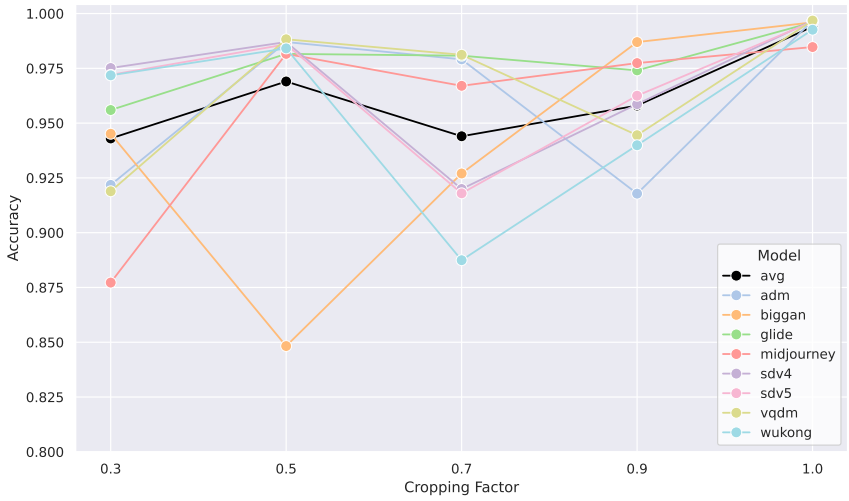

Figure 11: Detection accuracies of DejaVu across different models at varying center cropping factors. Cropping factor equals to 1.0 represents the baseline.

with accuracies dropping to approximately 0.82 and 0.85. All other models exhibit only marginal decreases, indicating robust detection performance.

5 What we have learned

Fully synthetic image detection and attribution. Experimental results show that larger and more expressive CLIP backbones provide more discriminative feature representations. Indeed, moving from the smallest backbone (ViT-B/32) to the largest (RN50×64) improves the average accuracy by approximately 9% for independent detection and attribution tasks, and by about 13% for the joint detection and attribution task. Among the classifiers, deeper MLPs can better leverage these features than the others considered, resulting in further performance gains across all tasks. In particular, moving from the smallest to the largest backbone increases average accuracy by roughly 17% for detection, 42% for attribution, and 48% for joint detection and attribution. These higher gains are largely driven by the comparatively lower performance of the kNN classifier. Specifically, kNN classifies a query image by computing the distances between its CLIP-extracted features and those of the training images, assigning the query to the class most represented among its k nearest neighbors. However, this procedure may not be optimal for this task Mistretta et al. (2025), as it operates in an *intra-modality* fashion, i.e., image-to-image comparison. We argue that adopting a *modality inversion* strategy, performing image-to-text comparison, could further improve the classifier's performance.

Training data size. Our experiments show that CLIP-based detectors do not require large amount of data to effectively perform detection and attribution tasks. Using the same classifier across all investigated backbones, we proved that 10,000 training images (5,000 real and 5,000 synthetic) are enough to achieve and surpass 90% of accuracy. Remarkably, for larger backbones such as RN50×64 and RN50×16, only 100

images (50 real and 50 synthetic) depicting the same subject are sufficient to reach 90% of accuracy, making these backbones suitable for few-shot scenarios where few amount of data are available.

Best performing configuration. We identified the best performing configuration, referred to as DejaVu, which employs RN50×64 backbone for feature extraction and a 3-MLP as classifier. Overall, DejaVu consistently outperforms existing state-of-the-art detectors in both detection and attribution tasks, confirming its effectiveness. We also observed that using CLIP's official image preprocessing, rather than customized version of it, results in higher performance of some state of the art methods.

Cross-model generalization. We find that training and testing DejaVu on the same generative model yields near-perfect performance, whereas cross-model generalization depends on the chosen training model. Specifically, we identified two generative clusters and observed that when DejaVu is trained on a model belonging to one cluster, it tends to achieve improved performance on other models within the same cluster, suggesting the presence of shared underlying characteristics in their generated images. Indeed, we observed that when training DejaVu using one representative model from each cluster, it can achieve near-optimal performance, with accuracy reaching 0.95 compared to 0.99 when all generative models are available during training.

Robustness. Our experiments show that DejaVu is generally robust to common image post-processing operations, although the degree of robustness varies across perturbation types and generative models. For most of diffusion-based models, DejaVu is robust to JPEG compression, whereas performance drops more noticeably for Stable Diffusion variants and GAN-generated images under stronger compression levels. Gaussian blurring leads to notable performance degradation for higher σ values, but consistently with the fact that the image becomes increasingly unrecognizable. Resizing operations have a limited impact, particularly when images are scaled above the input resolution, while performance drops for smaller resolutions are mainly driven by GAN-generated images. Similarly, both random and center cropping operations have only a minor effect on accuracy.

6 Conclusion

In this work, we investigated CLIP-based approaches for fully synthetic image detection and attribution, analyzing the impact of backbone selection, classifier design, image preprocessing strategy, and training data availability. Our results demonstrate that larger and more expressive CLIP backbones yield significantly more discriminative representations, especially when coupled with deeper MLP classifiers, leading to substantial performance improvements across detection, attribution, and joint detection and attribution tasks. Moreover, we show that high performance can be achieved even with limited training data, highlighting the suitability of CLIP-based detectors for few-shot scenarios. The proposed DejaVu configuration, combining the RN50×64 backbone with a 3-layer MLP, consistently outperforms existing state-of-the-art methods, confirming its effectiveness. While cross-model generalization remains dependent on the training generative models, our analysis reveals meaningful clustering behavior among generators, enabling near-optimal performance with a reduced set of training models. Finally, we proved that DejaVu is resilient to several common post-processing operations, particularly for diffusion-based models, although challenges persist for GAN-generated images and severe degradations. Overall, these findings underscore the critical role of backbone selection and classifier design when leveraging CLIP-based representations for synthetic image forensics. Future work may further explore the applicability of this approach to the detection of partially generated content, where only a subset of the image has been synthetically produced.

References

Wukong. `https://xihe.mindspore.cn/modelzoo/wukong`, 2022.

Midjourney. `https://www.midjourney.com/`, 2023.

Michael Albright, Scott McCloskey, and ACST Honeywell. Source generator attribution via inversion. In *CVPR workshops*, volume 8, pp. 3, 2019.

Roberto Amoroso, Davide Morelli, Marcella Cornia, Lorenzo Baraldi, Alberto Del Bimbo, and Rita Cucchiara. Parents and children: Distinguishing multimodal deepfakes from natural images. *ACM Transactions on Multimedia Computing, Communications and Applications*, 21(1):1–23, 2024.

Luca Bindini, Giulia Bertazzini, Daniele Baracchi, Dasara Shullani, Paolo Frasconi, and Alessandro Piva. Tiny autoencoders are effective few-shot generative model detectors. In *2024 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6. IEEE, 2024.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

George Cazenavette, Avneesh Sud, Thomas Leung, and Ben Usman. Fakeinversion: Learning to detect images from unseen text-to-image models by inverting stable diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10759–10769, 2024.

Dario Cioni, Christos Tzelepis, Lorenzo Seidenari, and Ioannis Patras. Are clip features all you need for universal synthetic image origin attribution? In *European Conference on Computer Vision*, pp. 363–382. Springer, 2024.

Federico Cocchi, Lorenzo Baraldi, Samuele Poppi, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Unveiling the impact of image transformations on deepfake detection: An experimental analysis. In *International Conference on Image Analysis and Processing*, pp. 345–356. Springer, 2023.

Davide Alessandro Coccomini, Andrea Esuli, Fabrizio Falchi, Claudio Gennaro, and Giuseppe Amato. Detecting images generated by diffusers. *PeerJ Computer Science*, 10:e2127, 2024.

Riccardo Corvi, Davide Cozzolino, Giovanni Poggi, Koki Nagano, and Luisa Verdoliva. Intriguing properties of synthetic images: from generative adversarial networks to diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 973–982, 2023a.

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

Davide Cozzolino, Giovanni Poggi, Riccardo Corvi, Matthias Nießner, and Luisa Verdoliva. Raising the bar of ai-generated image detection with clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4356–4366, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Hany Farid. Lighting (in) consistency of paint by text. *arXiv preprint arXiv:2207.13744*, 2022a.

Hany Farid. Perspective (in) consistency of paint by text. *arXiv preprint arXiv:2206.14617*, 2022b.

Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition. In *International conference on machine learning*, pp. 3247–3258. PMLR, 2020.

Patrick Grommelt, Louis Weiss, Franz-Josef Pfreundt, and Janis Keuper. Fake or jpeg? revealing common biases in generated image detection datasets. In *European Conference on Computer Vision*, pp. 80–95. Springer, 2024.

Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10696–10706, 2022.

Luca Guarnera, Oliver Giudice, and Sebastiano Battiato. Mastering deepfake detection: A cutting-edge approach to distinguish gan and diffusion-model images. *ACM Transactions on Multimedia Computing, Communications and Applications*, 20(11):1–24, 2024.

Syou Hirofumi, Kazuto Fukuchi, Yohei Akimoto, and Jun Sakuma. Did you use my gan to generate fake? post-hoc attribution of gan generated images via latent recovery. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

Christos Koutlis and Symeon Papadopoulos. Leveraging representations from intermediate encoder-blocks for synthetic image detection. In *European Conference on Computer Vision*, pp. 394–411. Springer, 2024.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Huan Liu, Zichang Tan, Chuangchuang Tan, Yunchao Wei, Jingdong Wang, and Yao Zhao. Forgery-aware adaptive transformer for generalizable synthetic image detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10770–10780, 2024.

RuiPeng Ma, Jinhao Duan, Fei Kong, Xiaoshuang Shi, and Kaidi Xu. Exposing the fake: Effective diffusion-generated images detection. In *The Second Workshop on New Frontiers in Adversarial Machine Learning*, 2023.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do gans leave artificial fingerprints? In *2019 IEEE conference on multimedia information processing and retrieval (MIPR)*, pp. 506–511. IEEE, 2019.

Falko Matern, Christian Riess, and Marc Stamminger. Exploiting visual artifacts to expose deepfakes and face manipulations. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pp. 83–92. IEEE, 2019.

Marco Mistretta, Alberto Baldrati, Lorenzo Agnolucci, Marco Bertini, and Andrew D. Bagdanov. Cross the gap: Exposing the intra-modal misalignment in CLIP via modality inversion. In *The Thirteenth International Conference on Learning Representations*, 2025.

Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16784–16804. PMLR, 17–23 Jul 2022.

Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. Towards universal fake image detectors that generalize across generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24480–24489, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Jonas Ricker, Denis Lukovnikov, and Asja Fischer. Aeroblade: Training-free detection of latent diffusion images using autoencoder reconstruction error. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9130–9140, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Zeyang Sha, Zheng Li, Ning Yu, and Yang Zhang. De-fake: Detection and attribution of fake images generated by text-to-image generation models. In *Proceedings of the 2023 ACM SIGSAC conference on computer and communications security*, pp. 3418–3432, 2023.

Zeyang Sha, Yicong Tan, Mingjie Li, Michael Backes, and Yang Zhang. Zerofake: Zero-shot detection of fake images generated and edited by text-to-image generation models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 4852–4866, 2024.

Chuangchuang Tan, Renshuai Tao, Huan Liu, Guanghua Gu, Baoyuan Wu, Yao Zhao, and Yunchao Wei. C2p-clip: Injecting category common prompt in clip to enhance generalization in deepfake detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 7184–7192, 2025.

Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8695–8704, 2020.

Zhendong Wang, Jianmin Bao, Wengang Zhou, Weilun Wang, Hezhen Hu, Hong Chen, and Houqiang Li. Dire for diffusion-generated image detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22445–22455, 2023.

Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7556–7566, 2019.

Yichi Zhang and Xiaogang Xu. Diffusion noise feature: Accurate and fast generated image detection. *arXiv preprint arXiv:2312.02625*, 2023.

Mingjian Zhu, Hanting Chen, Mouxiao Huang, Wei Li, Hailin Hu, Jie Hu, and Yunhe Wang. Gendet: Towards good generalizations for ai-generated image detection. *arXiv preprint arXiv:2312.08880*, 2023a.

Mingjian Zhu, Hanting Chen, Qiangyu Yan, Xudong Huang, Guanyu Lin, Wei Li, Zhijun Tu, Hailin Hu, Jie Hu, and Yunhe Wang. Genimage: A million-scale benchmark for detecting ai-generated image. *Advances in Neural Information Processing Systems*, 36:77771–77782, 2023b.

A Appendix

A.1 Evaluation of Training Data Size

Tables 18 and 19 report the results obtained for the detection task using a 3-MLP as classifier for each backbone, while varying the number of training samples (selected by prompts). As observed, with a single training prompt (corresponding to 50 real and 50 fake images), the ViT-B/32 and RN50 backbones exhibit the lowest performance, achieving accuracies of 0.665 and 0.674, respectively. Conversely, RN50×16 and RN50×64 exhibit significantly better performance, with accuracies of 0.900 and 0.923, respectively, despite the limited number of training images (100).

Figure 12 shows the ROC curves for all evaluated backbones, when using a a single prompt for the training set. Clear performance differences emerge across CLIP backbones. Lightweight architectures such as ViT-B/32 and RN50 exhibit a slower increase in true positive rate at low false positive rates, indicating limited discriminative capacity in this extremely low-data regime. In contrast, larger backbones, in particular RN50×16 and RN50×64, achieve substantially higher true positive rates even for very small false positive rates, demonstrating stronger robustness and better feature separability.

When increasing to 100 training prompts (10k images in total), all backbones surpass 0.90 of accuracy. Beyond this point, performance stabilizes, with only marginal improvements observed as additional training data are introduced. Notably, the RN50×64 backbone achieves only a 0.071 increase in accuracy and a 0.02 gain in average precision when the number of training prompts is raised from 1 to 800, while consistently remaining the top-performing model across all training set sizes.

Table 18: Detection results in terms of **Acc. / F1**. A 3-MLP classifier is used for all backbones. Bold values indicate the best result for each backbone. Bold and underlined value denotes the best overall result across all backbones and classifiers. Column headers indicate the number of prompts used for training, with the corresponding number of training images given in parentheses. The last column (Δ) shows the performance gain from 1 to 800 prompts.

	1 (100)	**5 (500)**	**10 (1 000)**	**50 (5 000)**	**100 (10 000)**	**300 (30 000)**	**500 (49 950)**	**800 (79 950)**	**Δ (1→800)**
ViT-B/32	0.665 / 0.635	0.833 / 0.832	0.845 / 0.845	0.890 / 0.889	0.915 /0.915	0.937 / 0.937	0.943 / 0.943	**0.951 / 0.951**	0.286 / 0.316
ViT-B/16	0.780 / 0.774	0.852 / 0.851	0.890 /0.890	0.916 / 0.916	0.936 /0.936	0.953 / 0.953	0.958 / 0.958	**0.963 / 0.963**	0.178 / 0.189
ViT-L/14	0.737 / 0.718	0.875 / 0.874	0.908 / 0.908	0.931 / 0.931	0.942 / 0.942	0.956 / 0.956	0.964 / 0.964	**0.969 / 0.969**	0.232 / 0.251
ViT-L/14@336px	0.794 / 0.784	0.933 / 0.933	0.950 / 0.950	0.964 / 0.964	0.969 / 0.969	0.978 / 0.978	0.981 / 0.981	**0.984 / 0.984**	0.190 / 0.200
RN50	0.674 / 0.600	0.876 / 0.876	0.889 / 0.889	0.927 / 0.927	0.942 / 0.942	0.959 / 0.959	0.966 / 0.966	**0.969 / 0.969**	0.295 / 0.369
RN101	0.687 / 0.649	0.828 / 0.828	0.856 / 0.856	0.910 / 0.910	0.926 / 0.926	0.946 / 0.946	0.955 / 0.955	**0.962 / 0.962**	0.275 / 0.313
RN50×4	0.763 / 0.753	0.879 / 0.879	0.905 / 0.905	0.941 / 0.941	0.950 / 0.950	0.964 / 0.964	0.969 / 0.969	**0.974 / 0.974**	0.211 / 0.221
RN50×16	0.900 / 0.899	0.952 / 0.952	0.951 / 0.951	0.967 / 0.967	0.975 / 0.975	0.983 / 0.983	0.986 / 0.986	**0.989 / 0.989**	0.089 / 0.090
RN50×64	0.923 / 0.923	0.969 / 0.969	0.974 / 0.974	0.981 / 0.981	0.986 / 0.986	0.992 / 0.992	0.993 / 0.993	**<u>0.994 / 0.994</u>**	0.071 / 0.071

Table 19: Detection results in terms of **AP / AUC**. A 3-MLP classifier is used for all backbones. Bold values indicate the best result for each backbone. Bold and underlined values denote the best overall result across all backbones and classifiers. Column headers indicate the number of prompts used for training, with the corresponding number of training images given in parentheses. The last column (Δ) shows the performance gain from 1 to 800 prompts.

	1 (100)	**5 (500)**	**10 (1 000)**	**50 (5 000)**	**100 (10 000)**	**300 (30 000)**	**500 (49 950)**	**800 (79 950)**	**Δ (1→800)**
ViT-B/32	0.790 / 0.788	0.899 / 0.913	0.928 / 0.928	0.958 / 0.958	0.973 / 0.973	0.983 / 0.983	0.986 / 0.986	**0.989 / 0.989**	0.083 / 0.088
ViT-B/16	0.906 / 0.901	0.937 / 0.937	0.960 / 0.960	0.977 / 0.975	0.984 / 0.983	0.990 / 0.989	0.992 / 0.992	**0.993 / 0.993**	0.056 / 0.056
ViT-L/14	0.926 / 0.921	0.965 / 0.964	0.971 / 0.970	0.983 / 0.982	0.987 / 0.987	0.992 / 0.992	0.994 / 0.994	**0.995 / 0.995**	0.069 / 0.074
ViT-L/14@336px	0.939 / 0.939	0.983 / 0.984	0.990 / 0.990	0.995 / 0.995	0.996 / 0.996	**0.998** / 0.998	**0.998** / 0.998	**0.998 / 0.999**	0.059 / 0.060
RN50	0.826 / 0.839	0.945 / 0.946	0.955 / 0.957	0.981 / 0.980	0.986 / 0.986	0.992 / 0.992	0.994 / 0.994	**0.995 / 0.995**	0.169 / 0.156
RN101	0.842 / 0.852	0.899 / 0.908	0.938 / 0.937	0.971 / 0.970	0.979 / 0.978	0.987 / 0.987	0.991 / 0.991	**0.993 / 0.993**	0.151 / 0.141
RN50x4	0.875 / 0.888	0.950 / 0.949	0.967 / 0.969	0.986 / 0.986	0.988 / 0.989	0.994 / 0.994	0.995 / 0.995	**0.996 / 0.996**	0.121 / 0.108
RN50x16	0.963 / 0.967	0.990 / 0.991	0.989 / 0.990	0.995 / 0.996	0.997 / 0.997	0.998 / 0.998	**0.999 / 0.999**	**0.999 / 0.999**	0.036 / 0.032
RN50x64	0.985 / 0.985	0.997 / 0.997	0.997 / 0.997	0.998 / 0.998	0.999 / 0.999	**<u>1.000 / 1.000</u>**	**<u>1.000 / 1.000</u>**	**<u>1.000 / 1.000</u>**	0.015 / 0.015

A similar trend is observed for both the attribution (Tables 20 and 21) and the joint detection+attribution (Tables 22 and 23) tasks.

A.2 Preprocessing

An important factor in CLIP-based fake image detectors is the choice of image preprocessing, which can significantly influence the extracted feature representations and, consequently, detection performance. We observed that existing detectors do not always apply the official CLIP preprocessing to images; instead, some methods implement their own custom preprocessing procedures. The key differences between different preprocessing strategies are summarized below:

- **CLIP's official preprocessing**: The image is first resized so that its shortest side becomes N pixels (with $N \in \{224, 288, 336, 384, 448\}$ depending on the backbone), using bicubic interpolation for smooth resizing. Then a (N, N) patch is cropped from the center of the image, ensuring the final spatial resolution matches what the model expects. Then the image is converted in RGB space and normalized[6].
- **De-Fake preprocessing**: The image is directly resized to (224,224) and then it is normalized using (0.5, 0.5, 0.5) as mean and as standard deviation.
- **UnivFD preprocessing**: During training, the image is first resized to the proper size (256); then it applies blur and JPEG compression for data augmentation. After that, the images is random cropped and normalized using the mean and the standard deviation from CLIP's official preprocessing.

[6]The image is normalized with a mean of (0.48145466, 0.4578275, 0.40821073) and a standard deviation of (0.26862954, 0.26130258, 0.27577711).

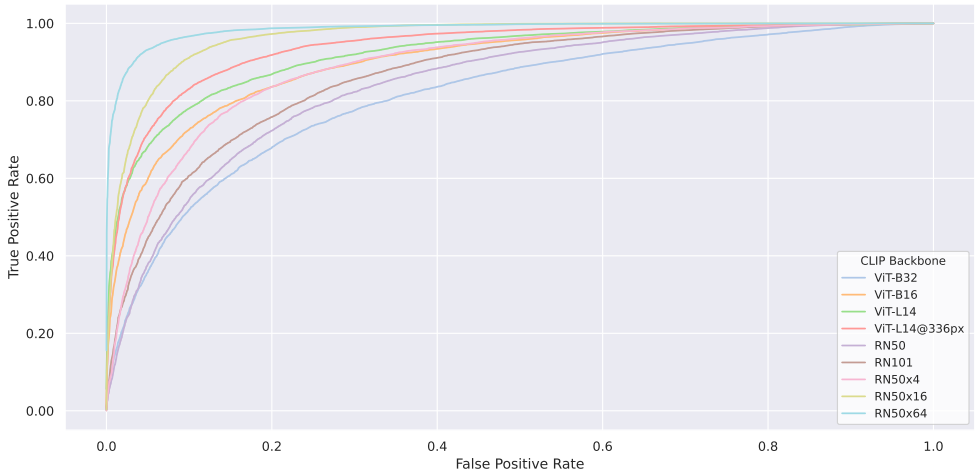

Figure 12: Detection task. ROC curves for all evaluated backbones, using 3-MLP as classifier and a single prompt for the training set. Each curve shows the trade-off between the True Positive Rate and False Positive Rate across varying decision thresholds.

Table 20: Attribution results in terms of **Acc. / F1**. A 3-MLP classifier is used for all backbones. Bold values indicate the best result for each backbone. Bold and underlined value denotes the best overall result across all backbones and classifiers. Column headers indicate the number of prompts used for training, with the corresponding number of training images given in parentheses. The last column (Δ) shows the performance gain from 1 to 800 prompts.

	1 (50)	**5 (250)**	**10 (500)**	**50 (2 500)**	**100 (5 000)**	**300 (15 000)**	**500 (25 000)**	**800 (50 000)**	**Δ (1→800)**
ViT-B/32	0.163 / 0.069	0.353 / 0.281	0.531 / 0.490	0.696/ 0.652	0.731 / 0.684	0.771 / 0.731	0.786 / 0.759	**0.795 / 0.759**	0.632 / 0.690
ViT-B/16	0.217 / 0.105	0.330 / 0.252	0.500 / 0.434	0.718 / 0.676	0.746 / 0.701	0.786 / 0.737	0.794 / 0.753	**0.804 / 0.752**	0.587 / 0.647
ViT-L/14	0.204 / 0.108	0.386 / 0.329	0.590 / 0.529	0.725 / 0.678	0.761 / 0.707	0.789 / 0.752	0.800 / 0.750	**0.809 / 0.775**	0.605 / 0.667
ViT-L/14@336px	0.178 / 0.077	0.351 / 0.288	0.557 / 0.516	0.746 / 0.704	0.772 / 0.728	0.802 / 0.774	0.808 / 0.773	**0.818 / 0.782**	0.640 / 0.705
RN50	0.186 / 0.089	0.371 / 0.315	0.643 / 0.604	0.740 / 0.694	0.762 / 0.718	0.788 / 0.745	0.800 / 0.767	**0.808 / 0.768**	0.622 / 0.679
RN101	0.169 / 0.077	0.290 / 0.226	0.551 / 0.519	0.725 / 0.691	0.751 / 0.700	0.786 / 0.739	0.795 / 0.762	**0.804 / 0.771**	0.635 / 0.694
RN50x4	0.185 / 0.097	0.470 / 0.415	0.630 / 0.582	0.761 / 0.714	0.779 / 0.729	0.800 / 0.769	0.813 / 0.771	**0.821 / 0.782**	0.628 / 0.685
RN50x16	0.218 / 0.124	0.407 / 0.323	0.659 / 0.615	0.775 / 0.741	0.788 / 0.744	0.814 / 0.781	0.818 / 0.782	**0.825 / 0.785**	0.607 / 0.661
RN50x64	**0.234 / 0.138**	**0.483 / 0.430**	**0.695 / 0.652**	**0.793 / 0.752**	**0.806 / 0.781**	**0.820 / 0.769**	**0.828 / 0.808**	**0.833 / 0.809**	0.599 / 0.671

Table 21: Detection results in terms of **AP / AUC**. A 3-MLP classifier is used for all backbones. Bold values indicate the best result for each backbone. Bold and underlined value denotes the best overall result across all backbones and classifiers. Column headers indicate the number of prompts used for training, with the corresponding number of training images given in parentheses. The last column (Δ) shows the performance gain from 1 to 800 prompts.

	1 (50)	**5 (250)**	**10 (500)**	**50 (2 500)**	**100 (5 000)**	**300 (15 000)**	**500 (25 000)**	**800 (50 000)**	**Δ (1→800)**
ViT-B/32	0.319 / 0.739	0.376 / 0.821	0.569 / 0.875	0.748 / 0.942	0.780 / 0.952	0.814 / 0.962	0.824 / 0.965	**0.832 / 0.967**	0.513 / 0.228
ViT-B/16	0.350 / 0.805	0.413 / 0.830	0.546 / 0.879	0.765 / 0.947	0.790 / 0.955	0.822 / 0.965	0.830 / 0.967	**0.838 / 0.969**	0.488 / 0.164
ViT-L/14	0.392 / 0.790	0.437 / 0.829	0.626 / 0.892	0.773 / 0.949	0.802 / 0.959	0.826 / 0.966	0.833 / 0.968	**0.840 / 0.970**	0.448 / 0.180
ViT-L/14@336px	0.401 / 0.824	0.451 / 0.842	0.615 / 0.902	0.795 / 0.957	0.813 / 0.962	0.834 / 0.969	0.840 / 0.970	**0.847 / 0.972**	0.446 / 0.148
RN50	0.385 / 0.789	0.540 / 0.865	0.677 / 0.917	0.782 / 0.953	0.804 / 0.960	0.826 / 0.966	0.832 / 0.968	**0.840 / 0.970**	0.455 / 0.181
RN101	0.389 / 0.788	0.474 / 0.844	0.606 / 0.887	0.769 / 0.949	0.798 / 0.958	0.825 / 0.966	0.831 / 0.968	**0.838 / 0.969**	0.449 / 0.181
RN50x4	0.394 / 0.805	0.450 / 0.843	0.668 / 0.916	0.801 / 0.959	0.813 / 0.963	0.833 / 0.968	0.842 / 0.971	**0.848 / 0.972**	0.454 / 0.167
RN50x16	0.448 / 0.845	0.567 / 0.879	0.700 / 0.928	0.809 / 0.961	0.822 / 0.965	0.841 / 0.970	0.845 / 0.971	**0.850 / 0.973**	0.402 / 0.128
RN50x64	**0.491 / 0.871**	**0.610 / 0.899**	**0.731 / 0.941**	**0.827 / 0.966**	**0.832 / 0.968**	**0.844 / 0.971**	**0.850 / 0.973**	**0.855 / 0.974**	0.364 / 0.103

Table 22: Joint detection and attribution results in terms of **Acc. / F1**. A 3-MLP classifier is used for all backbones. Bold values indicate the best result for each backbone . Bold and underlined value denotes the best overall result across all backbones and classifiers. Column headers indicate the number of prompts used for training, with the corresponding number of training images given in parentheses. The last column (Δ) shows the performance gain from 1 to 800 prompts.

	1 (100)	**5 (500)**	**10 (1 000)**	**50 (5 000)**	**100 (10 000)**	**300 (30 000)**	**500 (49 950)**	**800 (79 950)**	**Δ (1→800)**
ViT-B/32	0.139 / 0.135	0.424 / 0.496	0.551 / 0.612	0.694 / 0.731	0.729 / 0.767	0.774 / 0.805	0.788 / 0.819	**0.797 / 0.830**	0.658 / 0.695
ViT-B/16	0.175 / 0.206	0.427 / 0.561	0.540 / 0.649	0.725 / 0.769	0.749 / 0.784	0.785 / 0.826	0.796 / 0.836	**0.810 / 0.850**	0.635 / 0.644
ViT-L/14	0.180 / 0.229	0.454 / 0.562	0.597 / 0.689	0.732 / 0.777	0.765 / 0.796	0.793 / 0.835	0.807 / 0.847	**0.817 / 0.857**	0.637 / 0.628
ViT-L/14@336px	0.169 / 0.230	0.428 / 0.578	0.623 / 0.741	0.766 / 0.828	0.785 / 0.838	0.814 / 0.864	0.821 / 0.867	**0.830 / 0.875**	0.661 / 0.645
RN50	0.211 / 0.196	0.509 / 0.566	0.640 / 0.679	0.741 / 0.786	0.765 / 0.810	0.798 / 0.839	0.809 / 0.849	**0.818 / 0.860**	0.607 / 0.664
RN101	0.144 / 0.294	0.350 / 0.386	0.545 / 0.613	0.723 / 0.758	0.755 / 0.783	0.789 / 0.817	0.801 / 0.831	**0.812 / 0.853**	0.668 / 0.559
RN50x4	0.206 / 0.208	0.529 / 0.593	0.637 / 0.700	0.763 / 0.808	0.787 / 0.831	0.810 / 0.849	0.824 / 0.864	**0.833 / 0.874**	0.627 / 0.666
RN50x16	0.263 / 0.354	0.617 / 0.731	0.682 / 0.779	0.791 / 0.850	0.804 / 0.848	0.826 / 0.871	0.831 / 0.875	**0.839 / 0.887**	0.576 / 0.533
RN50x64	**0.281 / 0.327**	**0.627 / 0.738**	**0.722 / 0.802**	**0.812 / 0.863**	**0.822 / 0.877**	**0.837 / 0.886**	**0.845 / 0.897**	**<u>0.848 / 0.898</u>**	0.567 / 0.571

Table 23: Joint detection and attribution results in terms of **AP / AUC**. A 3-MLP classifier is used for all backbones. Bold values indicate the best result for each backbone. Bold and underlined value denotes the best overall result across all backbones and classifiers. Column headers indicate the number of prompts used for training, with the corresponding number of training images given in parentheses. The last column (Δ) shows the performance gain from 1 to 800 prompts.

	1 (100)	**5 (500)**	**10 (1 000)**	**50 (5 000)**	**100 (10 000)**	**300 (30 000)**	**500 (49 950)**	**800 (79 950)**	**Δ (1→800)**
ViT-B/32	0.281 / 0.743	0.421 / 0.863	0.566 / 0.906	0.715 / 0.954	0.752 / 0.965	0.799 / 0.975	0.813 / 0.978	**0.824 / 0.980**	0.543 / 0.237
ViT-B/16	0.332 / 0.824	0.470 / 0.879	0.583 / 0.921	0.746 / 0.962	0.769 / 0.968	0.811 / 0.978	0.823 / 0.981	**0.834 / 0.983**	0.502 / 0.159
ViT-L/14	0.370 / 0.798	0.482 / 0.869	0.630 / 0.925	0.761 / 0.968	0.787 / 0.973	0.820 / 0.980	0.833 / 0.983	**0.841 / 0.984**	0.471 / 0.186
ViT-L/14@336px	0.375 / 0.829	0.532 / 0.904	0.672 / 0.956	0.802 / 0.979	0.819 / 0.982	0.841 / 0.985	0.849 / 0.986	**0.856 / 0.987**	0.481 / 0.158
RN50	0.331 / 0.786	0.545 / 0.880	0.656 / 0.928	0.768 / 0.968	0.794 / 0.975	0.826 / 0.982	0.835 / 0.983	**0.843 / 0.985**	0.512 / 0.199
RN101	0.349 / 0.796	0.443 / 0.834	0.572 / 0.907	0.748 / 0.962	0.779 / 0.970	0.815 / 0.979	0.827 / 0.982	**0.837 / 0.984**	0.488 / 0.188
RN50x4	0.364 / 0.812	0.548 / 0.891	0.672 / 0.939	0.795 / 0.974	0.813 / 0.978	0.834 / 0.983	0.847 / 0.985	**0.855 / 0.986**	0.491 / 0.174
RN50x16	0.440 / 0.864	0.661 / 0.948	0.739 / 0.968	0.818 / 0.982	0.834 / 0.984	0.852 / 0.987	0.856 / 0.987	**0.863 / 0.988**	0.423 / 0.124
RN50x64	**0.499 / 0.903**	**0.653 / 0.960**	**0.756 / 0.974**	**0.841 / 0.985**	**0.847 / 0.986**	**0.859 / 0.988**	**0.864 / 0.989**	**<u>0.869 / 0.989</u>**	0.370 / 0.086