UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Contrastive Learning for Detection of AI-Generated Images and Source Attribution

MASTER THESIS

Submitted for the degree of Master by:

Jaime Álvarez Urueña

Madrid, 2025

UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Master Degree in Aprendizaje Automático y Datos Masivos

Contrastive Learning for Detection of AI-Generated Images and Source Attribution

MASTER THESIS

Submitted for the degree of Master by:

Jaime Álvarez Urueña

Under the supervision of:
Dr. David Camacho
Dr. Javier Huertas Tato

Madrid, 2025

"From success, you learn absolutely nothing.
From failure and setbacks, conclusions can be drawn."
— Niki Lauda.

Acknowledgement

This thesis has been partially supported by by the following projects: H2020 TMA-MSCA-DN TUAI project "Towards an Understanding of Artificial Intelligence via a transparent, open and explainable perspective" (HORIZON-MSCA-2023-DN-01-01, Grant agreement nº: 101168344); by project PCI2022-134990-2 (MARTINI) of the CHISTERA IV Cofund 2021 program; by European Comission under IBERIFIER Plus - Iberian Digital Media Observatory (DIGITAL-2023-DEPLOY- 04-EDMO-HUBS 101158511); by EMIF managed by the Calouste Gulbenkian Foundation, in the project MuseAI; and by Comunidad Autonoma de Madrid, CIRMA-CAM Project (TEC-2024/COM-404).

Abstract

Recent breakthroughs and rapid development in the field of artificial intelligence (AI) are significantly transforming various domains, including medicine, education, process optimization, and security, among others. AI is being used to increase productivity, reduce risk, and improve quality of life.

However, alongside its benefits, AI also introduces critical challenges, particularly when misused. One such challenge is the detection of disinformation and hoaxes. Generative AI, capable of producing highly realistic synthetic content, poses a serious threat when used maliciously (potentially harming individuals, organizations, and even nations). In particular, the rise of advanced image generation tools now enables the creation of synthetic images depicting well-known individuals in fabricated scenarios, raising ethical and security concerns. Consequently, reliable mechanisms for detecting AI-generated images and identifying their source have become essential to counteract the spread of misinformation and protect individual reputations.

The tasks of AI-generated image detection and source attribution face two primary challenges: (1) the rapid emergence of new generator models, and (2) the difficulty in retraining detection models due to limited access to large datasets per generator, especially as many of these models are proprietary and lack public documentation. To address these constraints, detectors must generalize well to unseen generators (a setting known as zero-shot detection).

In this work, we propose a zero-shot contrastive learning approach for detecting AI-generated images and attributing them to their source. Our pipeline consists of a CNN (EfficientNetB0) for extracting informative image embeddings, followed by a KNN classifier. We utilize the ForenSynths and GenImage datasets, which contain both real and synthetic images generated by a range of models. Each fake image is labeled with its generator, while real images are labeled as such.

Training is conducted using the Supervised Contrastive Loss (SupConLoss), leveraging class labels to benefit from both contrastive and supervised learning paradigms. A key requirement of our model is strong performance on images from unseen generators. To this end, we performed a series of zero-shot experiments in which certain generators were excluded from the training set. These experiments helped evaluate model generalization and identify the most useful generator types for training.

For AI-generated image detection, results showed that the model trained with only one generator and real images performed poorly on unseen generators (mean AUC of 60.5%), indicating the importance of diverse training data for SupConLoss. In contrast, when trained solely on images from multiple generators (excluding real images), the model achieved a high AUC of 97.8% in distinguishing real from synthetic images. This outcome is valuable, as it shows that is is possible to detect real images from fake ones without any of them in the training phase.

In the source attribution task, model performance varied depending on the generators used during training. The best model achieved an F1-score of 60.3% on zero-shot generators and

82.3% on non-zero-shot generators. Notably, the model achieved 89.3% accuracy in detecting real images (despite not having seen any during training) highlighting the ability to generalize of the learned representations.

The KNN classifier requires a set of base samples to calculate distances. These were selected as subsets from the training set. Experiments evaluating different values for the number of neighbors and base samples revealed that using 11 neighbors and just 50 samples per generator yielded nearly optimal performance, only 3.59% below the baseline using the full dataset (162,000 samples per generator), demonstrating the feasibility of this approach.

Additional experiments explored the effects of alternative loss functions (e.g., Euclidean distance), batch size (with larger sizes improving performance by up to 0.87%), image resolution, embedding size, and classifier architectures.

Visualization and explainability results were also included to further interpret the results. UMAP was used to project the high-dimensional embeddings into a two-dimensional space for cluster visualization. LIME was employed to identify image regions most influential in generating embeddings, providing insight into the features utilized by the contrastive model.

The proposed approach demonstrates that, given a sufficiently diverse set of generators in the training phase, the model is capable of generalizing to images produced by previously unseen generators. Additionally, the trained models can accurately detect real images, despite not having encountered any during training. The method is feasible, as it does not require a large number of samples per generator to perform zero-shot classification effectively. Visual analyses revealed that the models consistently positioned embeddings from different generators near each other in the latent space, regardless of which specific generators were included during training. Furthermore, the LIME models provided valuable insights by highlighting similar image regions across different generators that were key in the embedding extraction process.

Resumen

Los avances recientes y el rápido desarrollo en el campo de la inteligencia artificial (IA) están transformando significativamente diversos ámbitos, incluyendo la medicina, la educación, la optimización de procesos o la seguridad. La IA se está utilizando para aumentar la productividad, reducir riesgos y mejorar la calidad de vida.

Sin embargo, junto con sus beneficios, la IA también introduce desafíos críticos, sobre todo cuando se hace un uso indebido de ella. Uno de estos desafíos es la detección de desinformación y bulos. La IA generativa, capaz de producir contenido sintético altamente realista, representa una amenaza seria cuando se utiliza de manera maliciosa (afectando potencialmente a individuos, organizaciones e incluso países). En particular, el auge de herramientas avanzadas de generación de imágenes permite crear imágenes que representan a personas conocidas en escenarios falsos, lo cual plantea preocupaciones éticas y de seguridad. En consecuencia, es esencial disponer de mecanismos confiables y robustos para detectar imágenes generadas por IA e identificar su fuente, con el fin de contrarrestar la propagación de información errónea y proteger la reputación individual.

Las tareas de detección de imágenes generadas por IA y de atribución de su origen presentan dos retos principales: (1) la rápida aparición de nuevos generadores de imágenes, y (2) la dificultad para reentrenar los modelos de detección debido al acceso limitado a grandes conjuntos de datos por cada generador, especialmente debido a que muchos de estos modelos son privados y carecen de documentación pública. Para abordar estas limitaciones, los detectores deben generalizar bien a generadores no vistos durange la fase de entrenamiento (un escenario conocido como detección «zero-shot»).

En este trabajo, se propone un enfoque de aprendizaje contrastivo zero-shot para la detección de imágenes generadas por IA y su atribución de origen. El flujo de procesamiento consta de una CNN (EfficientNetB0) para extraer embeddings informativos de las imágenes, seguida de un clasificador KNN. Se utilizan los conjuntos de datos ForenSynths y GenImage, que contienen tanto imágenes reales como sintéticas generadas por una amplia variedad de modelos generativos. Cada imagen falsa está etiquetada con su generador, mientras que las imágenes reales se etiquetan como tales.

El entrenamiento se lleva a cabo utilizando la función de pérdida SupConLoss, aprovechando las etiquetas de clase para beneficiarse tanto de los paradigmas de aprendizaje contrastivo como supervisado. Un requisito clave de nuestro modelo es el buen desempeño en imágenes de generadores no vistos en la fase de entrenamiento. Con este fin, realizamos una serie de experimentos zero-shot en los que ciertos generadores se excluyeron del conjunto de entrenamiento. Estos experimentos ayudaron a evaluar la generalización del modelo e identificar los tipos de generadores más útiles para el entrenamiento.

Para la detección de imágenes generadas por IA, los resultados mostraron que el modelo entrenado con un único generador y con imágenes reales tuvo un desempeño subóptimo en generadores no vistos (AUC media del 60,5%), lo que indica la importancia de disponer de datos de entrenamiento diversos para la función de pérdida SupConLoss. En contraste,

cuando se entrenó exclusivamente con imágenes sintéticas de múltiples generadores (excluyendo imágenes reales), el modelo alcanzó una AUC del 97,8% . Este resultado es valioso, pues demuestra que es posible detectar imágenes reales de las falsas sin haber entrenado con ninguna imagen real.

En la tarea de atribución de origen, el rendimiento del modelo varió según los generadores empleados durante el entrenamiento. El mejor modelo obtuvo un F1-score del 60,3% en generadores zero-shot y del 82,3% en generadores no zero-shot. Cabe destacar que el modelo alcanzó una tasa de acierto del 89,3% en la detección de imágenes reales (a pesar de no haber visto ninguna durante el entrenamiento), lo que subraya la capacidad de generalización de las representaciones aprendidas.

El clasificador KNN requiere un conjunto de muestras base para calcular distancias. Estas se seleccionaron como subconjuntos del conjunto de entrenamiento. Los experimentos que evaluaron distintos valores para el número de vecinos óptimos en el algoritmo KNN y el número de muestras base revelaron que usar 11 vecinos y solo 50 muestras por generador ofrecía un rendimiento casi óptimo, apenas un 3,59% inferior al óptimo, donde se utiliza el conjunto completo (162.000 muestras por generador), demostrando la factibilidad del método propuesto.

Experimentos adicionales exploraron los efectos de funciones de pérdida alternativas (por ejemplo, Distancia Euclídea), tamaño de batch (tamaños mayores mejoraron el rendimiento hasta en un 0,87%), resolución de imágenes, dimensión de los embeddings y arquitecturas de clasificadores.

También se incluyeron resultados de visualización y explicabilidad para interpretar mejor los resultados. Se utilizó UMAP para proyectar los embeddings de alta dimensión en un espacio bidimensional y visualizar los clústeres. Se empleó LIME para identificar las regiones de la imagen más influyentes en la generación de los embeddings, proporcionando información sobre las características que utiliza el modelo contrastivo para producir sus resultados.

El enfoque propuesto demuestra que, dado un conjunto suficientemente diverso de generadores en la fase de entrenamiento, el modelo puede generalizar a imágenes producidas por generadores previamente no vistos. Además, los modelos entrenados pueden detectar con precisión imágenes reales, pese a no haber sido entrenado con ninguna de ellas. El método es factible, ya que no requiere un gran número de muestras por generador para realizar una clasificación zero-shot de manera eficaz. Los análisis visuales revelaron que los modelos colocan consistentemente los embeddings de diferentes generadores cerca unos de otros en el espacio latente, independientemente de qué generadores específicos se incluyeron durante el entrenamiento. Asimismo, los modelos LIME proporcionaron valiosa información al resaltar regiones de imagen similares en diferentes generadores que fueron clave en el proceso de extracción de embeddings.

Table of Contents

List of Figures

List of Tables

Abbreviations and acronyms

UPM	Universidad Politécnica de Madrid
AI	Artificial Intelligence
CNN	Convolutional Neural Network
ZED	Zero-shot Entropy-based Detector
GAN	Generative Adversarial Network
AUC	Area Under the Curve
DDIM	Denoising Diffusion Implicit Model
VLM	Vision Language Models
NLP	Natural Language Processing
LM	Language Models
MLM	Masked Language Modeling
MSE	Mean Squared Error
DDPM	Denoising Diffusion Probabilistic Model
VQ	Vector Quantization
LDM	Latent Diffusion Model
VAE	Variational Autoencoder
CLIP	Contrastive Language Image Pretraining
CRN	Cascaded Refinement Network
SITD	Seeing In The Dark
SAN	Second order Attention Network
IMLE	Implicit Maximum Likelihood Estimation
ES	Experiment Setting
TP	True Positives
FP	False Positives

TN True Negatives

FN False Negatives

TPR True Positives Rate

FPR False Positives Rate

ROC Receiver Operating Characteristic

GPU Graphics Processing Unit

VRAM Video Random Access Memory

DIRE Diffusion Reconstruction Error

FAM Forgery Awareness Module

LoRA Low-Rank Adaptation

NPR Neighboring Pixel Relationships

LGrad Learning on Gradients

SotA State of the Art

UMAP Uniform Manifold Approximation and Projection

LIME Local Interpretable Model-agnostic Explanations

Chapter 1

Introduction

1.1 Context and motivation

Generative Artificial Intelligence (Generative AI) has introduced numerous advantages, not only for AI researchers, but also across various fields such as science [1], business [2], medicine [3] or education [4], among others. For example, in oncology, generative AI is integrated with medical imaging to enhance the early detection of cancer, significantly reducing mortality rates and improving the quality of life for affected patients [5]. In the field of vaccine development, generative AI plays a crucial role in deciphering protein sequences, facilitating the creation and enhancement of vaccines [6].

However, despite its immense benefits and transformative potential, generative AI also presents significant challenges and risks [7]. For instance, AI-powered chatbots can be misused by students to complete academic assignments dishonestly [8]. Additionally, generative image models have the potential to create misleading or controversial images that could influence political decisions [9]. Similarly, AI-generated audio models pose risks related to copyright infringement and unauthorized content creation [10].

This project focuses on the detection of AI-generated images, which poses a significant threat as one of the most potentially harmful forms of fake news [11]. While text generation can lead to issues such as copyright infringement or misinformation, the creation of AI-generated images is particularly concerning. This worsens when images depict well-known individuals in misleading contexts, suggesting they have engaged in certain actions that may not be true.

Researchers and companies responsible for developing these models work to minimize these risks from the design phase [12, 13]. However, this is an almost impossible task, given the vast number of harmful AI-generated images that could arise, and the ease with which users can circumvent safeguards designed to prevent the creation of unethical content [14]. Additionally, the process of teaching a model to avoid generating harmful images that could negatively impact society is far from straightforward [15].

The landscape of AI-driven image generation has evolved rapidly in recent years, with the number of models capable of producing highly realistic images growing dramatically [16]. As

a result, a model trained on current image generators may not remain effective in the near future, as new and more sophisticated image-generation techniques will likely emerge [17]. Consequently, it is imperative to develop models that can reliably distinguish between real and AI-generated images not only with today's cutting-edge generators but also with future and unknown ones.

In this study, a comprehensive research is conducted to evaluate various approaches for effectively distinguishing AI-generated images from real ones. Additionally, we developed a model trained using contrastive learning techniques on a set of AI-generated images from different generators, which is capable of identifying the specific generator responsible for an image, even if that generator was not included in the training set. To enhance the interpretability of the model, explainability models such as LIME were incorporated, which offers insights into the decision-making process. Furthermore, visualizations are provided to facilitate the understanding of the results and support the analysis.

1.2 Problem statement

Detecting AI-generated images is a complex and increasingly challenging task. In earlier stages of development, image generation models often produced outputs with evident artifacts and inconsistencies which made distinguishing synthetic images from real ones relatively straightforward. However, the rapid advancement of generative models has significantly narrowed the perceptual gap between real and AI-generated content. Modern image generators can now produce highly realistic images across a broad range of domains. Figure 1.1 illustrates this challenge by presenting a real image of a bird (left) alongside an AI-generated counterpart (right). While the image on the right still exhibits subtle imperfections such as irregularities in the bird's outline or unnatural rock textures, these cues are becoming increasingly subtle as the technology advances. Consequently, the task of reliably identifying AI-generated images continues to grow more difficult.

Figure 1.1: AI-generated image detection problem.

Figure 1.2 presents a set of four images, 3 generated by AI and one real. Without any contextual information, distinguishing the real image from the synthetic ones becomes a

non-trivial task. Each image contains realistic elements that can easily mislead even a trained observer. For instance, the image featuring a shark displays highly convincing visual details, such as accurate light reflections on the shark's body and realistic shadows cast on the sand. Similarly, the second image includes natural features like fallen leaves contributing to a lifelike appearance. The third image depicts a bird with no obvious artifacts or visual inconsistencies. Lastly, the fourth image, which shows a match being lit, is the only real photograph among the four. Notably, it does not exhibit any distinctive qualities that clearly separate it from the synthetic images, underscoring the sophistication of modern generative models and the growing difficulty in reliably identifying AI-generated visuals.

Figure 1.2: Examples of real and AI-generated images.

Another significant challenge in detecting AI-generated images lies in the wide variety of domains in which generative models can now produce highly realistic content. While Figure 1.1 focuses on bird content, Figure 1.2 showcases images from diverse domains, highlighting the versatility of modern image generators. If generative models were limited to producing images within a narrow set of domains, the detection task would be considerably more manageable. In such cases, domain-specific or ad hoc techniques could be developed to extract tailored features relevant to each context. However, given that AI-generated images can span virtually any domain (ranging from wildlife and landscapes to objects and human activities) these specialized strategies become impractical. Features that are effective in one domain may be irrelevant or even misleading in another, thereby complicating the design of robust and generalizable detection methods.

The challenge becomes even more complex when the goal extends beyond simply distinguishing real from AI-generated images to also identifying which specific image generator produced a given synthetic image. Figure 1.3 presents eight AI-generated images, each created by a different generative model. Training an AI system to accurately attribute an image to its source generator requires the model to move beyond semantic understanding. That is, while multiple generators may produce images with similar semantic content (e.g., portraits of people), the model must instead learn to recognize subtle visual patterns, artifacts, or stylistic signatures unique to each generator. This task is further complicated by the wide range of domains in which these generators operate. For instance, two generators may produce images with entirely different visual styles when applied to the same semantic domain, or conversely, may generate visually similar outputs across distinct domains. In Figure 1.3, the 8 images span a variety of semantic categories, making it essential for the classification model to identify generator-specific fingerprints rather than relying on domain-related cues. Successfully solving this problem demands robust feature extraction techniques capable of capturing generator-level patterns, independent of image content.

Modern image generators continue to improve rapidly, producing images with increasingly fewer artifacts, lower noise levels, and fewer detectable "errors". As a result, if a detection model fails to identify consistent and distinctive patterns, it may not only misclassify the generator responsible for producing the image but also incorrectly label synthetic content as real. This presents a significant challenge for both detection and source attribution tasks.

Additionally, the fast-paced development of new image generation models introduces another layer of complexity. In practical terms, it is not feasible to retrain detection systems each time a new generator is released. Consequently, it is critical for detection models to generalize effectively to generators that were not encountered during the training phase. This requirement is particularly pressing given that many state-of-the-art generative models are proprietary or closed-source, making it difficult to obtain a sufficient number of training examples. The limited availability of images per generator further constrains the training process, hindering the model's ability to learn robust and generalizable features. Overcoming these challenges requires the development of detection methods that are both data-efficient and resilient to the rapid evolution of generative technologies.

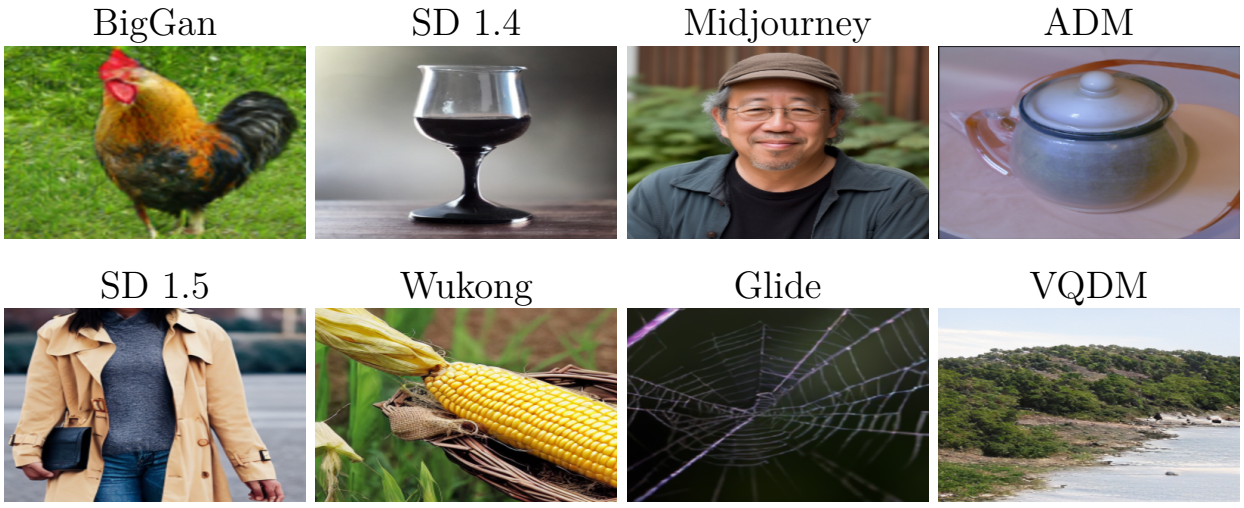


Figure 1.3: Examples of AI-generated images by different generators.

1.3 Objectives

1.3.1 General objective

This project aims to develop a series of models capable of distinguishing between AI-generated and real images. To achieve this objective, multiple models were trained using contrastive learning approaches. The test dataset used across all experiments consisted of images from eight distinct AI generators, each contributing 6,000 synthetic images (except one generator which contributed 8,000), alongside 6,000 real images. A series of experiments were conducted to evaluate the models' generalization capabilities with different training datasets. Specifically, the models were trained on images produced by a subset of AI image generators and then tested on images from previously unseen generators.

1.3.2 Specific objectives

Apart from the main objective of this project which consists in detecting real versus AI-generated images, a set of secondary objectives was established to enrich the scope of the research:

- **Distinguish images from different generators.** Each generator has its own distinctive style and visual patterns, making it potentially feasible to determine which generator produced a given image. Achieving this goal could contribute to and facilitate the completion of the primary task.
- **Use of disjoint sets of generators in training and testing phases.** The number and diversity of image generation techniques are rapidly increasing, making it essential to develop models capable of handling images produced by previously unseen generator architectures. To address this challenge, a series of experiments was conducted to explore effective strategies and methodologies for enabling such generalization.
- **Visualization of results.** Visualizing the obtained results is a well-established practice to enhance interpretability and support the discussion of findings. Accordingly, a series of plots are included in this document to illustrate and analyze the performance of the models.
- **Explainability.** Closely related to the previous point, explainability techniques are employed to provide deeper insights into the model's decision-making process. In particular, the LIME framework is used to identify the most influential image regions contributing to the classification of a given image.

1.4 Preliminaries

The proposed approach to tackle both AI-generated image detection and source attribution problem leverages EfficientNet architectures, which are based on Convolutional Neural Networks (CNNs), for image processing. Additionally, contrastive learning is employed to enable the model to effectively learn and distinguish the unique patterns and characteristics inherent to each image generator.

1.4.1 EfficientNet

EfficientNets represent state-of-the-art architectures for image classification, offering two significant advantages: they contain approximately seven times fewer parameters than conventional CNNs and achieve up to six times faster inference speeds [18]. EfficientNets effectively address the challenge of scaling convolutional neural networks by uniformly increasing their three key dimensions; depth (number of layers), width (number of features per layer), and resolution (input image size).

Traditionally, scaling CNNs involved increasing only one or two of these dimensions, which, as demonstrated in [18], limits the network's learning capacity and leads to suboptimal performance. The EfficientNet approach of compound scaling ensures balanced growth across

all dimensions, resulting in improved accuracy and efficiency.

- Increasing only the depth of the network can lead to the degradation problem. This issue manifests as higher training error due to optimization difficulties. As the number of layers increases, less gradient information is backpropagated to the initial layers (known as the vanishing gradient problem), preventing effective optimization of these early layers and hindering overall training. The degradation problem explains why, in some cases, smaller and simpler networks outperform deeper, more complex ones. This issue is significantly alleviated by the use of residual connections, which link non-consecutive layers and allow better gradient flow during backpropagation, improving updating the weights of the earlier layers.
- Disproportionately increasing the width of the network impairs generalization. A wider network drastically increases the number of parameters, increasing the risk of overfitting. Since each filter in a convolutional layer generates a feature map, adding many filters may cause the network to learn redundant or irrelevant features. This not only reduces generalization performance but also increases computational and memory costs due to the enlarged parameter space.
- Increasing the input resolution introduces two major challenges. First, it may cause underfitting if the model is too simple to capture the added complexity of higher-resolution inputs. Second, it substantially raises memory usage and computational complexity required to process the larger images, potentially limiting training efficiency.

To build larger CNNs that enhance the network's capabilities while avoiding the aforementioned issues, it is necessary to scale all three dimensions proportionally. In [18], three scaling coefficients are introduced (α, β and γ), corresponding to the depth, width, and resolution of the network, respectively. These parameters are used to uniformly balance the scaling of the CNN dimensions. They must satisfy both the constraint in Equation 1.1 and the condition of being greater than or equal to 1.

$$\alpha * \beta^2 * \gamma^2 \approx 2 \tag{1.1}$$

In [18], the recommended values for the scaling coefficients α, β, and γ are 1.2, 1.1, and 1.15, respectively. Based on these parameters, the user defines a compound scaling factor φ, which controls the overall scale of the network. If the goal is to use 2^N times more resources, then φ is set equal to N, and the final scaled dimensions are given by Equation 1.2.

$$\begin{aligned} d &= \alpha^{\varphi} \\ w &= \beta^{\varphi} \\ r &= \gamma^{\varphi} \end{aligned} \tag{1.2}$$

where d, w, and r represent the actual scaling factors for the depth, width, and resolution of the CNN, respectively.

There are eight different EfficientNet architectures, each varying in these dimensions. The choice of which model to use depends on the resolution of the input images. Table 1.1 shows the optimal EfficientNet model for different input image resolutions.

In this project, EfficientNetB0 was chosen as the base architecture, despite the input images

EfficientNet model	Input resolution (pixels)
EfficientNetB0	224×224
EfficientNetB1	240×240
EfficientNetB2	260×260
EfficientNetB3	300×300
EfficientNetB4	380×380
EfficientNetB5	456×456
EfficientNetB6	528×528
EfficientNetB7	600×600

Table 1.1: Optimal images' input sizes for each EfficientNet architecture.

having a resolution of 256×256 (see Section 3.3.1). This decision was driven by strict VRAM constraints (see Section 3.5.1). As discussed in Section 3.1, models trained with contrastive learning techniques tend to achieve better performance when trained with large batch sizes. Accordingly, in this project, batch size was prioritized over selecting the EfficientNet model strictly based on input image resolution.

1.4.2 Contrastive learning

Contrastive Learning is a type of self-supervised learning that focuses on learning representations (embeddings) of the input data by comparing pairs of data samples. Unlike traditional supervised learning, which relies on labeled data, contrastive learning leverages the inherent structure within the data itself to learn to distinguish between different classes.

The main principle behind contrastive learning is to pull together the embeddings of similar (positive) pairs and push apart those of dissimilar (negative) pairs in the latent space of the model. In the context of image data, positive pairs are typically generated by applying different augmentations (i.e., cropping, rotation) to the same image, while negative pairs consist of distinct images. Nonetheless, in this project no augmentations were applied, instead, similar pairs consist of images generated by the same image generator. By optimizing a contrastive loss function, such as the InfoNCE loss, the model learns to encode meaningful and discriminative features that explains each class present in the training set.

This approach enables the models to learn robust and generalizable features without requiring explicit annotations, making it especially valuable when labeled data is scarce or expensive to obtain. Contrastive learning has demonstrated remarkable success in various computer vision tasks, including image classification, object detection, and more recently, in tasks like AI-generated image detection and source attribution. However, since labels are available in this context, a supervised contrastive learning approach can be employed. This method combines the strengths of both contrastive and supervised learning by leveraging the class labels of each image sample during the contrastive process. Instead of treating all other images as negatives (as in traditional contrastive learning), samples belonging to the same class are treated as additional positives. Moreover, further information about other classes is provided, enhancing the final results. This provides the model with richer semantic information during training, allowing it to learn more structured and discriminative feature representations. As

a result, supervised contrastive learning typically achieves better performance, especially in tasks that require fine-grained class distinctions.

Moreover, recent studies show that contrastive learning benefits significantly from larger batch sizes and longer training times, as these increase the number of negative samples the model can compare against, leading to richer and more discriminative representations (for this reason, in this project, a larger batch size was prioritized over selecting a higher-capacity EfficientNet model).

1.5 Organisation and structure

This report of the conducted research presents the next structure:

Chapter 1. Introduction: Definition of the problem, its context, the motivation to solve it, preliminary information, as well as the objectives of the project.

Chapter 2. Related work: Research conducted by other authors to tackle the presented problems, including approaches, methods, techniques and models used in this domain. Problems and solutions given by other authors are also included in this chapter.

Chapter 3. Materials and Methods: This chapter provides a detailed description of the data used and its preprocessing throughout the research, along with the methodologies, machine learning techniques, architectures, and models employed. Additionally, the conducted experiments and the rationale behind their design are thoroughly explained. Metrics used across the whole research are also included, as well as visualization techniques and explicability models used.

Chapter 4. Results: Exposition of the results obtained of all conducted experiments are presented in this chapter. It includes detailed findings from ablation studies, as well as the outcomes of a data feasibility analysis

Chapter 5. Discussion: This chapter provides an analysis of the results presented in the previous chapter, offering a thorough interpretation of the findings. It also includes an in-depth discussion of the explainability and visualization outcomes, highlighting their significance and implications in the context of the study.

Chapter 5. Conclussions: Conclussions of the conducted research, along with the findings and the contributions made in the detection of AI-generated images domain. Future work is also included in this section.

Chapter 2

Related work

Detection of AI-generated images has recently emerged as a critical area of research due to its far-reaching implications and consequences. Fake images can significantly impact societies worldwide, not only by fueling misinformation and spreading fake news, but also through the creation of images that feature real individuals in fake contexts. This raises serious legal and ethical concerns, as such images can depict real people doing or saying things they never did, potentially damaging reputations.

In this context, several domains are currently being explored by researchers and authors. One critical area of investigation is the differentiation between AI-generated images and real photographs, which plays a vital role in combating disinformation and fake news. This task can extend beyond simply identifying whether an image is real or synthetic to also include attributing the image to a specific generative model. Such capabilities are valuable not only for detecting emerging, previously unknown generators, but also for identifying when existing detection models become outdated and require retraining with images from newer generation techniques.

Modern generative models possess the capability to perform image inpainting, seamlessly synthesizing specific regions within real images. As a result, it becomes increasingly important not only to determine whether an image has been AI-generated, but also to localize the altered or synthetic regions within it. This level of granularity is crucial for accurately assessing the authenticity of visual content and for developing more robust detection systems.

2.1 Datasets

For the AI-generated image detection, there are some open-sourced datasets that can be used. GenImage [19] consists of images from 8 different generators (ADM [20], Stable diffusion 1.4 [21], Stable diffusion 1.5 [21], VQDM [22], Midjourney [23], Glide [24], BigGan [25] and Wukong [26], each with 162,000 samples for training and 6,000 for testing (except Stable Diffusion 1.5 with 8,000 samples). For each generator, over 162,000 real images for training and 6,000 for testing from ImageNet dataset [27] are also included in this dataset (except Stable Diffusion 1.5 with 8,000 samples). This dataset can be utilized not only for distinguishing

between real and synthetic images, but also for the task of source attribution, as each synthetic image is labeled with the specific generative model that created it.

Another dataset that can be used for both the AI-generated image dectection and source attribution is the DiffusionForensics dataset [28]. It is organized into three distinct domains:

- **Bedroom domain:** This domain contains synthetic bedroom images generated using eight diffusion models: four unconditional models (ADM, DDPM [29], iDDPM [30], and PNDM [31]) and four text-to-image models (LDM [21], Stable Diffusion 1, Stable Diffusion 2 , and VQDM). Each model contributes 40,000 training images, 1,000 validation images, and 1,000 test images. Additionally, three more text-to-image models (IF [32], DALLE-2 [33], and Midjourney) are used to generate further test samples, contributing 1,000, 500, and 100 images, respectively. For comparison, 42,000 real bedroom images (split into 40,000 for training, 1,000 for validation, and 1,000 for testing) are sourced from the LSUN-Bedroom dataset [34].
- **ImageNet Domain:** This domain expands beyond bedroom scenes to include a broader set of classes derived from the ImageNet dataset. From this, 40,000 training images, 1,000 validation images, and 1,000 test images were collected. Two diffusion models are used to generate synthetic data in this domain: ADM (now used as a conditional model) and Stable Diffusion 1 (text-to-image). Each model produces 50,000 synthetic images from diverse ImageNet categories, with 40,000, 5,000 and 5,000 images for training, validation, and testing, respectively.
- **Face Domain:** This domain focuses on human faces. A total of 50,000 real face images are obtained from the CelebA-HQ dataset [35] (40,000 for training, 5,000 for validation, and 5,000 for testing). Synthetic counterparts are generated using Stable Diffusion 2, following the same train-validation-test distribution. Additional test images are collected from three other models: IF (1,000 images), DALLE-2 (500 images) and Midjourney (100 images).

Chamaleon dataset [36] is a dataset that attempts to depict realistic images. To gather all images, the authors used user-created AI-generated images from well-known AI-painting communities, which use commercial APIs (Midjourney, DALLE-3 or Stable diffusion). To create the prompts for these models, they used GPT-4 [37] to generate diverse keywords. The images were not labeled by their original generators; hence, this dataset cannot be used for the source attribution problem. A filtering process was also applied to retain only images that depict one of four distinct categories: humans, animals, objects, or scenes. Any images that did not fall within these predefined categories were excluded from the dataset. On the other hand, Real images were acquired through public platforms using the same prompts used to create the synthetic images. The same filtering criteria applied to AI-generated images was applied to real images. In the end, they managed to gather 150,000 synthetic images and 20,000 real images.

For the experiments conducted in this project, the GenImage dataset was the selected one. This choice was made primarily because the DiffusionForensics and Chamaleon dataset contains significantly fewer images per generator compared to GenImage. While Diffusion-Forensics includes images corresponding to ImageNet classes (offering considerable diversity)

in one of its domains, these were generated using only two models, both based on diffusion architectures. Chameleon employs three models to generate images, organizing them into four abstract categories (humans, animals, objects, and scenes). Besides, it does not label images according to the specific model that generated them, making it unsuitable for training a source attribution model. In contrast, GenImage features images produced by eight different models, including both diffusion models and GANs. Additionally, the remaining domains of images in DiffusionForensics dataset are somewhat biased, focusing primarily on human faces and bedroom scenes. GenImage not only offers a more balanced and diverse range of content but also provides over four times more images per generator, including outputs from GAN-based models, making it more suitable for comprehensive analysis than DiffusionForensics, and more than 10 times more images than Chamaleon. As the aim of this project is to provide a model able to tackle images created from future generators, it is a must to train it with all types of architectures and models in order to avoid its deprecation in the short term.

2.2 Detection of AI-generated images

2.2.1 Contrastive learning

Contrastive learning is a widely used technique for training AI models across various domains, including the detection of AI-generated images. Its primary advantage over supervised learning lies in its ability to generalize to previously unseen classes during inference [38]. Furthermore, contrastive learning has demonstrated superior performance across a broad range of tasks and datasets [39]. A zero shot learning method to solve the AI-generated images problem can be performed [40, 41]. In [40] authors did not use any fake image in the training set, using a zero-shot entropy-based detector (ZED). At inference time, statistics are extracted from the image, and it is measured how different those statistics are from the ones of real images. They achieve state-of-the-art metrics not needing any synthetic image in the training set. On the other hand, in [41], authors used CLIP [42] as image encoder, matching the metrics of other state-of-the-art approaches. In that paper, images generated from 18 distinct generators with different architectures were used (GANs, diffusion models and commercial models with disclosed architectures). Using a zero-shot approximation with CLIP it is claimed that it is not necessary to use huge number of images to train the models (a total of 32,000 images were used). With this reduced number of samples they match state-of-the-art metrics, and implementing data augmentation they managed to improve the Area Under the Curve (AUC) metric by 6%. CLIP is a model that translates both images and text into the same latent space, hence images and captions that have the same semantic meaning would theoretically be close in the latent space. Nevertheless, CLIP was trained exclusively on real images, with no synthetic images included during the training phase. This presents a significant limitation, particularly for the task of classifying images based on their generative source, as all such images are synthetic and thus fall outside the distribution on which CLIP was originally trained on. Moreover, CLIP separates the embeddings of the images based on their semantic meaning. If the same prompt is given to two distinct generators, their semantic meaning would be the same, thus their embeddings in the latent space of CLIP would be similar, making it highly challenging to classify them correctly. For that reason, in this

project, an ad hoc model will be trained with both real and synthetic images, so that its objective is to distinguish image embeddings based on their generative origin, regardless of their semantic content. Unlike CLIP, which organizes images in the latent space according to their semantic meaning, this approach aims to separate them by the generator that produced them. Another major challenge is the constrained capacity to generate sufficient data. While some models are open-source and allow the generation of numerous images using diverse prompts, others are proprietary, resulting in fewer publicly accessible samples. Consequently, developing methods that perform well on low-sample datasets is essential. In [43], the authors proposed a method based on a LoRA-based Forgery Awareness Module, which achieved a 14.55% improvement in classification accuracy while using only 0.56% of the training samples compared to other approaches. In [44], the authors introduced a novel approach that incorporates a different contrastive learning method. Their method involved perturbing images using Denoising Diffusion Implicit Models (DDIM), followed by reconstructing and classifying them. This strategy is motivated by findings demonstrating that synthetic images exhibit greater robustness than real images when subjected to DDIM-based perturbation and reconstruction [45]. Such robustness facilitates more reliable classification, achieving superior performance metrics compared to alternative methods, while also requiring fewer training samples.

2.2.2 Supervised learning

Supervised learning is the traditional approach used for any classification problem, including the detection of AI-generated images. It consists in performing a binary classification between AI-generated images and real images [36, 46, 47, 48]. Researchers in [46] conducted a series of experiments testing their architecture in GenImage and DiffusionForensics datasets. Their experiments consisted in training their model with only images from one generator (Stable Diffusion v1.4) and testing it with all of them. The approach followed in this paper is based in extracting artifact features from images and performing a binary classification (real vs. fake). Artifact features are visual or statistical anomalies unintentionally introduced during the image generation process by generators (mistakes when the image was created). The extraction of this features was followed by many other authors, as generators (specially GANs) introduce lots of anomalies when creating images [49, 50]. On the other hand, the authors in [36] leveraged CLIP to extract high-level semantic features from images. To further enhance these features, they applied patch-wise feature extraction, which captures local inconsistencies at the patch level. In addition to patch-wise methods, pixel-wise feature extraction is also commonly used in this domain [51, 52, 53]. While patch-wise feature extraction aims to identify inconsistencies that may indicate AI-generated content, it differs from artifact-based techniques by focusing on small patches or pixel neighborhoods rather than global image artifacts. However, both patch-wise and artifact-based methods overlook the semantic context of each patch or pixel, relying solely on visual irregularities. This limitation can lead to misclassification, particularly when AI-generated images exhibit no detectable inconsistencies but lack semantic coherence. To overcome this issue, the Global-Local Feature Fusion technique was used, which combines both semantic and local features, obtaining great results, especially in distinguishing real human faces from synthetic ones[54, 55, 56, 57, 58]. Global-Local feature Fusion uses both semantic features of the image (layout, symmetry, general style) and local features (texture

or pixel-level noise), hence images that are technically correct but not semantically can be correctly classified as AI-generated. This technique is widely used in other domains, such as vehicle detection [59], semantic segmentation [60] or cancer detection [61].

In this domain, all authors evaluated their models using a series of binary classification tasks, each corresponding to a specific synthetic image generator. For each task, a set of test images was created, comprising real images and images generated by a particular model. These images were then classified as either real or AI-generated. To assess the overall performance of the model, the classification results were aggregated by computing the mean accuracy across all tasks.

The approach of all previous papers enables to use their trained models with unseen generators during the training phase, however, they perform a binary classification (AI-generated or real image), hence the detection of generators of synthetic images was not addressed. The objective of this project is not only to detect whether an image is AI-generated, but also to detect the source of each image.

2.3 Source Attribution problem

The problem of attributing images to their specific source models is a more complex task than classifying real and fake images [62, 63, 64]. Image generators' architectures tend to have specific features or fingerprints that can be used to distinguish them, such as GANs [65, 66, 67] or diffusion models [68, 69, 70]. For instance, in [71], the authors disentangle the GANs' fingerprints from the semantically relevant features within the latent space through a semantic eliminator whose inputs are images, and its output are GAN's fingerprints. These fingerprints can be effectively used to determine whether an image was generated by a diffusion model or a GAN. Moreover, if model-specific fingerprints can be reliably identified, this approach could enable more fine-grained source attribution, distinguishing between different diffusion models or GAN architectures. Among the possible techniques to perform fake image attribution, Vision Language Models (VLMs) are becoming popular, due to the combination of vision and language, which obtains great results [72, 73]. Authors in [74] use a VLM to ask the model if an image is AI-generated or real, addressing the problem of detecting AI-generated images. On the other hand, it is also possible to ask the model whether an image was created by a specific model, addressing the source-attribution problem. This method, although highly effective, could be costly computationally.

Another task related to this context is inpainting detection [75, 76, 77, 78, 79]. Inpainting involves removing a specific region of an image and reconstructing that missing area using a generative model (typically GANs [80, 81, 82], autoencoders [83, 84], or diffusion models [85, 86, 87]). While this technique can be beneficial for image enhancement tools, such as removing unwanted objects [88], it also carries the potential for misuse.

Chapter 3

Materials and methods

3.1 Contrastive learning

Contrastive learning is gaining momentum lately, as it is a widely used technique in the machine learning domain. It can be used in both the supervised domain (i.e., classification, regression) and unsupervised domain (i.e., feature extraction for clustering algorithms). The reason behind the wide range of domains in which contrastive learning can be applied is the fact that it is a self-supervised method. Self-supervised methods do not need as much data as other machine learning paradigms, as models trained with self-supervised methods use their input as ground-truth for gradient computation, hence the expensive process of data labeling is circumvented. Models trained with contrastive learning learn to create meaningful representations from the data itself by comparing samples rather than predicting labels to each of them as in supervised learning. The number of possible combinations of samples to be compared is huge, even with low sampled datasets; hence, as much data is not needed.

Self-supervised learning is widely applied in areas such as Natural Language Processing (NLP), image classification, image segmentation, etc. In the case of training Language Models (LMs), a common approach is Masked Language Modeling (MLM). In MLM, a full sentence is provided as input to the model but with some tokens masked (typically replaced with a special token or set as padding). The LM is then tasked with predicting the original values of the masked tokens, and the loss is computed by comparing the model's predictions against the true tokens masked (ground truth). This technique allows the same sentence to be reused multiple times by masking different tokens, thereby enriching the model's learning from the same piece of data. Notably, there are no explicit external labels associated with each sentence or token; instead, the ground truth is derived directly from the input itself (a defining characteristic of self-supervised methods).

Another example of how self-supervised learning philosophy can be used to train complex and large models is CLIP. CLIP is a dual encoder for images and text that learns to produce similar embeddings for image–text pairs that share the same semantic meaning. To achieve it, pairs of images and texts were given as input to CLIP, and if those had the same semantic meaning, the label was 0 (1 otherwise). Hence, having two sets of images and texts, several

pairs of them can be made to train the model (positive and negative labels). Through this approach, the model learns a joint latent space where semantically related images and texts are mapped close to each other, enabling flexible use of either modality depending on the task. Furthermore, this framework can be extended to other data types, such as audio, by introducing additional encoders that project different data types into the same shared latent space.

The basic idea of contrastive learning is to try to position analogous instances close within a latent space while also separating dissimilar ones. Self-supervised learning is actually a supervised method because the model is optimized through a loss function whose inputs are the output of the model and the ground truth (as in any supervised method). Equation 3.1 represents a loss function for a model trained with a supervised learning approach.

$$\mathcal{L} = -\sum_{i=1}^{C} f(y_i, \hat{y}_i) \tag{3.1}$$

where:

- y_i is the ground truth label for class i (one-hot encoded)
- $\hat{y}_i$ is the predicted probability for class i (output of a softmax function)
- f can be any loss function (i.e., Cross entropy, Mean Squared Error (MSE), etc.).

Self-supervised learning differs from supervised learning in that it does not rely on externally provided labels. Instead, the ground-truth targets used for optimization in loss functions are derived directly from the inputs through a predefined transformation (see Eq. 3.2). A basic example is the autoencoder, where the identity function is applied to the input, making the input itself the ground truth. In the case of MLM, the transformation consists of randomly masking a subset of tokens from the input sequence; the model is then trained to predict these masked tokens based on the surrounding context.

$$y = f(d) \tag{3.2}$$

where

- y is the ground truth label.
- f can be any transformation applicable to data (masking tokens, identity, etc.).
- d data given as input to the model.

In this project, for each image gathered there is one label (the generator it belongs to). It is possible to train a model with those labels with a supervised learning approach, but it would not be able to tackle correctly instances of generators not sampled during the training phase. Contrastive learning can deal with unseen classes during the training phase as it does not predict its class but a numerical representation of it. It learns to extract valuable features (embeddings) that represent each class, hence unseen but related classes to the ones used in the training phase can also be processed correctly. In the AI-generated image detection and source attribution problem it is highly probable that new and unknown generators emerge,

thus it is compulsory to also detect images created by those generators. For that reason, a contrastive learning approach was applied.

Traditional contrastive learning methods consist in creating pairs of images so that those images could belong to the same generator (their label would be 0) or could belong to different generators (their label would be 1). The function applied to the data to extract the ground truth labels to train the model with a contrastive learning approach is shown in Equation 3.3.

$$y_i = \begin{cases} 0, & \text{if } g(im1) = g(im2) \\ 1, & \text{otherwise} \end{cases} \tag{3.3}$$

where

- y_i is the ground truth label.
- im_1 is preprocessed image 1.
- im_2 is preprocessed image 2.
- g represents the generator that created the image.

When multiple classes are present in the training set, supervised contrastive learning typically outperforms standard contrastive learning. This method combines key elements of supervised learning (where each sample has an associated class label) and contrastive learning, which encourages the model to pull similar samples closer and push dissimilar ones apart in the embedding space. Rather than constructing explicit pairs of similar and dissimilar samples (e.g., labeled 0 for similar and 1 for dissimilar), supervised contrastive learning compares each image in a batch to all other images, leveraging real class labels to identify positive (same-class) and negative (different-class) examples. This results in significantly more comparisons per batch and promotes the formation of more coherent and discriminative class clusters in the learned representation space.

Contrastive learning requires a substantial number of both positive and negative samples per batch. When batch sizes are small, the resulting clusters (learned embeddings of images) tend to overlap, which hinders effective differentiation and classification. Moreover, large batch sizes stabilize the convergence of the model, easing the learning phase and generalization of the model.

3.1.1 Types of contrastive learning

Models trained with contrastive learning learn to extract semantic embeddings from input data, capturing both statistical patterns and conceptual structures. This enables them to generalize to unseen classes not encountered during training (a major advantage over standard supervised learning, where the model can typically only recognize classes present during the training phase).

Depending on the number of samples available from the new classes, different learning regimes are defined:

- **Zero-shot learning:** The model has never encountered any examples of the new classes

during training. It is trained on a predefined set of classes and evaluated on a broader or disjoint set without additional fine-tuning.

- **One-shot learning:** The model is provided with a single labeled example per new class. Although only one instance is available, data augmentation techniques can generate synthetic variations, enriching the model's representation and aiding generalization.
- **Few-shot learning:** A small number of labeled examples per class (typically between five and ten) are available. Data augmentation can again be employed to expand the effective dataset, helping the model form more robust representations and improve its performance in new classes.

In this project, a zero-shot learning approach was adopted. This choice was motivated by the fact that new and unknown AI-based image generators are likely to emerge in the future. Therefore, building a model capable of handling unseen images produced by such potential generators is essential to ensure its robustness and long-term applicability.

3.2 Architecture

The base architecture of the proposed model is presented in Figure 3.1. Two stacked machine learning models compose the processing pipeline. The first model is in charge of extracting embeddings of images (embeddings extractor), while the second model classifies those embeddings (embeddings classifier). The embedding extractor is detailed in Section 3.2.1, and the embeddings classifier is described in Section 3.2.2. Each model is trained independently: the former using contrastive learning and the latter using supervised learning. This modular approach allows for flexible combinations of embedding extractors and classifiers, facilitating a thorough evaluation of which pairings yield the best performance.

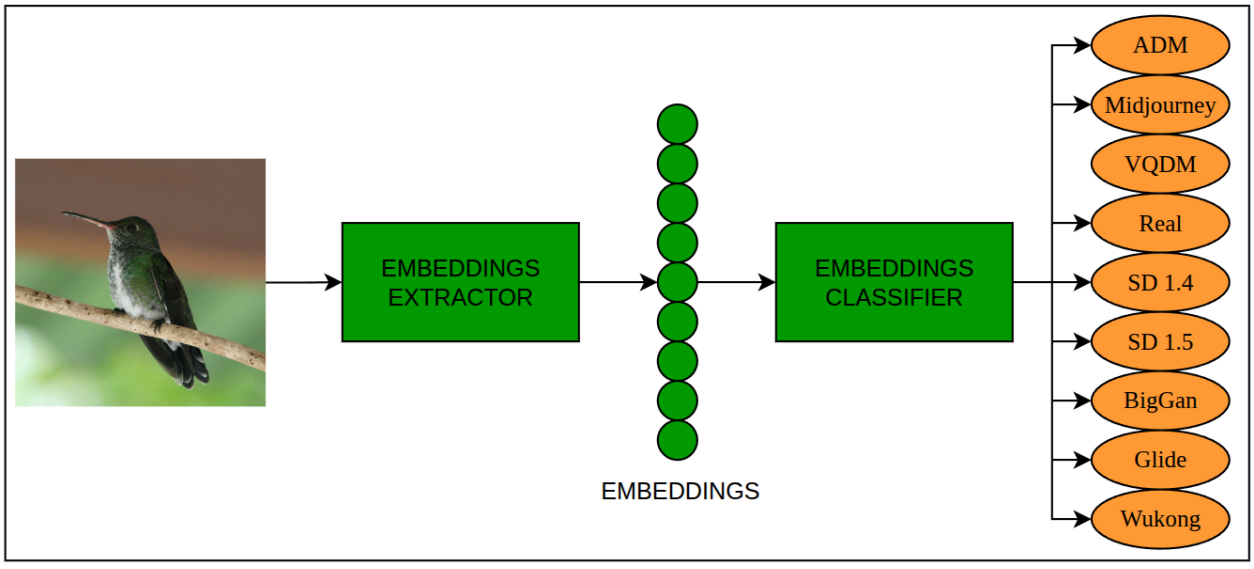


Figure 3.1: Full architecture of the proposed method.

3.2.1 Embeddings extractor

Since a contrastive learning approach, specifically zero-shot learning, is employed in this project, it is necessary to design a model capable of processing two images, extracting their embeddings, and comparing them in order to update the model's weights. To effectively compare the embeddings of two images, each position within the embeddings must correspond to the same semantic feature. To do so, two approaches can be followed:

- **Use the same latent space for embeddings:** To extract consistent semantic features from a pair of images, both images must be processed by the same neural network. This constraint is naturally satisfied by Siamese Networks, as the backend networks of siamese networks have the same architectures and share their weights (they are two copies of the same network). In siamese networks, both images are processed by the same backbone, so that the obtained embeddings represent the same semantic features (see Fig. 3.2).
- **Project embeddings into a shared latent space:** If images are processed by different networks (the architectures of the backends may differ greatly), the semantic meaning associated with each position in the embeddings is likely to differ. To address this, a feedforward network can wrap the backend networks to project the embeddings into a common latent space, ensuring that both representations capture comparable semantic information. Figure 3.3 depicts this approach, where the embeddings of the images are initially extracted using different backbone architectures (represented in blue and green). These embeddings are then projected into a common latent space through a wrapper network (depicted in pink). As a result, both embeddings encode the same semantic features, enabling a meaningful comparison and allowing the loss to be computed accordingly.

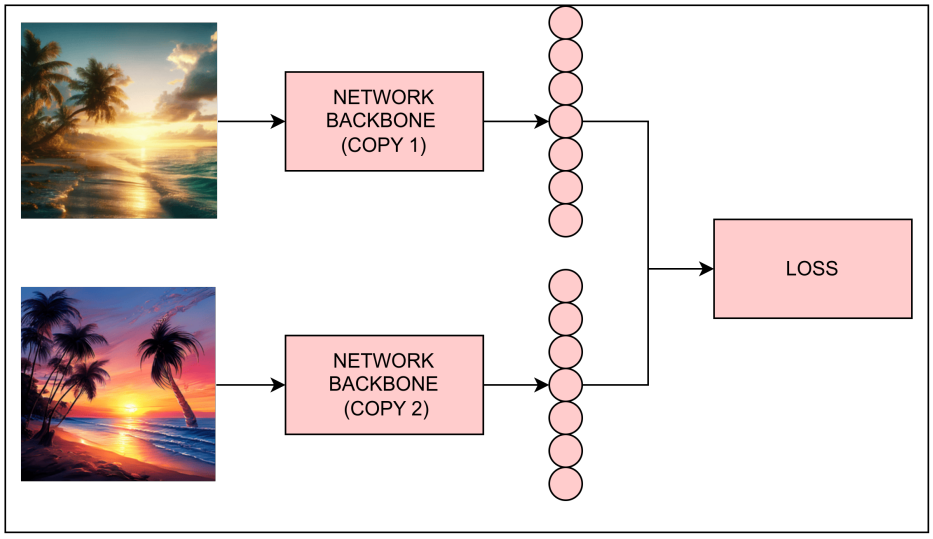


Figure 3.2: Siamese Network architecture.

Siamese networks are deterministic models, as their backbones consist of two identical copies of the same architecture that share weights. Consequently, it does not matter which backbone

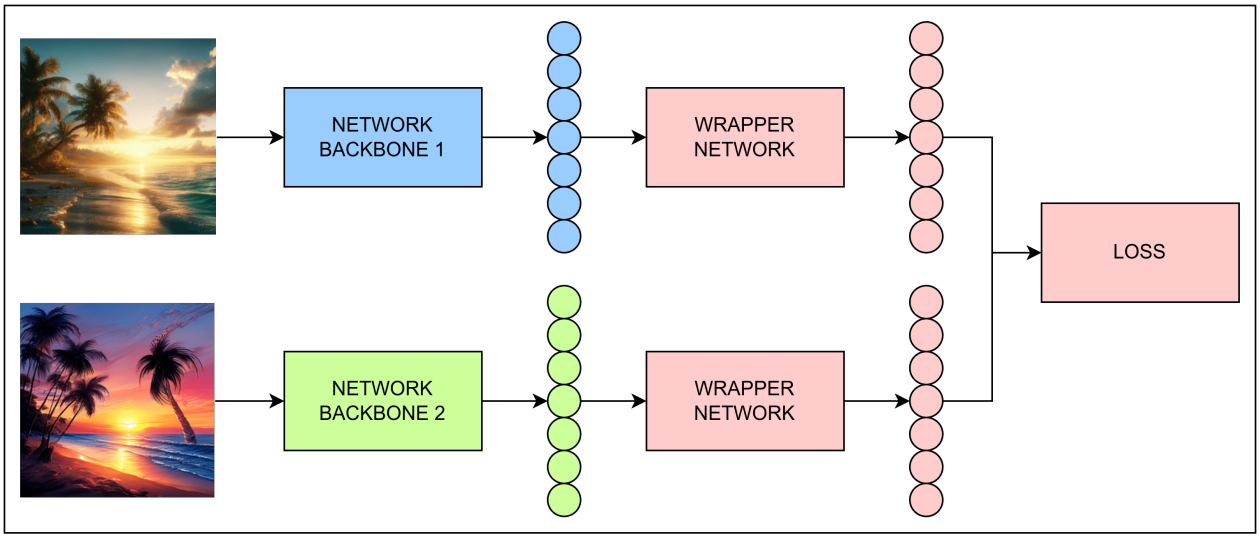


Figure 3.3: Architecture of projecting embeddings into the same latent space.

processes which image, since both networks are identical, and the resulting embeddings remain unchanged.

In contrast, using different backbone networks results in a stochastic model, where the output depends on which backbone processes each image, introducing significant variability. To mitigate this variability, the wrapper network must be considerably more complex, which increases the risk of overfitting and capturing noise from the training set. Moreover, the added complexity leads to higher computational and processing costs.

For these reasons, the backbone selected for this project is based on Siamese networks (see Fig. 3.2), with EfficientNetB0 [18] selected as the backbone architecture. EfficientNet is an architecture in which its dimensions are proportionate. Increasing one dimension but maintaing the others unchanged may arise a series of problems:

- **Depth:** Refers to the number of layers stacked in the network. If depth increases disproportionately, a degradation problem may arise. This phenomenon is characterized by an increase in training error beyond a certain point, not due to overfitting but due to optimization difficulties. The network becomes too complex, requiring the adjustment of a large number of parameters, often with insufficient data to support effective training.
- **Width:** Denotes the number of channels in each layer. If the network is excessively wide relative to its other dimensions, it may suffer from overfitting. Additionally, very wide architectures can encounter vanishing or exploding gradient problems, complicating the training process.
- **Resolution:** Corresponds to the input image size. Increasing the resolution raises the computational and time complexity of the model. Furthermore, higher-resolution inputs may introduce data complexity that exceeds the model's capacity, making it challenging for the network to effectively learn all relevant features.

EfficientNets are designed with proportionate scaling across depth, width, and resolution,

enabling them to achieve optimal performance. For this project, EfficientNetB0, the smallest variant in the EfficientNet family, was chosen as the backbone for the Siamese network. This decision was primarily driven by the limitations in available hardware resources (see Section 3.5.1. Large datasets were used in all experiments (GenImage and ForenSynths), and increasing the model size would significantly raise both time and computational complexity. The Siamese Network architecture was used for the Euclidean Distance loss (see Section 3.4.1), while the base EfficientNetB0 was used with the SupconLoss function (see Section 3.4.2)

3.2.2 Embeddings classifier

The goal of this model is to classify the embeddings extracted by the preceding contrastive learning model. It is trained using a supervised learning approach, where the input consists of image embeddings (treated as feature representations), and the output corresponds to the predicted generator that produced each image. This approach solves the source attribution problem. On the other hand, to address the task of AI-generated image detection, all images predicted as being produced by any generator are grouped under a single AI-generated class, while images originating from real sources form the real class.

Several approximations were tested to decide not only which machine learning model yields to the best results, but how to aggregate the results from the embeddings extractor model:

- **Feedforward network prediction with generators' centroids:** The centroid of each generator in the embedding space is computed by averaging the embeddings of all training images belonging to that generator. To classify a new instance (referred to as the anchor), its embedding is compared against the centroids. A feedforward neural network is used for this comparison: its input consists of the concatenation of the anchor embedding and a centroid embedding, and its output is binary (0 if both embeddings are predicted to belong to the same class, and 1 otherwise). The anchor is compared to all centroids, and the final prediction corresponds to the centroid that yields the highest probability of being from a class (i.e., the highest probability of class match).
- **Feedforward network prediction using a single representative per generator:** A single image is randomly selected from each generator to serve as its representative. The classification process follows the same strategy as in the centroid-based method. The embedding of the anchor image is compared to the embeddings of all representatives using a feedforward neural network, whose input is the concatenation of the anchor and representative embeddings. The network outputs 0 if both embeddings are predicted to belong to the same class, and 1 otherwise. The final prediction is determined by selecting the representative with the highest probability of class match. This method was initially tested, but it presented high variability due to the random representative of each generator, hence it was not used across all experiments.
- **Nearest distance to generators' centroids:** Centroids for each generator are computed as in the first method, and a distance metric (Euclidean distance) is computed between the anchor image and all centroids. The label of the centroid to which the distance is the shortest is the final prediction.

- **KNN:** A KNN algorithm is trained with the embeddings of the images of a subset of the training set. The algorithm identifies the N nearest embeddings in the latent space, so that the predicted label for the anchor image is determined by the mode of the labels associated with these N nearest neighbors.

3.3 Data and preprocessing

As discussed in Section 2.1, several datasets are available for both AI-generated content detection and source attribution tasks. However, GenImage and ForenSynths stand out as the datasets comprising the largest number of images generated by a diverse and heterogeneous set of models [19].

GenImage consists of images generated by eight distinct generative models, each contributing 162,000 synthetic images for training and 6,000 for testing (except Stable Diffusion 1.5, which contains 8,000 samples for testing). For each generator, it is also included 162,000 real images for training and 6,000 for testing (except for Stable Diffusion 1.5, where 8,000 real images are included). The synthetic images were generated to match the class distribution of ImageNet, ensuring consistency in content and facilitating direct comparison between real and synthetic samples.

The 8 generators that uses GenImage to create its synthetic images are:

- **ADM:** based on Denoising Diffusion Probabilistic Models (DDPMs), using a UNet-like architecture trained to reverse a gradual noising process. It generates high-fidelity images through iterative denoising, conditioned on class labels for controllability.
- **BigGan:** it is a large-scale, class-conditional GAN based on the traditional GAN framework, enhanced with techniques like spectral normalization and self-attention. It generates high-resolution images with strong realism but requires significant computational resources and is less prompt-controllable compared to diffusion models.
- **Glide:** text-to-image diffusion model developed by OpenAI, based on a denoising diffusion process conditioned on text descriptions. It uses a UNet-like architecture with classifier-free guidance, enabling both generation and editing of images in a highly controllable manner.
- **Midjourney:** Midjourney is a proprietary model whose exact architecture is not publicly disclosed. However, it is believed to be based on an ensemble of diffusion models, likely similar to or evolved from latent diffusion models, heavily fine-tuned on aesthetic and artistic datasets for high-quality, stylized image generation.
- **VQDM:** VQDM combines vector quantization (VQ) techniques with diffusion models. Images are first encoded into a discrete latent space via a VAE, and diffusion is applied in this compressed space rather than pixel space, enhancing efficiency and scalability while maintaining generation quality.
- **Stable Diffusion 1.4 and Stable Diffusion 1.5:** Stable Diffusion is a latent diffusion model (LDM) that operates on a compressed latent space instead of pixel space,

dramatically improving efficiency. It uses a UNet backbone combined with a variational autoencoder (VAE) for encoding/decoding and a text encoder (CLIP) for conditioning on prompts. Versions 1.4 and 1.5 mainly differ in training data scale and fine-tuning, with 1.5 being a slight improvement in quality and prompt alignment.

- **Wukong:** a large-scale text-to-image generation model trained on a Chinese-language dataset. Architecturally, it is based on a diffusion model similar to those used in GLIDE and Stable Diffusion, incorporating a CLIP-style text encoder for semantic alignment between text and generated images.

ForenSynths is another widely-used dataset used in the AI-generated image detection. Contrary to GenImage, ForenSynths dataset uses just one generator to create synthetic images in the training set, while 12 in the test set. The training set is comprised of 360,000 ProGan-generated images and 360,000 real images.

Test set consists of real images and fake images, whose generators are: BigGan, CycleGan, DeepFake, GauGan, Cascaded Refinement Network (CRN), ProGan, Seeing In The Dark (SITD), StarGan, StyleGan, StyleGan2, Second Order Attention Network (SAN) and Implicit Maximum Likelihood Estimation (IMLE). The number of real and fake images for each generator differs (see Table 3.1). Table 3.1 also depicts the native resolution of each generator.

Table 3.1: Number of real and fake images (and their resolution in pixels) per generator in ForenSynths dataset.

	BigGan	CycleGan	DeepFake	GauGan	CRN	ProGan	SITD	StarGan	StyleGan	StyleGan2	SAN	IMLE
# Real Images	2,000	1,321	2,707	5,000	6,382	4,000	180	1,999	5,991	7,998	219	6,382
# Fake Images	2,000	1,321	2,698	5,000	6,382	4,000	180	1,999	5,991	7,988	219	6,382
Width Resolution	256	256	256	256	512	256	4,256	256	256	256	500	512
Height Resolution	256	256	256	256	256	256	2,848	256	256	256	480	256

Only the training split of the ForenSynths dataset was used in this project. Following the approach adopted by other authors, models were trained on the ForenSynths training set and evaluated on the GenImage test set. This setting was maintained to enable a fair comparison between our proposed method and other state-of-the-art approaches.

3.3.1 Preprocessing

Each generator produces images at a different native resolution. Therefore, all images must be rescaled to a common resolution to ensure compatibility with the input requirements of EfficientNet. Tables 3.1 and 3.2 presents the original resolution used by each generator in GenImage and ForenSynths respectively in pixels. In GenImage, all generators produce square images, meaning that the width and height of each image are equal.

Table 3.2: Native image resolution (in pixels) produced by each generator in GenImage dataset.

	ADM	BigGan	Glide	Midjourney	VQDM	SD 1.4	SD 1.5	Wukong
Width Resolution	256	256	1024	512	256	512	512	128
Height Resolution	256	256	1024	512	256	512	512	128

All images were rescaled to a resolution of 256×256 pixels, selected to accommodate the limited computational resources available. This decision is particularly relevant in the

context of contrastive learning, which benefits from large batch sizes to optimize performance. Using higher-resolution images would significantly increase memory consumption per batch, potentially exceeding hardware limitations and hindering training efficiency.

All rescaled images were normalized to the [0,1] interval, as neural networks generally perform better when input data is normalized. Since the images are encoded in RGB format, with pixel values originally ranging from 0 to 255, each pixel was divided by 255 to achieve the desired normalization.

3.3.2 Dataset Partitioning

The GenImage dataset is inherently imbalanced, containing over 1,296,000 real images but only 162,000 images per generator. This imbalance arises because the dataset was originally designed for training binary classifiers to distinguish between real and synthetic images produced by a specific model (each generator subset had over 162,000 real and 162,000 synthetic images). However, in this project, the goal was to train a single model capable of detecting AI-generated images from any generator, including potentially unknown ones. To create a balanced training set, real images were downsampled to 162,000, matching the number of images available for each generator. Since each class (including the real images) had a sufficiently large number of samples, data augmentation was unnecessary. As a result, the dataset consisted of 9 classes (8 generators and the real class), with 162,000 images for training and 6,000 for testing (except Stable Diffusion 1.5 which had 8,000 images),

ForenSynths training set is balanced (720,000 real and ProGan-synthetic images, 360,000 each), thus no downsample nor data augmentation were required. In this project, ForenSynths test set is not used.

To construct the validation sets, the original training sets were split into a final training set and a validation subset. The former consisted of 90% of the data (145,800 images per class in GenImage and 324,000 in ForenSynths), while the latter comprised the remaining 10% (16,200 images per class in GenImage and 36,000 in ForenSynths).

To evaluate the performance differences between models trained with varying class subsets under zero-shot and non-zero-shot conditions, distinct splits for training, validation, and testing were created. For clarity, we refer to each experimental configuration as an Experiment Setting (abbreviated as ES), followed by a number (e.g., *ES1*, *ES2*, etc.). Each setting corresponds to a specific training and evaluation partition, as detailed below.

- **ES1:** *8 generators, no real images.*
  In this configuration, real images are excluded from both the training and validation sets. The model is trained solely on synthetic data generated by eight different models, and evaluated on a test set containing both real and synthetic images. This constitutes a zero-shot setting for real image detection. The goal is to assess whether a model exposed only to synthetic content during training can effectively distinguish real images from generated ones. This setup is applied to both the AI-generated image detection and source attribution problem.
- **ES2:** *8 generators with real images.*

This configuration represents the most comprehensive training scenario, utilizing images from all eight generative models along with real-world images. The purpose is to establish an upper bound on the performance of the proposed method (see Section 3.2). It serves as a benchmark for evaluating the model's ability to distinguish AI-generated images from real ones, and accurately attribute generated images to their source models.

- **ES3:** *1 generator with real images.*
  This experiment addresses the problem of AI-generated image detection under a highly restricted training setup. The model is trained with the training set of ForenSynths dataset, composed of ProGAN-generated and real images. Evaluation is conducted on the GenImage test set, which includes outputs from a diverse set of generative models, being all generators tested in a zero-shot approach, as none of them were included in the training set.

  This model was employed exclusively for the task of AI-generated image detection. In this domain, the evaluation metrics used were the Area Under the Curve (AUC) and accuracy. For the purpose of AUC calculation, the "Real" class was designated as the positive class. The predicted scores were sorted in descending order, and the corresponding ground truth labels were reordered accordingly. For each unique prediction score, a threshold was applied to generate binary classifications, from which the counts of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) were computed. These values were then used to calculate the True Positive Rate (TPR) and False Positive Rate (FPR), as defined in Equation 3.4. The Receiver Operating Characteristic (ROC) curve is constructed by plotting TPR against FPR at all thresholds, and the AUC is obtained by computing the area under this curve.

  On the other hand, accuracy is defined as the proportion of correctly predicted samples relative to the total number of samples. Both accuracy and AUC were computed separately for each image generator, enabling a generator-specific evaluation of the model's performance. This approach provides a more granular understanding of how well the model distinguishes real from generated images across different generation models.

$$\begin{aligned} \mathrm{TPR} &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \\ \mathrm{FPR} &= \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \end{aligned} \tag{3.4}$$

- **ES4:** *5 generators with no real images.*
  Real images and three of the eight generators are removed from the training set. This setup evaluates the zero-shot generalization capabilities of the model (specifically, whether it can differentiate between unseen generators and real images, having been trained on a limited subset of generators). These experiments were conducted and tested only in the source attribution domain. A comprehensive experiment would involve evaluating all possible subsets of five generators. However, this results in a total of 56 combinations, which presents a significant computational cost. Therefore, a representative sample of these subsets was selected for evaluation.

- **ES4.1:** *Exclusion of BigGAN, Stable Diffusion 1.4, and Stable Diffusion 1.5 from training.*
  This scenario evaluates the model's capacity to differentiate between two closely related generative models (Stable Diffusion 1.4 and 1.5) neither of which are seen during training. Since both models share architectural foundations but differ in training data and fine-tuning, the goal is to test whether the model can identify subtle differences and generalize to variants of diffusion models it has not been exposed to. This also serves as an indirect measure of the model's sensitivity to intra-family variations.
- **ES4.2:** *Removal of Midjourney, BigGAN, and VQDM from the training set.*
  This configuration isolates the training to only include diffusion-based generators, intentionally excluding GANs and vector quantized models. The objective is to evaluate the model's ability to detect and classify samples from entirely different generative paradigms (GAN, VQ) when it has only been trained on diffusion-based images. Success in this task would imply robustness in identifying structural and statistical anomalies between different generation techniques.
- **ES4.3:** *Exclusion of GLIDE, VQDM, and Stable Diffusion 1.5 from training.*
  Here, the model is trained with Stable Diffusion 1.4 but not 1.5, aiming to quantify how transferable representations from one variant are to another. Since Stable Diffusion 1.5 is a fine-tuned version of 1.4, the ability to accurately classify its outputs suggests that the learned features from 1.4 sufficiently generalize. The inclusion of VQDM and GLIDE further challenges the model by increasing the diversity of excluded architectures.
- **ES4.4:** *Wukong, Stable Diffusion 1.4, and Stable Diffusion 1.5 are excluded from training.*
  These models are CLIP-based conditioning for text-to-image generation. This scenario explores whether a classifier trained on generators that do not employ CLIP can accurately detect those that do. It serves as a test of architectural awareness and investigates whether CLIP-conditioned outputs exhibit distinguishable patterns or visual characteristics that a model can learn to detect.
- **ES4.5:** *Exclusion of Wukong, ADM, and GLIDE from training.*
  This experiment removes a diverse mix of diffusion models from the training phase: Wukong and GLIDE are CLIP-conditioned, while ADM represents a class of unconditional diffusion models. The intent is to assess the model's ability to generalize across both conditional and unconditional architectures. Success in this setting would suggest that the model captures deep, transferable generative patterns for each generator.

3.4 Losses

Loss functions play a central role in deep learning by guiding the model to learn the task effectively and accurately. They provide the only feedback for the model to evaluate the

quality of its predictions. Therefore, employing a well-designed and informative loss function is essential for successful training.

In the context of contrastive learning, two types of loss functions are particularly relevant: Euclidean distance-based loss and Supervised Contrastive Loss (SupConLoss). Each offers distinct advantages and trade-offs. This work explores and evaluates both loss functions within the framework of the proposed model.

As discussed in previous sections, the central objective of contrastive learning is to pull together the embeddings of instances belonging to the same class while pushing apart those from different classes. To quantify this similarity or dissimilarity, a natural choice is to use a distance metric. Common options include Euclidean distance, Minkowski distance, and Manhattan distance, among others.

To be valid and meaningful in the context of contrastive learning, any function used to measure the distance between two embeddings must satisfy the three fundamental axioms of a mathematical distance. These properties ensure that the geometry of the learned embedding space is consistent and interpretable, which is critical for effectively separating different classes and clustering similar ones. The three essential properties are as follows:

1. **Non-negativity:**

$$d(x, y) \geq 0 \quad \text{and} \quad d(x, y) = 0 \iff x = y$$

   This property ensures that distances are always non-negative and that the only time the distance between two points is zero is when the points are identical. It prevents the model from learning incorrect representations where dissimilar points could have zero or negative distances.

2. **Symmetry:**

$$d(x, y) = d(y, x)$$

   Symmetry guarantees that the distance from point x to point y is the same as the distance from y to x. This represent the reflexive symmetry: if x is similar to y, then y should be equally similar to x. Violating this property could lead to inconsistent similarity rankings and unstable optimization.

3. **Triangle Inequality:**

$$d(x, z) \leq d(x, y) + d(y, z)$$

   The triangle inequality enforces a coherent spatial structure by stating that the direct path between two points x and z should never be longer than taking a detour through a third point y. This prevents contradictory distances and ensures that the learned space remains consistent, which is especially important in high-dimensional embedding spaces where transitivity of similarity is crucial.

Together, these three properties ensure that the embedding space respects the foundational rules of distance measurement, allowing contrastive loss functions to operate effectively and reliably. Using a distance function that violates any of these properties could lead to degraded performance and useless representations.

3.4.1 Euclidean distance

Euclidean distance satisfies the aforementioned constraints, hence it can be used as loss metric to train a contrastive learning model. Ideally, similar instances' distance would be 0 (their embeddings are the same), and dissimilar ones' distance would be infinite (their embeddings are completely different). Nevertheless, setting a distance to infinite is hard computationally and probably challenging for the model to predict it. For that reason, a margin is set, so that distances between dissimilar instances are minimum that set margin. The formula for the Euclidean distance loss is as follows:

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two embeddings of the images, and let $y \in \{0, 1\}$ be a binary label indicating whether the pair is similar ($y = 0$) or dissimilar ($y = 1$). The Euclidean distance between the embeddings is defined as:

$$D = \|\mathbf{x}_1 - \mathbf{x}_2\|_2$$

The contrastive loss function is then given by:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (1 - y_i) \cdot D_i^2 + y_i \cdot [\max(0, m - D_i)]^2$$

where:

- D_i is the Euclidean distance between the i-th pair,
- y_i is the label for the i-th pair,
- m is the predefined margin,
- N is the number of pairs in the batch.

If the images in a pair are dissimilar ($y_i = 1$), the first term of the contrastive loss function becomes zero, and only the second term contributes. In this case, the loss evaluates how far the embeddings are separated relative to a predefined margin m. If the Euclidean distance D_i between the embeddings is greater than the margin, it indicates that the model has correctly pushed apart the dissimilar pair, resulting in zero loss. However, if D_i is less than the margin, the model has not sufficiently separated the pair, and a penalty proportional to $(m - D_i)^2$ is returned, encouraging greater separation.

On the other hand, if the images are similar ($y_i = 0$), the second term of the loss is zero, and the only contribution comes from the first term, which is D_i^2. A small distance implies that the model is correctly clustering similar embeddings, with the ideal scenario being $D_i = 0$, leading to no loss. As the distance between embeddings increases, the loss increases quadratically, penalizing the model for failing to bring similar samples close together in the learned embedding space.

This formulation encourages dissimilar pairs ($y = 1$) to further than the predefined margin m, and similar pairs ($y = 0$) to be as close as possible.

This approach presents two main disadvantages. The first one is the presence of the margin hyperparameter, which must be manually defined. In practice, this often requires performing an extensive grid search to identify the optimal value, increasing computational cost and experimentation time.

The second drawback is related to efficiency. Training with distance-based methods involves learning from pairs of images, which reduces the number of unique samples available per batch. As discussed in Section 3.1, contrastive learning methods typically benefit from large batch sizes to extract valuable and rich semantic embeddings. However, due to the pairwise structure, distance-based losses inherently constrain batch size, potentially limiting the performance of the model.

3.4.2 SupConLoss

SupConLoss mixes both the supervised and contrastive learning approaches. This loss compares each feature vector embeddings (anchor) with all other samples in the batch. It encourages representations of samples from the same class to be closer (high similarity), and representations from different classes to be pushed apart.

Compared to distance-based methods, which rely on comparing images in isolated pairs, the Supervised Contrastive Loss (SupConLoss) considers each image in relation to all other images in the batch. This enables the model to simultaneously pull together samples from the same class and push apart samples from different classes within a single loss computation. Negative samples (images with different classes) encourage the model to separate embeddings of samples of different classes, while positive samples (images with the same class) serves to produce similar ones. In conclusion, negative samples produce separation between clusters, and positive samples make those clusters denser. Figure 3.6 illustrates the distinction between intra-class compactness and inter-class separation. Albeit 3.4 has lager distances between each sample to their cluster center (intra-cluster distance, produced due to several negative pairs and few positive ones in train set), it is easier to classify instances than in Figure 3.5, whose intra-cluster and inter-cluster distance are smaller (balanced positive and negative samples in training set).

For this reason, negative samples play a crucial role in contrastive learning, as they enforce the separation between clusters necessary for effective instance classification. When clusters are sufficiently well-separated in the embedding space, the precise compactness of each cluster becomes less critical, since the inter-class boundaries are already clearly distinguishable.

A key advantage of SupConLoss, particularly in non-binary problems, is its ability to differentiate between various negative classes. While distance-based methods treat all dissimilar pairs uniformly (penalizing them equally regardless of the specific class difference) SupConLoss is class-aware. It leverages label information to ensure that only instances of the same class are considered positive pairs, while all other instances serve as structured and informative negatives. This results in a richer learning feedback, especially in non-binary classification tasks. The performance of SupConLoss improves significantly with both the number of classes in the training set and the batch size. As the number of classes increases, more negative pairs can be formed, leading to greater inter-class separation and, consequently, improved

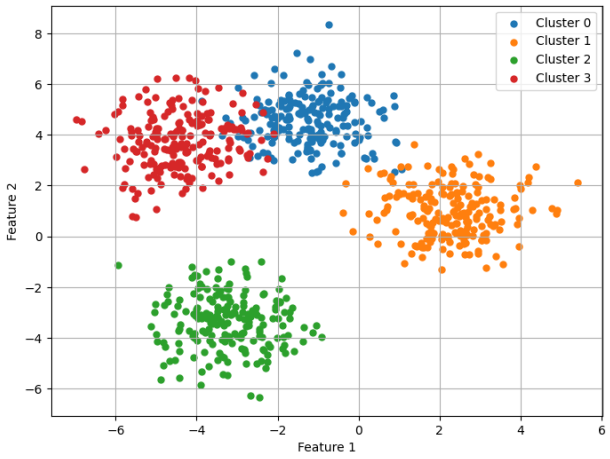


Figure 3.4: Embeddings representation with higher inter-cluster separation.

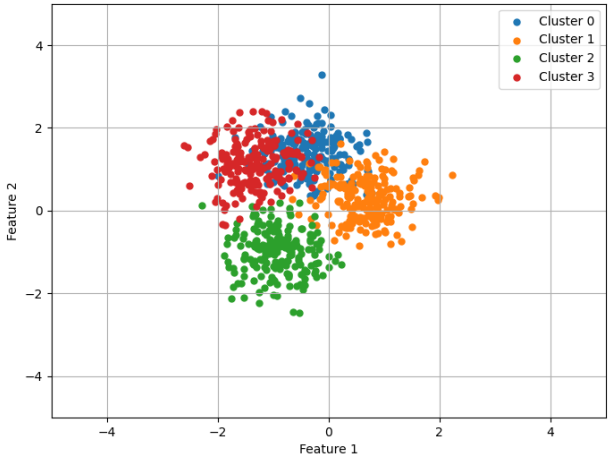


Figure 3.5: Embeddings representation with higher intra-cluster separation.

Figure 3.6: Difference between inter-cluster and intra-cluster separation.

classification accuracy.

Given a batch of N samples let $\mathbf{z}_i \in \mathbb{R}^d$ denote the embeddings of the i-th view, and y_i its corresponding class label. In this case, due to the fact that plenty of images per class were available (162,000), no augmentation views were needed.

The supervised contrastive loss for an anchor $\mathbf{z}_i$ is defined as:

$$\mathcal{L}_i = \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)}$$

Where:

- $P(i)$ is the set of indices of all positive samples of z_i (same class).
- $A(i)$ is the set of indices of all samples except z_i.
- τ is the temperature scalar (a learnable parameter),
- $z_i \cdot z_j$ is the dot product (cosine similarity).

The total loss is averaged over all anchors in the batch:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i$$

3.5 Experiments

A series of experiments were conducted to optimize the performance of the model. Specifically, these experiments aimed to identify which loss function enabled the model to learn more

efficiently, as well as which classifier best distinguished the embeddings produced by the model. Further experiments were carried out to assess how changes in hyperparameter values affect the model's overall performance (i.e., batch size, embedding size, etc.).

3.5.1 Hardware

The hardware configuration was kept consistent across all experiments to ensure the comparability of the results. The Graphics Processing Unit (GPU) employed was an NVIDIA GeForce RTX 4070 with 12 GB of Video Random Access Memory (VRAM). The relatively limited VRAM capacity posed a significant bottleneck, requiring smaller batch sizes and, as a result, more frequent updates to the model's weights, slowing down the entire process. This constraint might notably influence model performance; therefore, specific experiments were designed to assess its impact.

The experiments were carried out on a system equipped with an Intel Core i9-14900K processor, featuring 24 cores and 48 threads (2 threads per core). However, the processor was minimally utilized, as the majority of computations were offloaded to the GPU to accelerate and streamline the overall process.

3.5.2 AI-generated image detection

A series of experiments were conducted to assess how the proposed method perform when trained on different sets. Specifically, the experiments made were *ES1*, *ES2* and *ES3* (see Section 3.3.2). The aim of *ES1* was to assess whether a contrastive model trained solely with synthetic images can differentiate between AI-generated and real images. *ES2* measurds the higher-bound metrics our method can achieve (it was trained with the full training set), and *ES3* performed a zero-shot test, as it was trained with the training set of ForenSynths dataset (real and ProGan images) and tested with the GenImage test set.

The hyperparameters and configurations of the models remained unchanged across the experiments, to effectively compare the differences that the training sets can produce, as well as provide comprehensive results and discussion.

The loss function used to train the models was SupConLoss. Unlike traditional contrastive loss functions like Euclidean distance, SupConLoss takes advantage of class label information by encouraging embeddings of samples from the same class to cluster together in the latent space, while pushing apart embeddings from different classes. The effectiveness of SupConLoss increases with larger batch sizes, as more comparisons can be made between samples within a batch. This allows the model to gather richer information and receive more informative gradients, resulting in better weight updates. In this work, a batch size of 186 was used as it was the maximum supported given the available VRAM.

Another critical hyperparameter affecting VRAM usage is the image resolution. While some studies use higher resolutions, 256×256 pixels was chosen here as a balance between computational cost and performance. This resolution is often sufficient to retain essential features while keeping memory usage low. Since VRAM usage scales quadratically with image resolution (due to increases in both width and height), higher resolutions can drastically reduce

the batch size, which may degrade performance when using SupConLoss. Thus, reducing image resolution in favor of a larger batch size often yields better results.

Finally, the embedding size was set to 128, which strikes a compromise between VRAM usage and representational capacity. This dimensionality is sufficient to encode distinguishing features from the image generators while remaining small enough to fit within the memory constraints.

In this framework, the task of detecting AI-generated content is approached through the source attribution problem. The contrastive learning models trained in each experiment were trained to map images into a representation space where embeddings reflect the characteristic patterns of each synthetic generator. The reason behind it is that SupConLoss obtains better results when used with more classes, so instead of grouping all synthetic images in the AI-generated label, the model had to cluster samples of the same generators and separate instances of different ones in the latent space. A crucial advantage of this approach is that these models can be used both for the AI-generated image detection and source attribution.

To train the KNN classifier which makes the final classification, embeddings of all images used to train the contrastive learning were obtained. Synthetic images were grouped into the AI-generated class, so that the KNN classifier performed a binary classification (synthetic or real).

At inference time, embeddings were computed for all test images, including the 8 generators and real images, and the KNN algorithm classified those embeddings between AI-generated and real images. The results of the three experiments (*ES1*, *ES2* and *ES3*) are reported in Section 4.1 and further analyzed in Section 5.1.

The primary evaluation metrics used in this domain were Area Under the Curve (AUC) and accuracy. To obtain more granular insights into model performance, these metrics were computed separately for each synthetic generator in comparison with the real class. This pairwise evaluation enables a detailed analysis of how well the model distinguishes real images from those generated by each specific source, providing a more comprehensive assessment than aggregated metrics alone.

Baselines

A series of models and architectures used to address the AI-generated image detection problem were selected as baselines to be compared with our proposed approach. All the baselines results presented in Section 4.1 were trained with ForenSynths's training set (comprised of Pro-Gan and real images), and tested on GenImage's test set, to assure all generators in the test set were not seen during the training phase.

- [28] introduces a novel approach to distinguish real images from those generated by diffusion models (though it was also tested in generators with other architectures). The authors proposed a metric called Diffusion Reconstruction Error (DIRE), which quantifies the difference between an input image and its reconstruction by a pretrained diffusion model. They observed that diffusion-generated images can be accurately reconstructed by the same model, whereas real images exhibit significant discrepancies upon reconstruction.

This characteristic enables DIRE to effectively identify AI-generated images, even from unseen diffusion models.

- The authors in [41] used pretrained vision-language models (specifically CLIP) to address the AI-generated image detection problem. They used the vision encoder of a pretrained CLIP model to extract high-level visual features from images, and on top of it, they trained a linear classifier. Contrary to previous assumptions, the authors demonstrate that a lightweight CLIP-based detector can achieve high generalization and robustness by training on only a few examples from a single generative model (strong zero-shot restriction).

- [43] introduces a novel approach to detecting AI-generated images using limited training samples (as in [41]). The authors proposed a Forgery Awareness Module (FAM) based on Low-Rank Adaptation (LoRA), which effectively adapts pretrained models to the task of AI-generated image detection. Additionally, they introduce a Semantic feature-guided Contrastive learning strategy (SeC) that guides the model to focus on distinguishing features between real and fake images. This combination allows the model to generalize well across different generative models with minimal training data.

  Regarding the image processing process, FAMSeC utilizes a pretrained backbone network to extract features from input images. The FAM module adapts these features to be more sensitive to forgeries, while the SeC strategy ensures that the model learns to contrast real and fake images based on semantic differences. This approach enables the model to effectively detect AI-generated images even when trained on a small number of samples.

- [49] proposed a method to identify AI-generated images by leveraging the inherent noise patterns present in real images. Unlike traditional approaches that focus on detecting visual artifacts or invisible patterns in generated images, this method examines the frequency domain characteristics of real images. The authors observed that real images exhibit similar noise patterns in the frequency domain, whereas generated images differ significantly. By training a simple classifier to distinguish these noise patterns, the method effectively detects a wide range of generative models.

- The authors in [89] introduced a novel approach by focusing on the up-sampling operations in CNN-based generative networks rather than frequency-based artifacts. The authors observed that up-sampling layers, commonly used in GANs and diffusion models, introduced local interdependencies among image pixels, leading to unique structural artifacts. They proposed the concept of Neighboring Pixel Relationships (NPR) to capture these artifacts, which are invariant to translation and can generalize across different generative models. By training a detector on these NPRs, the method achieved state-of-the-art performance.

- In [90], authors followed a similar approach to [41], as they also used the image encoder of a CLIP pretrained model. Nevertheless, in this case, they did not train any layer on top of it to carry out a binary classification, but they used a KNN model, comparing test images to a small set of real image embeddings. Rather than detecting model-specific visual artifacts, this method leveraged general semantic and visual inconsistencies

between real and synthetic images in CLIP's feature space, enabling detection with minimal or no training, as CLIP was pretrained, and the KNN algorithm does not need any training.

- In [91] the authors proposed the method Learning on Gradients (LGrad), which consists in using a pretrained CNN as a transformation model to convert images into their gradient representations. These gradients are then analyzed to identify generalized artifacts that are characteristic of GAN-generated images (initially proposed for GAN models). By focusing on these artifacts, the method aimed to construct a detector that generalized across different generative models and datasets. The transformed gradient representations emphasized the generalized artifacts introduced by GAN-generated images, which are less sensitive to the specific content of images and more tied to generation artifacts.

3.5.3 Image source attribution

The approach described in Section 3.5.2 for training contrastive models is extended to image source attribution task. In this case, the conducted experiments were *ES1*, *ES2* and *ES4* (which comprises *ES4.1*, *ES4.2*, *ES4.3*, *ES4.4* and *ES4.5*). The test set remained consistent with the previous section and included images from all eight generators as well as real images. Contrastive models from *ES1* and *ES2* were reused, as commented in the previous section, they were trained to the detect the sources of each image.

Once trained, the contrastive models were used to extract embeddings for all test images. These embeddings were then passed through a supervised classifier for final classification. Although in some experiments the contrastive model was trained in a zero-shot setting (without exposure to three of the generators), the supervised classifier is trained on embeddings corresponding to all eight generators and real images (supervised models cannot predict classes not seen during the training phase). This follows the same supervised classification strategy described in Section 3.2.2.

In the context of source attribution, although the field is less mature than AI-generated image detection, existing research typically evaluates performance using the F1-score and standard accuracy. Therefore, these metrics were adopted to facilitate comparisons between the proposed method and existing state-of-the-art approaches.

Results of all conducted experiments regarding the source attribution problem are presented in 4.2 and further discussed in 5.2.

3.5.4 Ablations

A series of preliminary experiments were conducted prior to the main evaluations commented in Sections 3.5.2 and 3.5.3 in order to optimize key hyperparameters and ensure optimal performance of the models.

- **Batch Size:** This is a critical hyperparameter, as discussed in Section 3.1. To empirically determine the most effective batch size, several experiments were conducted using different values for this hyperparameter. Three models were trained using SupConLoss,

with batch sizes of 39, 96, and 182. The performance of these models was evaluated on the source attribution task, chosen due to its increased complexity and stricter requirements on representation quality. These results are commented in Section 4.3.

- **Losses:** Two losses were used throughout the project (Euclidean distance loss and SupConLoss). Each of these presents distinct advantages and trade-offs in the context of contrastive learning. To better understand their impact, comparative experiments were conducted, the results of which are analyzed in detail in Section 4.3.
- **Euclidean Distance Margin:** To evaluate the impact of the margin hyperparameter in contrastive learning with Euclidean distance, a series of models were trained with margin values set to 1, 2, 5, 10, 50, and 100. As in the batch size experiments, the source attribution task was used as the evaluation benchmark due to its complexity. The results indicated no statistically significant differences in performance across the different margin values. Consequently, a margin of 1 was selected for subsequent experiments for consistency and simplicity.
- **Embedding Size:** This hyperparameter defines the dimensionality of the latent space, and thus the size of the image embeddings produced by the contrastive model. Models were trained with embedding sizes of 64, 128, 256, and 512 to explore its impact on performance. Similar to the experiments with the Euclidean distance margin, no statistically significant performance differences were observed across these configurations, except for 64, which significantly obtained worse results. Consequently, an embedding size of 128 was chosen for subsequent experiments. This decision was also motivated by computational considerations, as smaller embeddings allow for larger batch sizes, which is known to benefit contrastive learning.
- **Image Resolution:** As detailed in Section 3.3.1, all images were rescaled to a resolution of 256×256 pixels. Higher resolutions (512×512 pixels) were also tested; however, due to computational and GPU memory limitations, this required significantly smaller batch sizes. As a result, no improvement in performance was observed, and the 256×256 resolution was adopted for all subsequent experiments.
- **Embeddings Classifiers:** Four distinct embeddings classifiers were proposed in Section 3.2.2, being the one which best results obtained the KNN algorithm (used across all experiments done in Section 3.5.2 and 3.5.3. The results of these models are presented in Section 4.3.

Chapter 4

Results

In this section, we present the results of the experiments conducted in the project, as discussed in detail in Sections 3.5.2 and 3.5.3. Additionally, we include results from other studies to allow comprehensive comparisons between the proposed solution and existing approaches.

4.1 AI-generated detection problem

The models used to address the AI-generated image detection problem were those developed in *ES1*, *ES2*, and *ES3*. However, *ES1* and *ES2* are not directly comparable to other state-of-the-art (SotA) models, as they were trained on different datasets. All SotA models presented in Tables 4.1 and 4.2 were trained using the ForenSynths training set, which, as previously discussed, consists of real images and ProGAN-generated images. In contrast, the models from *ES1*, *ES2* were trained on alternative datasets (see Section 3.3.2 for details). *ES3* was trained with the same dataset as SotA models, hence being comparable.

Table 4.1 presents the AUC scores achieved by both the baselines and our proposed models. The baseline models were not individually tested on images generated by Stable Diffusion 1.4 and 1.5. Instead, they were evaluated on a combined set of images generated by various versions of the Stable Diffusion models. As a result, for SotA models, the reported performance was identical for both Stable Diffusion 1.4 and 1.5.

On the other hand, Table 4.2 presents the accuracy results obtained by both the baselines and our proposed models. In this evaluation, the baseline models were tested on the ForenSynths BigGAN test set, rather than the GenImage test set. Although these test sets were not identical, they were comparable, as both ForenSynths and GenImage were constructed following the class distribution of ImageNet. Consequently, they contained images covering a wide range of topics, luminance variations, and semantic content.

ES3 is the only model directly comparable to SotA approaches, as it was trained using the same dataset (the ForenSynths training set). In contrast, *ES1* and *ES2* were trained on different datasets to evaluate the feasibility of the proposed contrastive learning approach. Although *ES3* does not surpass SotA performance, the results provide valuable insights. One key factor is the use of the SupConLoss function, which relies on a diverse set of classes within

Table 4.1: Comparison of AUC between SotA models and ours in the AI-generated problem.

Model	ADM	BigGan	Glide	Midjourney	VQDM	SD 1.4	SD 1.5	Wukong
[41]	79.9	-	99.7	81.7	-	91.3[a]	91.3[a]	-
[89]	86.3	-	79.3	77	-	64.5[a]	64.5[a]	-
[90]	86.7	-	80.8	66.2	-	89.5[a]	89.5[a]	-
[49]	82.5	-	76.5	40.7	-	77.4[a]	77.4[a]	-
ES1	98.2	98.4	98.4	94.7	98.3	98.1	98.5	97.9
ES2	**99.8**	**99.7**	**99.8**	**99.6**	**99.8**	**99.8**	**99.8**	**99.8**
ES3	74.5	68.4	60.3	47.4	48.7	64.6	64.6	55.9

[a] Not the same test set present in GenImage dataset.

Table 4.2: Comparison of ACC between SotA models and ours in the AI-generated problem.

Model	ADM	BigGan	Glide	Midjourney	VQDM	SD 1.4	SD 1.5	Wukong
[90]	68.07	95.28[a]	64.01	57.35	85.17	61.15	62.99	71.32
[28]	76.36	67.12[a]	72.41	58.35	54.37	49.63	49.76	55.39
[43]	87.12	94.18[a]	96.32	65.07	95.59	97.98	97.18	55.39
[91]	61.44	82.05[a]	70.76	67.35	67.82	63.02	64.17	70.76
ES1	98.2	98.4	98.4	94.7	98.3	98.1	98.5	98.0
ES2	**99.8**	**99.7**	**99.8**	**99.6**	**99.8**	**99.8**	**99.8**	**99.8**
ES3	74.5	68.4	60.3	47.4	48.7	64.6	64.6	55.9

[a] Set used from ForenSynths dataset.

each batch to generate a larger number of negative samples. This helps produce embeddings with greater intra-class separation. However, *ES3* was trained with only two classes (ProGAN and Real), limiting the number of negative samples per batch and hindering classification performance. On the other hand, *ES1* and *ES2*, which were trained with more diverse classes, achieved significantly better results than current SotA models. Notably, *ES1* was trained exclusively on synthetic images, yet it is still capable of generating embeddings that accurately differentiate real images. SupConLoss function leverages the heterogeneity of labels within the training set, and more specifically, the batch. In the SotA benchmark, the models were trained solely on one generator and real images (2 classes), thus SupConLoss suffers from low diversity, and the model lacks enough information to effectively generalize to other image generators. when more classes were used for training (*ES1* and *ES2*), the model achieved great results, even for classes not seen during the training phase (real class in *ES1*). A more detailed discussion of these results and their implications is provided in Section 5.1.

4.2 Source attribution problem

In the source attribution problem, more classes were available for the model in the training set, hence better results were obtained leveraging the advantages of SupConLoss.

For each model, precision, recall, and F1-score were computed separately for the classes

included during training (non–zero-shot) and those excluded from training (zero-shot), as shown in Table 4.3. This approach allows for a direct evaluation of model performance on both seen and unseen classes. Table 4.3 summarizes these metrics, highlighting the differences in performance between zero-shot and non–zero-shot settings.

Table 4.3: Results of the trained models over zero-shot and non-zero-shot generators.

	Zero-shot			No Zero-shot		
Model	Precision	Recall	F1-Score	Precision	Recall	F1-Score
ES1	**91.2**	**83.4**	**87.1**	82.9	83.6	83.2
ES2	-	-	-	84.6	84.3	84.4
ES4.1	59.9	61.3	60.3	83.6	81.5	82.3
ES4.2	60.5	66.9	63.2	67.6	61.7	64.0
ES4.3	63.4	54.8	58.4	67.1	73.1	69.8
ES4.4	38.2	33.9	35.1	**88.7**	**95.9**	**91.6**
ES4.5	72.9	66.0	68.5	68.9	73.0	70.7

ES1 was trained using all generators from the GenImage dataset; therefore, only the "real" class is considered zero-shot. As observed in the AI-detection task discussed in the previous section, this model effectively distinguished real images from synthetic ones, despite not being exposed to any real images during training. In fact, it achieved a higher F1-score on the zero-shot class than on the non–zero-shot classes, outperforming them by 4%. These results suggest that while synthetic images may appear realistic to human observers, they remain distinguishable to a model that has been sufficiently trained on diverse synthetic data alone.

In contrast, *ES2* was trained not only with images from the same eight generators but also with real images. However, the performance gain over *ES1* on non–zero-shot classes is marginal (just 1.2% in F1-score). This indicates that when a model is exposed to a sufficiently diverse set of synthetic images from multiple sources, including real images in the training set does not provide significant additional value for distinguishing real from synthetic images. This result was also observed in the previous section, where *ES2* obtained only 2% higher accuracy compared to *ES1*, though *ES1* not being trained with any real image.

The remaining models were trained using five generators and evaluated on images from all generators, as well as on real images. The results varied depending on which generators were included during training. *ES4.1* achieved performance comparable to *ES1* and *ES2* on the non–zero-shot classes but showed a notable drop in performance on the zero-shot classes, with an F1-score 22% lower. It is worth noting that across all experiments, precision and recall remained consistently balanced.

ES4.2 performed similarly to *ES4.1* on the zero-shot classes but underperformed on the non–zero-shot classes. The results of *ES4.3* were close to those of *ES4.2*, showing slightly improved performance on the non–zero-shot classes but slightly worse performance on the zero-shot classes. Overall, *ES4.3* were outperformed by *ES4.1* across all metrics, suggesting that the set of generators used for training in *ES4.1* provided more informative and generalizable information than *ES4.3* ones.

The results for *ES4.5* are notably polarized: it outperforms even the *ES2* model on the

non–zero-shot classes, yet performs poorly on the zero-shot classes. This performance disparity is likely due to significant differences in the feature distributions between the generators used for training and those used for testing. Since the model was not exposed to any images from the test generators, the model failed to generalize and classify correctly not seen generators.

ES4.5 obtained stable results over the zero-shot and non-zero-shot classes (as *ES4.2*) but with better overall metrics, hence generators used in *ES4.2* provide more information and generalization capability than *ES4.5* ones. These results can provide insight into which are the most useful generators for training a contrastive models using SupConLoss. Moreover, these results can depict which generators produce more similar images and which ones produce dissimilar ones.

Table 4.4 depicts the precision obtained for each model over their respective zero-shot classes, while Table 4.5 presents the recall in those same classes. As already commented, depending on which classes were used to train the contrastive learning model, different metrics were obtained both in the zero-shot and non-zero-shot classes. Notably, some generators were used various times as a zero-shot class (e.g., Stable Diffusion 1.5 in *ES4.1*, *ES4.3* and *ES4.4*; BigGan in *ES4.1* and *ES4.2*, etc.). The precision and recall values are stable for each class across the experiments in which they were removed from the training set, although distinct generators were used in the training set. For instance, recall for Stable Diffusion 1.5 ranges from 26.7 in *ES4.4* to 41% in *ES4.3*. VQDMS's recall varies only 10% even though training sets in *ES4.2* and *ES4.3* shared only 3 generators (ADM, Stable Diffusion 1.4 and Wukong). The same applies to BigGan, whose precision and recall difference between *ES4.1* and *ES4.2* vary just 11% and 10.9% respectively, while using only three common generators in the training set (Wukong, Glide and ADM). The same observation is present in Wukong generator.

However, this tendency does not apply in the "Real" class. While the precision and recall metrics are stable from *ES4.2* to *ES4.5*, the metrics obtained in *ES4.1* are notably better than the others (more than 30% improvement in recall and 25% in precision). Moreover, its precision outperforms *ES1* and its recall is almost the same as *ES1*, which were trained with all generators.

Table 4.4: Comparison of precision obtained by each model over zero-shot generators.

Model	ADM	SD 1.4	SD 1.5	VQDM	Midjourney	Real	Glide	BigGan	Wukong
ES1	-	-	-	-	-	**91.2**	-	-	-
ES2	-	-	-	-	-	-	-	-	-
ES4.1	-	**31.9**	42.3	-	-	89.3	-	**82.2**	-
ES4.2	-	-	-	61.1	**51.3**	58.3	-	71.3	-
ES4.3	-	-	**50.5**	**61.4**	-	59.3	86.8	-	-
ES4.4	-	28.9	37.5	-	-	46.1	-	-	40.3
ES4.5	**68.3**	-	-	-	-	74.6	**92.9**	-	**55.6**

From Table 4.6 to Table 4.12 the confusion matrices for each model are presented. Albeit rich and meaningful metrics were provided previously (precision, recall and F1-score over zero-shot and non-zero-shot generators, as well as precision and recall for each generator when used as a zero-shot class), the confusion matrices offer additional insight into which generators were confused, and thus which generators produce similar images both semantically and visually. If images from different generators depicts similar features (both visual and semantically), the

Table 4.5: Comparison of recall obtained by each model over zero-shot generators.

Model	ADM	SD 1.4	SD 1.5	VQDM	Midjourney	Real	Glide	BigGan	Wukong
ES1	-	-	-	-	-	83.4	-	-	-
ES2	-	-	-	-	-	-	-	-	-
ES4.1	-	**45.2**	37.6	-	-	**88.3**	-	82.1	-
ES4.2	-	-	-	**68.8**	**53.6**	52.2	-	**93.0**	-
ES4.3	-	-	**41.1**	67.9	-	49.8	64.9	-	-
ES4.4	-	30.4	26.7	-	-	58.2	-	-	22.9
ES4.5	**80.7**	-	-	-	-	55.7	**89.5**	-	**38.0**

model extracts similar embeddings for them, hindering effective and accurate classification.

Table 4.6 presents the confusion matrix for the *ES1* model. Overall, the model demonstrates strong performance, correctly classifying nearly all instances across most classes. Misclassifications occur primarily between images generated by Stable Diffusion 1.4 and 1.5 (an expected result given the minimal differences between these two versions of Stable Diffusion). Consequently, the contrastive model produces similar embeddings for both, often confusing one for the other. Additionally, Wukong is occasionally misclassified with both Stable Diffusion models, suggesting that it shares characteristics with those models.

Interestingly, the "Real" class is frequently confused with Midjourney and vice versa, indicating that Midjourney-generated images are the most realistic. Notably, the contrastive model was not trained on any real images. Despite this, it achieves an accuracy of 83.3 in distinguishing real images from synthetic ones. Furthermore, if Stable Diffusion 1.4 and 1.5 were merged into a single class, the model's overall accuracy would increase to 95.6%.

Table 4.6: Confusion matrix of ES1.

		Predicted								
		ADM	BigGan	Glide	Midjourney	VQDM	Real	SD 1.4	SD 1.5	Wukong
True	ADM	5992	0	1	0	0	7	0	0	0
	BigGan	0	6000	0	0	0	0	0	0	0
	Glide	0	1	5989	0	0	10	0	0	0
	Midjourney	0	0	0	5652	0	346	1	1	0
	VQDM	0	0	0	0	5974	26	0	0	0
	Real	11	0	9	887	37	5005	7	4	40
	SD 1.4	0	0	1	0	0	19	2848	2870	262
	SD 1.5	0	0	0	0	0	25	3833	3852	290
	Wukong	0	0	0	0	0	50	259	183	5508

ES2 is the model that would be used commercially to effectively distinguish the source of AI-generated images, as it is trained with all known generators, as well as real images. The results are similar to those of the previous model in which no real images were used in the training set. Nevertheless, considerably more instances are correctly classified as "real" (the accuracy of "real" class raises from 83.3 to 98.3). However, there is a significant drop of accuracy in Wukong class, confusing it with Stable Diffusion 1.4 and 1.5. These last models are also confused between each other, providing more proof to the statement that the architecture and images generated by those generators are so similar that they cannot be distinguished.

In experiment *ES4.1*, all classes except BigGAN, Real, and the two Stable Diffusion models were used during training. The goal of this experiment is to evaluate whether a contrastive

Table 4.7: Confusion matrix of ES2.

		Predicted								
		ADM	BigGan	Glide	Midjourney	VQDM	Real	SD 1.4	SD 1.5	Wukong
True	ADM	5997	0	0	1	0	2	0	0	0
	BigGan	0	5998	2	0	0	0	0	0	0
	Glide	0	2	5994	0	1	2	1	0	0
	Midjourney	0	0	0	5929	0	57	0	2	12
	VQDM	0	1	0	1	5987	7	1	0	3
	Real	2	0	3	89	2	5895	2	1	6
	SD 1.4	0	0	1	4	2	1	2823	2694	475
	SD 1.5	0	0	0	8	0	1	3815	3635	541
	Wukong	0	0	1	4	3	7	592	444	4949

learning approach can generalize to distinguish between two highly similar generators (Stable Diffusion 1.4 and 1.5) that were not included in the training set. The resulting confusion matrix is shown in Table 4.9.

The model shows minor confusion between BigGAN and Glide, as well as between Real and Midjourney. These results suggest that BigGAN-generated images share the greatest similarities with those produced by Glide. A similar conclusion can be obtained for Midjourney, whose outputs are often confused with real images (indicating that it produces the most photorealistic results). As in *ES1* and *ES2*, Stable Diffusion 1.4 and 1.5 are frequently misclassified as one another, reaffirming their high similarity. Interestingly, in this experiment, samples from Wukong are often misclassified as belonging to either Stable Diffusion model, despite Wukong being included in the training set. This outcome is insightful: incorporating Stable Diffusion 1.4 and 1.5 into the training data does not significantly improve the model's ability to distinguish between them, but it does enhance classification performance for the Wukong generator.

This suggests that Wukong and the two Stable Diffusion models produce visually similar images. Therefore, including only one of these generators in the training set does not provide the model with enough information to differentiate them effectively. Notably, Wukong is more frequently confused with the Stable Diffusion models than the opposite (although Wukong was used for training), indicating a stronger overlap from Wukong toward Stable Diffusion models than vice versa.

Table 4.8: Confusion matrix of ES4.1.

		Predicted								
		ADM	BigGan	Glide	Midjourney	VQDM	Real	SD 1.4	SD 1.5	Wukong
True	ADM	5980	0	0	0	0	20	0	0	0
	BigGan	0	4927	1060	0	0	13	0	0	0
	Glide	0	1044	4949	0	0	7	0	0	0
	Midjourney	0	0	0	5574	0	421	2	3	0
	VQDM	0	0	0	0	5982	15	0	3	0
	Real	7	21	1	615	12	5298	16	24	6
	SD 1.4	0	0	0	0	0	63	2712	2245	980
	SD 1.5	0	0	0	0	0	79	3591	3005	1325
	Wukong	0	0	0	0	0	16	2195	1830	1959

Glide, Midjourney, VQDM and the Real class were the classes removed from the training set in *ES4.2*. The aim of this experiment was to assess whether training solely with diffusion models, the obtained model can effectively classify images produced by GANs. Its results are

presented in 4.9.

As in the confusion matrix of *ES4.1*, Glide is confused with BigGan. In *ES4.1* BigGan was not used for training and was confused with Glide, while in *ES4.2* Glide was not used for training and confused with BigGan, hence there exist similarities between the synthetic images of those generators. Midjourney and Real are both confused between each other, as in other experiments occurred. Nevertheless, VQDM was also classified as Real and Midjourney images incorrectly in several occasions, thus VQDM images are also realistic. The Stable Diffusion models were confused again with each other, with similar metrics than in previous conducted experiments. Wukong instances, though present in the training set, are also classified in numerous occasions as Stable Diffusion 1.4 or 1.5. Regarding the results of Tables 4.4 and 4.5, as well as the confusion matrices, *ES4.1* outperforms *ES4.2*, hence the generators split present in *ES4.1* offers richer and more valuable information than the ones in *ES4.2*.

Table 4.9: Confusion matrix of ES4.2.

		Predicted								
		ADM	BigGan	Glide	Midjourney	VQDM	Real	SD 1.4	SD 1.5	Wukong
True	ADM	5963	0	0	18	6	13	0	0	0
	BigGan	0	5581	409	0	9	1	0	0	0
	Glide	0	2169	3731	0	85	15	0	0	0
	Midjourney	8	0	0	3213	664	788	439	344	544
	VQDM	3	68	109	511	4128	821	136	100	124
	Real	18	11	13	1164	1017	3133	67	53	524
	SD 1.4	0	0	0	365	262	77	2634	2348	314
	SD 1.5	0	0	0	493	389	89	3468	3178	383
	Wukong	0	0	0	496	197	438	365	268	4236

Results obtained in *ES4.3* (presented in Table 4.10) are similar to the ones obtained in *ES4.1*. Glide and Midjourney are used both again just in the test set, hence they were again confused with BigGan and Real respectively. Even though VQDM was present in the training set, Real and Midjourney were confused with it, as occurred in *ES4.2*. These results along with the ones from Table 4.9 supports the idea that Midjourney, followed by VQDM, are the most realistic image generators, as their synthetic images are often classified as real images. Albeit Stable Diffusion 1.4 was present in the training set, the model was not able to produce embeddings different enough for Stable Diffusion 1.4 and 1.5 images to be correctly classified. In this case, Wukong was not misclassified as often with the Stable Diffusion models. Regarding the Stable Diffusion models and Wukong, *ES4.3* obtained better results than*ES4.2*, even though it did not use Stable Diffusion 1.5 in the training set while *ES4.2* used both Stable Diffusion models and Wukong. These results resembles the fact that Stable Diffusion 1.5 is not useful for the model, hence it should not be used for training.

In Table 4.11 the confusion matrix of the experiment *ES4.4* is depicted, where Wukong, Real and both Stable Diffusion generators were removed from the training set. In this case, the zero-shot metrics are extremely poor. All the zero-shot generators were confused with each other. Furthermore, they were confused with Midjourney, which was present in the training set. The generators used in *ES4.1* for training are the same than in *ES4.4* except that in the former Wukong is included, while in the latter Wukong is changed by BigGan. This fact produces huge differences but not in the Wukong or BigGan class, but in the Real class. *ES4.1* manages to classify correctly most of real images (see Table 4.8), while *ES4.4* failed to

Table 4.10: Confusion matrix of ES4.3.

		Predicted								
		ADM	BigGan	Glide	Midjourney	VQDM	Real	SD 1.4	SD 1.5	Wukong
True	ADM	5969	1	0	23	1	6	0	0	0
	BigGan	0	5435	546	0	15	4	0	0	0
	Glide	0	1991	3897	0	105	7	0	0	0
	Midjourney	7	0	0	3240	659	744	443	435	472
	VQDM	3	155	42	508	4073	757	181	142	139
	Real	19	17	5	1230	1116	2985	98	90	440
	SD 1.4	0	0	0	328	201	53	2849	2351	218
	SD 1.5	0	0	0	441	298	66	3635	3286	274
	Wukong	0	0	0	465	170	413	292	210	4450

classify them correctly. Furthermore, the recall of Wukong is stable across both experiments, but its precision is much better in *ES4.1*, where it was used in the training set. In *ES4.1* the generators which produced the more misclassifications were Stable Diffusion 1.4 and 1.5 with each other, which up to some point is reasonable. Nevertheless, in *ES4.4*, all the zero-shot generators were confused with each other, and even Midjourney was also confused with them, though it was used in the training phase.

Table 4.11: Confusion matrix of ES4.4.

		Predicted								
		ADM	BigGan	Glide	Midjourney	VQDM	Real	SD 1.4	SD 1.5	Wukong
True	ADM	5981	0	0	0	0	18	0	0	1
	BigGan	0	6000	0	0	0	0	0	0	0
	Glide	0	0	5980	0	1	14	0	0	5
	Midjourney	0	0	0	5074	0	310	255	234	127
	VQDM	0	0	0	0	5965	33	1	0	1
	Real	12	2	12	741	27	3492	654	622	438
	SD 1.4	0	0	7	829	13	1090	1821	1620	620
	SD 1.5	0	0	5	1187	12	1434	2382	2139	841
	Wukong	1	0	12	1147	17	1181	1184	1086	1372

Table 4.12 depicts the results of the experiment in which ADM, Glide, Wukong and Real were removed from the training set (*ES4.5*). The results of this experiment are far from straightforward. ADM was slightly confused with all other generators, while Stable Diffusion 1.4 and 1.5 were confused with each other as in other experiments, along with Wukong. Even though Wukong was removed from the training set, it obtained better results than in *ES4.4*, where Wukong was also removed from the training set. The difference between *ES4.4* and *ES4.5* resides in the fact that both Stable Diffusion models were removed from the training set in the former one. The aftermath of it is that to effectively distinguish Wukong from the Stable Diffusion models, it is necessary to train with Wukong and at least one Stable Diffusion model. If both Stable Diffusion models are considered the same, then with this approach it is impossible to correctly distinguish Wukong samples from Stable Diffusion ones without training with both generators. In *ES4.2* and *ES4.3* Wukong and at least one Stable Diffusion model was present in the training set, and it was only in those models where reasonable good metrics were obtained regarding the Stable Diffusion and Wukong generators. On the other hand, *ES4.5* confuses totally VQDM with Midjourney, although both of them were present in the training set.

The conclusions of these zero-shot experiments shows that the inclusion or removal of one generator in the training set does affect not only the performance of that generator itself,

Table 4.12: Confusion matrix of ES4.5.

		Predicted								
		ADM	BigGan	Glide	Midjourney	VQDM	Real	SD 1.4	SD 1.5	Wukong
True	ADM	4839	2	271	127	69	602	28	26	36
	BigGan	0	5933	67	0	0	0	0	0	0
	Glide	165	421	5371	2	11	16	1	4	9
	Midjourney	88	0	1	5563	0	334	2	2	10
	VQDM	81	0	6	5889	24	0	0	0	0
	Real	1571	1	47	872	58	3342	26	26	57
	SD 1.4	67	0	7	4	0	31	2690	2471	730
	SD 1.5	100	0	6	11	0	46	3563	3297	977
	Wukong	175	0	5	27	2	84	1847	1582	2278

but also the performance of other generators (e.g., inclusion of one Stable Diffusion model to effectively distinguish the Wukong class, inclusion of BigGan or Wukong in the training set has paramount influence in the metrics of the Real class, etc.).

4.3 Ablations

A series of hyperparameters had to be set to ensure consistency and comparability across all contrastive learning experiments. The first key hyperparameter was image resolution. As shown in Table 3.2, image resolutions varied significantly depending on their source. Since deep learning models require uniform input dimensions, a common resolution had to be established. In other papers in the literature, a resolution of 256×256 pixels was chosen, which strikes a balance between information retention and computational efficiency. Lower resolutions lead to a loss of visual details, while higher resolutions substantially increase computational cost and memory usage. Given the tight computational constraints, particularly in terms of available VRAM, the 256×256 resolution offered a suitable trade-off between resource consumption and the preservation of relevant image features.

The embedding dimensionality was another critical parameter. Four different sizes (64, 128, 256, and 512 dimensions) were evaluated. While no substantial differences in model performance were observed among the 128, 256, and 512 dimensional embeddings, the 64 dimensional variant exhibited a slight performance degradation. Consequently, 128 dimensions was selected as the optimal embedding size, providing a balanced trade-off between computational cost and representational capacity. This dimensionality was sufficient to support both AI-generated image detection and source attribution tasks effectively.

As discussed in previous sections, batch size is a fundamental hyperparameter in contrastive learning frameworks. The presence of a larger number of negative samples per batch enhances the inter-cluster separation in the latent space, facilitating better discrimination and improved classification performance. To assess its impact, three batch sizes were tested: 39, 96, and 182, using the SupConLoss function. The respective confusion matrices are presented in Tables 4.13, 4.14, and 4.15.

The results do not show significant differences between batch sizes 39 and 96. Both models obtained similar results in all generators except in Glide, where the former predicts correctly 500 additional instances. Nevertheless, the latter outperformed the model trained with batch 39 in other generators. Their accuracy are almost identical, being 83.42% and 83.24%

respectively. In contrast, the model trained with batch size of 182 performed substantially better, as it improved the number of correct instances from the "Real" class, as well as instances from Glide. However, it performed slightly worse in the generators that were confused between each other (both Stable Diffusions and Wukong), though the final accuracy is higher, obtaining 84.29%. The batch size's value for the experiments ultimately selected was 182, following the general principle that larger batch sizes yield better contrastive learning outcomes. Additionally, larger batches result in fewer weight updates per epoch, leading to faster and more cost-effective training.

Table 4.13: Confusion matrix of the model trained with batch size 39.

		Predicted								
		ADM	BigGan	Glide	Midjourney	VQDM	Real	SD 1.4	SD 1.5	Wukong
True	ADM	5988	0	0	0	0	12	0	0	0
	BigGan	0	5998	1	0	0	1	0	0	0
	Glide	1	0	5985	0	0	14	0	0	0
	Midjourney	0	0	0	5584	0	412	0	0	4
	VQDM	0	0	0	0	5983	15	0	0	2
	Real	10	2	7	639	25	5217	16	12	72
	SD 1.4	0	0	0	0	1	41	2873	2770	315
	SD 1.5	0	0	0	0	0	58	3648	3891	403
	Wukong	0	0	0	0	0	138	365	289	5208

Table 4.14: Confusion matrix of the model trained with batch size 96.

		Predicted								
		ADM	BigGan	Glide	Midjourney	VQDM	Real	SD 1.4	SD 1.5	Wukong
True	ADM	5985	0	0	0	0	15	0	0	0
	BigGan	0	5998	2	0	0	0	0	0	0
	Glide	1	3	5598	0	2	13	1	0	0
	Midjourney	0	0	0	5598	0	391	1	3	7
	VQDM	0	0	1	0	5971	27	0	0	1
	Real	14	0	11	687	28	5184	7	13	56
	SD 1.4	0	0	0	1	0	29	2999	2649	322
	SD 1.5	0	0	0	1	0	38	3584	3988	389
	Wukong	0	0	0	0	1	92	356	254	5297

Table 4.15: Confusion matrix of the model trained with batch size 182.

		Predicted								
		ADM	BigGan	Glide	Midjourney	VQDM	Real	SD 1.4	SD 1.5	Wukong
True	ADM	5997	0	0	1	0	2	0	0	0
	BigGan	0	5998	2	0	0	0	0	0	0
	Glide	0	2	5994	0	1	2	1	0	0
	Midjourney	0	0	0	5929	0	57	0	2	12
	VQDM	0	1	0	1	5987	7	1	0	3
	Real	2	0	3	89	2	5895	2	1	6
	SD 1.4	0	0	1	4	2	1	2823	2694	475
	SD 1.5	0	0	0	8	0	1	3815	3635	541
	Wukong	0	0	1	4	3	7	592	444	4949

As discussed in Section 3.4, two loss functions were evaluated during training: Euclidean distance-based contrastive loss and SupConLoss. The SupConLoss function outperformed its Euclidean counterpart, achieving an accuracy of 84.6% (see Table 4.6). Although multiple margin values were tested for the Euclidean Distance loss (1, 2, 5, 10, 50, and 100), no significant improvements in performance metrics were observed across the different settings.

This reinforces the conclusion that SupConLoss is better suited for this source attribution task, likely due to its effective handling of positive and negative pairs within a batch.

Another critical component influencing system performance is the embedding classifier, responsible for assigning a label to the extracted embeddings of input images. As discussed in Section 3.2.2, four different classification strategies were explored:

The first approximation involved computing the mean embedding (centroid) for each generator using the training data. A feedforward neural network was then trained to classify an input embedding by comparing it to each centroid. The generator with the highest predicted probability was selected as the output. This method outperformed KNN in non-zero-shot scenarios, but significantly underperformed in zero-shot settings, as the feedforward network had no exposure to embeddings from unseen generators.

In the second method, one representative image per generator was selected, and a feedforward network was trained like in the previous method, but changing the centroid of each generator for its representative. However, this method exhibited high variance in performance depending on the representative chosen and, like in the previous method, it failed to generalize well to unseen generators.

The third approach tested achieved better results when tested on classes not seen during the training phase. It consisted on computing the centroids for each generator, and the class of the instance to be classified would be the nearest centroid in the latent space. This is a cost-effective approach, as for each instance to be classified, N distances must be computed, being N the number of possible generators. This method achieved great results both in zero-shot and non-zero-shot classes. Nonetheless, all methods that extracts the centroids for each generator underperformed compared to other methods because the distances between the centroids of the clusters (inter-cluster distance) were very small, hence the centroids in the latent space were placed very close. Moreover, there are lots of images per generator (162,000), with different semantic meaning, visual features, etc. For that reason, averaging them all might mean several loss of information. Moreover, there might be sub-clusters statistically representative that, by averaging them in a common centroid, are lost, hindering the classification.

The best performance, particularly in zero-shot settings, was achieved using a K-Nearest Neighbors classifier. Unlike centroid-based methods, KNN considers the structure of the entire latent space, allowing it to capture subcluster level variations and better reflect local similarity relationships. Several values of k were tested (11, 21, 101, 1,001, and 10,001), with k = 11 yielding the highest accuracy and being selected as the final configuration. This approach demonstrated superior robustness and generalization, making it the most suitable classifier for this application.

4.4 Data feasibility analysis

In all the aforementioned experiments, embeddings for instances from each generator were obtained to enable classification using a KNN algorithm during inference. However, the entire set of training instances per generator was utilized (amounting to 162,000 samples for each)

which may not reflect realistic deployment scenarios. In practice, it is unlikely that such a large volume of data would be available for a newly encountered generator. To address this limitation, additional experiments were conducted to evaluate the minimum number of instances per generator required to maintain acceptable classification performance, ensuring that key metrics do not deteriorate significantly.

For this purpose, the model selected was the one obtained in *ES4.1*, as it achieved the highest performance metrics and demonstrated the greatest robustness. A total of 11 KNN classifiers were trained using varying numbers of instances per generator: 10, 20, 50, 100, 500, 1,000, 5,000, 10,000, 20,000, 50,000, and 100,000. The objective of these experiments was to determine the minimum number of samples that strikes a balance between classification performance and the practical constraints of real-world scenarios, where obtaining tens of thousands of examples for a newly encountered image generator is unlikely. It is important to note that all KNN models in Section 4.2 were trained using 11 neighbors. However, for the experiments involving only 10 and 20 samples, this number of neighbors was disproportionately high, and was therefore reduced to 3. For all other configurations, the number of neighbors was set back to 11, as this value yielded the best performance in the grid search discussed in Section 4.3.

To evaluate and compare the outcomes of these experiments, the same set of metrics used in Section 4.2 were computed. These include overall accuracy; mean recall, precision and F1-score for both zero-shot and non-zero-shot generators, as well as individual recall and precision for each zero-shot generator.

Table 4.16 presents the performance results of the KNN models trained with varying numbers of samples. The table includes precision, recall, and F1-score for each model, evaluated separately on zero-shot and non-zero-shot generators (using the same generator split described in *ES4.1*; see Section 3.3.2). It is important to note that the increments in sample size are not linear, and thus the results should not be interpreted as such. For a more visual representation, Figure 4.3 illustrates the same metrics as Table 4.16, divided into two subfigures: Figures 4.1 and 4.2. The former shows the performance metrics of the KNN models on the zero-shot generators, while the latter focuses on the non-zero-shot generators.

From the KNN model trained with the fewest samples per generator (10) to the one trained with the most (162k), there is a notable 8.6% improvement in F1-score, indicating a significant performance increase. However, the model trained with just 50 samples per generator achieves an F1-score only 3.7% lower than the model trained with 162k samples, suggesting a reasonable trade-off between sample size and performance. Interestingly, the model trained with 500 samples per generator marks the point where precision and recall converge for both zero-shot and non-zero-shot generators, resulting in a balanced F1-score. For models trained with fewer than 500 samples per generator, precision tends to be considerably higher than recall for zero-shot generators, while the opposite holds for non-zero-shot generators (precision is significantly lower). In contrast, models trained with more than 500 samples per generator show improved recall on zero-shot generators but experience a decline in recall performance on non-zero-shot generators.

Table 4.17 presents the precision and recall scores of the trained KNN models evaluated

Table 4.16: Results of the KNN models trained with different number of samples over zero-shot and non-zero-shot generators.

	Zero-shot			No Zero-shot		
Model	Precision	Recall	F1-Score	Precision	Recall	F1-Score
KNN-10	51.4	50.0	49.8	73.2	74.7	73.7
KNN-20	52.8	50.1	51.0	74.8	77.3	76.0
KNN-50	57.9	53.2	55.2	78.5	**81.8**	80.0
KNN-100	58.7	56.2	57.2	79.4	81.0	80.1
KNN-500	57.9	57.7	57.6	80.7	80.5	80.5
KNN-1k	57.8	59.0	58.1	81.2	79.7	80.3
KNN-5k	59.0	60.6	59.5	82.0	80.0	80.8
KNN-10k	58.9	60.7	59.7	82.5	80.2	81.1
KNN-20k	59.2	61.5	60.0	82.3	80.2	81.2
KNN-50k	59.4	**61.8**	**60.3**	83.4	80.6	81.8
KNN-100k	59.3	61.4	60.0	**83.7**	81.0	82.1
KNN-162k	**59.9**	61.3	**60.3**	83.6	81.5	**82.3**

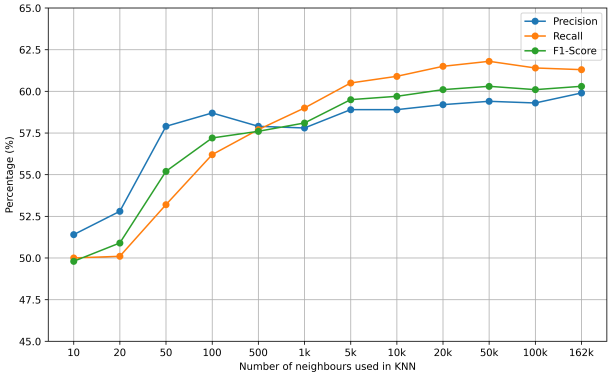


Figure 4.1: Comparison of KNNs' metrics over zero-shot generators.

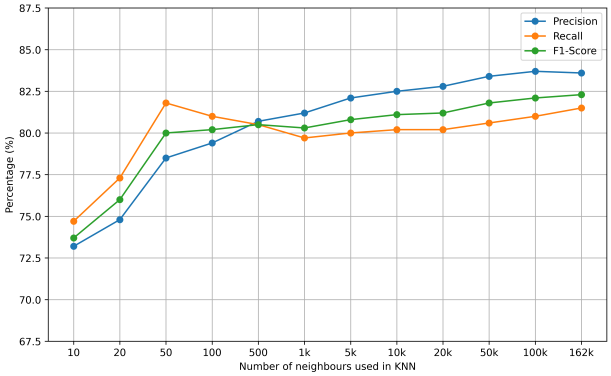


Figure 4.2: Comparison of KNNs' metrics over non-zero-shot generators.

Figure 4.3: Comparison of KNNs' metrics.

individually on each zero-shot class.

For Stable Diffusion 1.4, the differences across sample sizes are not significant and do not indicate that increasing the number of training samples consistently improves performance. For instance, the KNN model trained with only 10 samples achieves higher recall than the one trained with 162k samples, while their precision scores are nearly identical. Moreover, the performance metrics for this generator oscillate without a consistent pattern, indicating that the number of samples per generator used to train the KNN models has no significant impact on the final results for Stable Diffusion 1.4. A similar trend is observed for Stable Diffusion 1.5, where the model trained with just 100 samples per generator outperforms the model trained with 162k samples. These observations suggest that both Stable Diffusion models are particularly prone to misclassification, often being confused with each other and,

occasionally, with the Wukong class. These results are consistent with the findings presented in earlier sections, where Stable Diffusion models were frequently misclassified.

In contrast, for the remaining zero-shot classes (Real and BigGAN), both precision and recall generally improve as the number of training samples increases. Minor exceptions exist, for example, the Real class achieves slightly higher precision with the KNN-50 model, and the KNN-1k model shows a slight drop in recall for the BigGan class. However, these deviations are marginal, and the overall trend remains consistently upward.

Table 4.17: Comparison of accuracy obtained by each KNN model trained with different number of samples over zero-shot generators in *ES4.1* model.

	Zero-shot				No Zero-shot			
Model	SD 1.4	SD 1.5	Real	BigGan	SD 1.4	SD 1.5	Real	BigGan
KNN-10	29.2	40.0	83.1	57.4	46.0	27.1	73.3	61.1
KNN-20	29.2	40.6	83.7	61.8	39.4	29.0	75.4	63.8
KNN-50	29.8	42.1	**91.0**	73.8	32.7	33.2	77.8	75.9
KNN-100	30.6	41.4	88.4	80.4	31.9	**42.3**	82.1	74.1
KNN-500	31.0	40.3	87.1	79.0	40.3	35.1	85.4	77.7
KNN-1k	30.7	40.2	86.8	79.5	43.7	34.8	86.5	89.0
KNN-5k	31.5	42.0	86.8	81.1	44.4	38.2	87.0	80.1
KNN-10k	31.6	42.0	87.0	80.8	44.0	38.4	87.3	81.4
KNN-20k	32.2	42.1	87.4	80.6	**45.4**	38.6	87.6	82.0
KNN-50k	**31.9**	42.0	87.2	82.0	45.0	37.9	88.0	84.2
KNN-100k	31.3	41.7	87.5	**82.5**	44.7	36.9	88.1	**85.0**
KNN-162k	**31.9**	**42.3**	89.3	82.2	45.2	37.6	**88.3**	82.1

The most substantial improvement in accuracy occurs between the KNN model trained with 20 samples per generator and the one trained with 50, yielding a gain of 3.8%. Considering that the total gap between the lowest and highest observed accuracy is 8.9%, increasing the training set from 10 to 50 samples per generator results in a 5.3% improvement (equivalent to 59.66% of the total gap).

These results suggest that using 50 samples per generator (an amount that is realistically obtainable even from a newly developed generator) leads to only a 3.59% decrease in accuracy compared to the best-case scenario. While this drop may seem notable, it comes at a fraction of the data cost: only 50 samples per generator compared to 162,000, which represents just 0.03% of the total dataset. According to the elbow rule (a heuristic used to identify the point at which increasing a resource yields diminishing returns), 50 samples per generator offer an optimal trade-off, as further increases yield significantly smaller gains in accuracy, despite requiring larger increments in data (i.e., the steps between tested sample sizes are not linear).

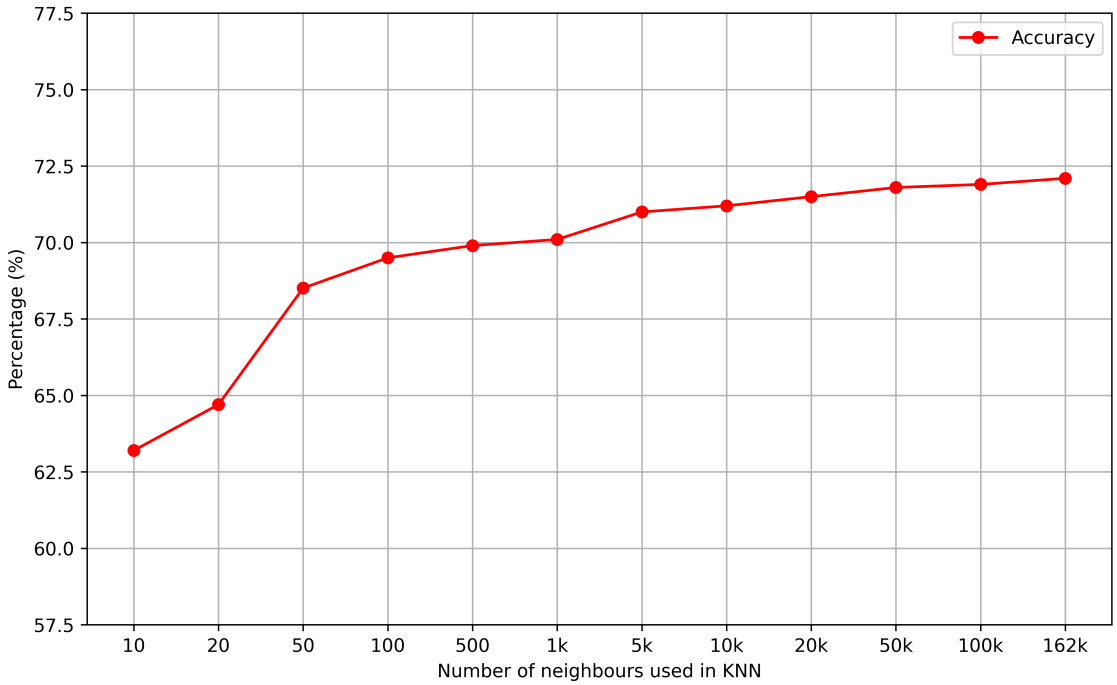


Figure 4.4: Comparison of KNNs' accuracies.

Chapter 5

Discussion

5.1 AI-generated detection problem

The proposed approach demonstrates an exceptional ability to distinguish between real and synthetic images, achieving near perfect precision when the training set includes a diverse range of image generators. However, this condition is not always met, as new AI image generators with unknown architectures are constantly emerging. Notably, our method can still effectively classify real and synthetic images even when no real images were seen during the training phase, as evidenced by the *ES1* experiment.

The success of our method lies primarily in the use of SupConLoss function. As discussed in the previous section, SupConLoss leverages the presence of multiple negative samples within each batch, promoting the separation of embeddings in the latent space (inter-cluster separation). This ability to enhance feature discrimination is crucial to the method's robustness. However, when the batch size is insufficient to include a variety of negative samples, or when class diversity is limited, the effectiveness of SupConLoss decreases. In scenarios involving only two classes (as in *ES3*) SupConLoss behaves similarly to traditional contrastive losses, such as those based on Euclidean Distance. This is because, in a binary classification context, SupConLoss lacks the supervised signal needed to differentiate between multiple negative classes, thus reducing its advantage.

Nonetheless, when a diverse set of synthetic generators is included in training, the benefits of SupConLoss are fully leveraged. In such settings, the network learns to produce distinct embeddings for images originating from different sources, leading to strong generalization capabilities. Remarkably, even without having seen real images during training, the model is able to extract discriminative features from them during inference, achieving an average precision exceeding 0.97% in distinguishing real from synthetic images. When not only a wide range of image generators were used to train the contrastive learning models (*ES2*), but also real images, precision of over 0.99% were achieved.

5.2 Source attribution problem

To effectively analyze the results obtained from the conducted experiments (see Section 3.3.2), the 128-dimensional embeddings extracted from the base images were reduced to two dimensions to allow for comprehensive visualization. The Uniform Manifold Approximation and Projection (UMAP) algorithm was employed for this purpose, as it is a widely used technique for dimensionality reduction, particularly suited for high-dimensional data such as ours. UMAP offers key advantages over other dimensionality reduction methods, notably its ability to preserve global data structure and its superior scalability to large datasets. These characteristics are critical in the context of source attribution, given the size of our dataset (over 1,296,000 images) and the need to retain the structure of the latent space in order to interpret intra and inter-cluster relationships accurately.

For consistency with the loss function, cosine distance was used as the metric in UMAP (the same one was used in SupConLoss during model training). Due to the substantial number of samples per generator, only a representative subset was visualized to facilitate computation and enhance interpretability.

Figure 5.1 illustrates the latent space derived from the model trained in experiment *ES2*, which is the most commercially viable model as it was trained with all generators and real images. In contrast, Figure 5.2 shows the latent spaces of all models trained using a zero-shot setting (*ES1*, *ES4.1*, *ES4.2*, *ES4.3*, *ES4.4*, and *ES4.5*).

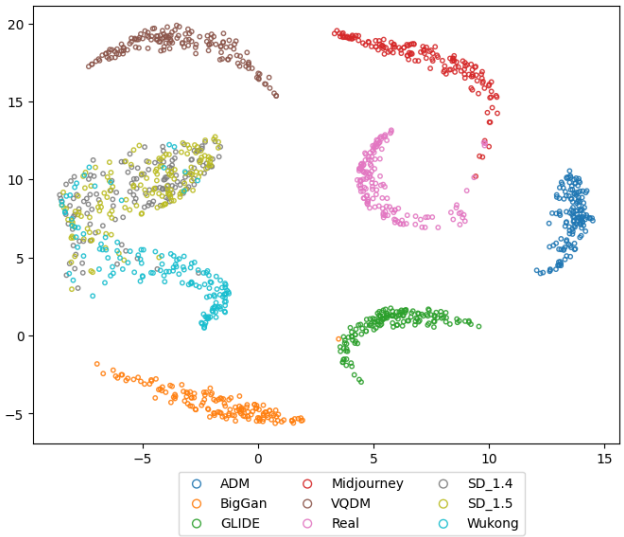


Figure 5.1: *ES2* latent space representation.

Figure 5.1 provides a visual representation of the latent space obtained from the model trained in experiment *ES2*, reduced from 128 to 2 dimensions using UMAP. This figure offers a deeper understanding of the results previously discussed. Notably, image generators such as BigGan and GLIDE, which are often confused with each other when either was excluded from the training set, they are close within the latent space. This spatial closeness suggests that these models produce images with highly similar visual characteristics.

Conversely, Midjourney and VQDM are the closest in the latent space to the "Real" class, indicating that these generators produce the most realistic images according to the model's learned representations. A particularly expected outcome is the significant overlap between Stable Diffusion 1.4 and 1.5 models, which may indicate that the visual output of these two versions is nearly indistinguishable, or that the current architecture, trained with SupConLoss, lacks the discriminative power to effectively separate them in the embedding space.

Wukong also shows partial overlap with the Stable Diffusion models, although the overlapping region is smaller. It is important to note that Figure 5.1 only reflects a two-dimensional projection of a 128-dimensional latent space, and thus, some structural information may have been lost during dimensionality reduction. The full latent space may reveal distinctions not visible in this projection.

Figure 5.2 displays the latent spaces from all experiments employing a zero-shot learning approach to the source attribution task, specifically experiments *ES1*, *ES4.1*, *ES4.2*, *ES4.3*, *ES4.4*, and *ES4.5*. Among these, the model trained in *ES1* achieves the best results, as the only unseen class during training is the "Real" class.

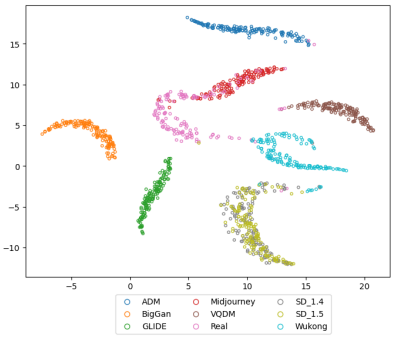
Similar patterns emerge here as in Figure 5.1: BigGan and GLIDE are located near each other in the latent space, while Midjourney and VQDM show substantial overlap with the "Real" class, reinforcing their visual realism. The Stable Diffusion models again exhibit considerable overlap, suggesting a high degree of similarity in their generated content. Interestingly, Wukong appears closer to the more realistic generators (Midjourney and VQDM) than to Stable Diffusion in this representation, with less overlap than observed in Figure 5.1.

As with the earlier figure, it is worth noting that if the Stable Diffusion variants were treated as a single class, classification accuracy would significantly improve. Overall, the results from *ES2* highlight the promising capability of the model to accurately distinguish real images from synthetic ones, even when trained exclusively on synthetic data.

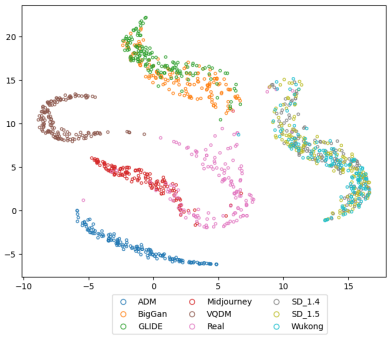
In the remaining experiments, a consistent trend is observed: BigGan and GLIDE tend to be closely located in the latent space (except in *ES4.4*) regardless of whether both were included in the training set. Midjourney also consistently appears in close proximity to the "Real" class across all experiments, followed by VQDM, which frequently overlaps with the real image cluster, suggesting their outputs are visually similar to real images according to the model's learned embeddings.

A notable case is ADM, which behaves distinctly compared to the other generators. When included in the training set (i.e., in *ES4.1*, *ES4.2*, *ES4.3*, and *ES4.4*), ADM is correctly classified. However, when excluded during training (*ES4.5*), it is often misclassified. Additionally, ADM's position in the latent space varies significantly across experiments: in *ES4.1*, it appears close to the "Real" and Midjourney clusters, whereas in *ES4.4*, it is located much further from them. In *ES4.2* and *ES4.3*, ADM is relatively isolated, appearing distant from all other cluster centroids. These observations suggest that the embedding learned for ADM is sensitive to the composition of the training set and lacks stability across different experimental settings.

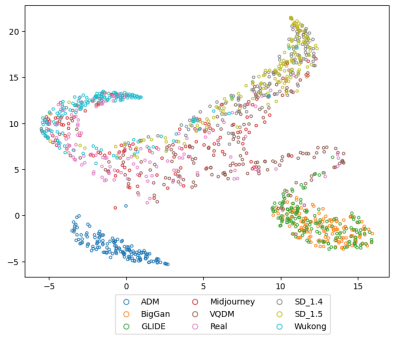
As previously discussed, the Stable Diffusion models consistently exhibit substantial overlap



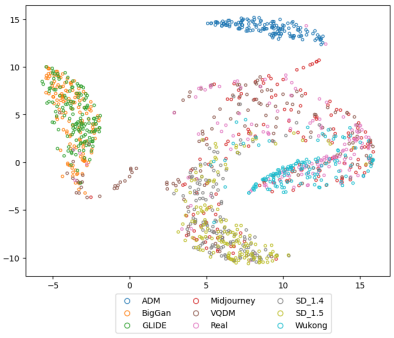
ES1 latent space representation.



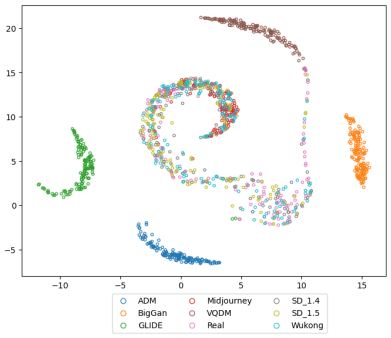
ES4.1 latent space representation.



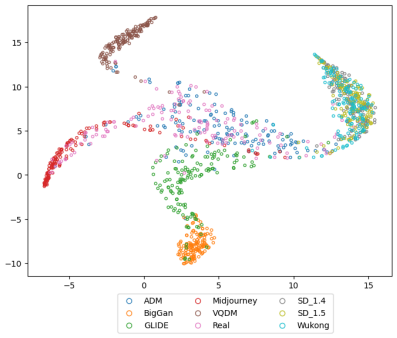
ES4.2 latent space representation.



ES4.3 latent space representation.



ES4.4 latent space representation.



ES4.5 latent space representation.

Figure 5.2: Latent spaces representations of models with zero-shot classes.

across all experiments, regardless of their presence in the training set. The Wukong dataset also frequently overlaps with the Stable Diffusion clusters, particularly in *ES4.1*, *ES4.4*, and *ES4.5*. However, in *ES4.2* and *ES4.3*, Wukong shows notable overlap with the "Real" class, indicating some inconsistency in how its images are embedded.

According to the evaluation metrics (see Table 4.3), the model trained in *ES4.1* achieves the best overall performance, followed by those in *ES4.4* and *ES4.5*. In contrast, *ES4.2* and *ES4.3* yield poorer results, characterized by extensive overlap between clusters (even for generators present in the training set) indicating weak separation in the latent space.

In *ES4.5*, despite ADM being frequently misclassified, improved inter-cluster separation is observed for other groups such as the "Real" class, GLIDE, BigGan, and VQDM. Still, as in other experiments, the Stable Diffusion and Wukong embeddings remain highly overlapping, making them hard to distinguish.

The latent space learned in *ES4.4* displays stronger separation for several generators (i.e., ADM, GLIDE, BigGan, and VQDM) each forming more clearly defined clusters. While Midjourney is occasionally misclassified, it still yields relatively good performance. However, the model fails to distinguish the remaining classes (Stable Diffusion 1.4 and 1.5, VQDM, and "Real") with high precision.

Ultimately, *ES4.1* demonstrates the most robust performance, particularly in classifying real images. Despite a slight overlap with Midjourney and VQDM, the "Real" class remains largely separable. Furthermore, this model achieves good separation for ADM, Midjourney, VQDM, and even for GLIDE and BigGan, despite the latter being excluded from the training set. Nevertheless, as in all other experiments, there is a complete overlap between Stable Diffusion models and Wukong, suggesting that their underlying architectures or outputs are so similar that the proposed method cannot differentiate them, hence they are close in the latent space.

To further interpret the behavior of the contrastive learning models, a series of Local Interpretable Model-agnostic Explanations (LIME) models were trained. The goal was to identify which regions of the input images contribute most to the embeddings learned by the model, and consequently, to the classification process. For consistency and to control for semantic variability, six images per generator were selected, all depicting a common domain (orange fish) ensuring a stable semantic meaning across experiments.

LIME operates by computing the embedding of a reference (anchor) image, then generating perturbed versions of the image (views) by slightly altering pixel values. These perturbed images are passed through the model to obtain their corresponding embeddings, which are then compared to the embedding of the original image. If the resulting embeddings diverge significantly, it indicates that the altered pixels play a substantial role in the model's representation. Conversely, if the embeddings remain relatively stable, the perturbed pixels are considered less influential in the learned representation.

One known limitation of LIME is its non-deterministic nature, as the generation of perturbed views is inherently stochastic. To address this and to ensure statistical robustness and pseudo-deterministic outputs, 10,000 perturbed views were generated for each image. This approach allows for a more accurate estimation of pixel importance by approximating the true distribution of features that influence the model's embedding process. The most influential pixels (those that most significantly impact the embeddings) are highlighted in the resulting images (see Figures 5.3–5.9).

Figure 5.3 illustrates the pixel importance obtained using the model trained in *ES2*. For images generated by ADM and BigGan, the model primarily focuses on the background to extract the embeddings. In contrast, for real images, it focuses not only in the background, but also on the fish themselves, particularly their bodies and edges. Midjourney images are also classified using the fish (predictable output, as Midjourney is the generator which produced the most realistic images, as commented in previous sections). Interestingly, for both Stable Diffusion models and Wukong, the model appears to utilize a combination of both fish highlights and background features. An interesting case is observed in the final Wukong example, where the model only highlights three specific fish out of a total of fourteen, indicating selective attention possibly based on positioning, lighting, or visual clarity.

A particularly insightful observation is that even when two generators produce embeddings that are spatially close in the latent space (e.g., BigGan and GLIDE, or Midjourney and "Real"), the model does not necessarily focus on the same regions of the image when extracting features. For instance, BigGan embeddings are predominantly derived from background pixels, whereas for GLIDE, the model focuses more on the fish itself. Similarly, while Midjourney embeddings

emphasize the fins of the fish, embeddings from real images rely more on background elements. These results suggest that proximity in latent space does not imply that the model attends to similar patterns or characteristics in the images.

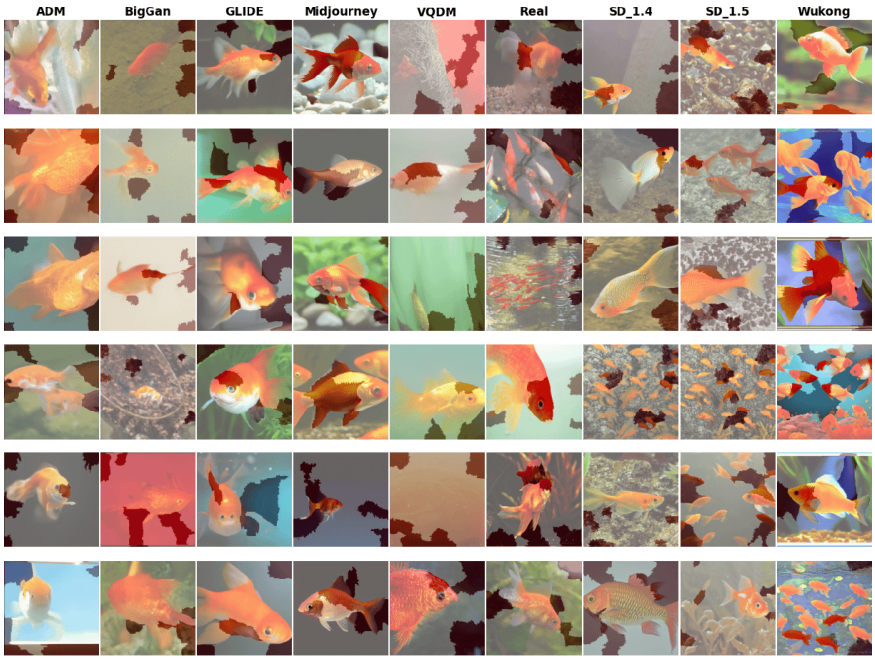


Figure 5.3: *ES2* LIME explanation.

Figure 5.4 presents the LIME outputs for anchor images corresponding to each generator, using the model trained in experiment *ES1*. A notable observation is that, for several generators, the model does not primarily focus on the fish, but instead places emphasis on the background regions of the image. For instance, in the case of ADM and BigGan, the highlighted regions predominantly lie in the background, suggesting that these areas contribute most significantly to the learned embeddings.

In contrast, for Midjourney, the model's attention is directed toward the fish themselves, particularly the rear fins, indicating a different embedding strategy. Similarly, GLIDE images show saliency concentrated on the upper body and lower fins of the fish, while the model's attention in VQDM images is again focused on the background, rather than the fish.

It is important to note that no real images were included during training in this experiment. Despite this, the LIME results show that the models attend to both semantic (fish) and non-semantic (background) regions, reinforcing the model's ability to generalize its learned representations beyond the training distribution.

Comparable results were observed in the LIME analysis for *ES2*, including those involving the "Real" class, even though it was not present in the training set in *ES1*. This finding further supports the conclusion that a contrastive model trained exclusively on synthetic data can still effectively extract semantically meaningful features from real images.

Figures 5.5 to 5.9 present the LIME outputs generated for each model trained with only five generators. These outputs differ more from those in Figures 5.4 and 5.3, which is expected

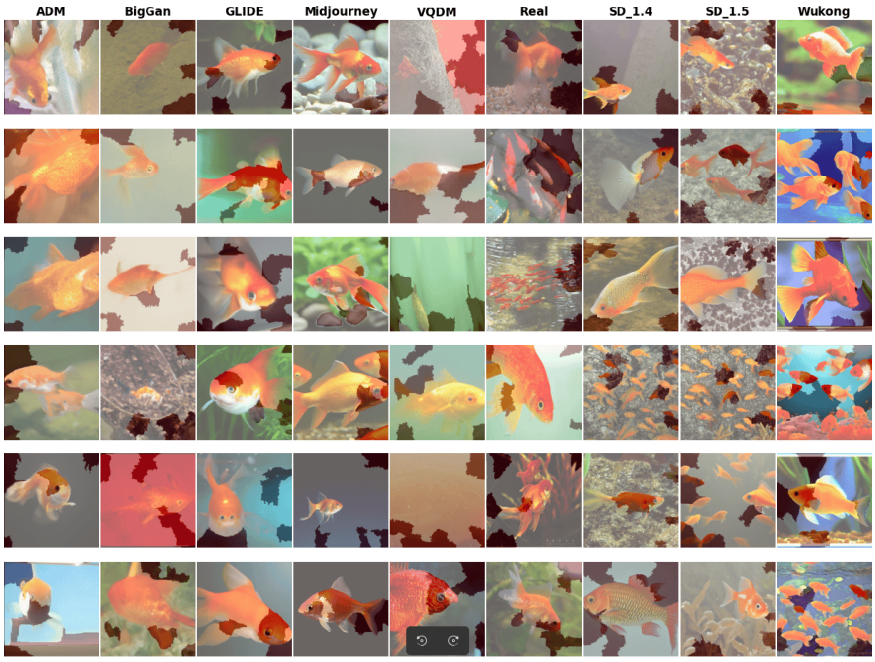


Figure 5.4: *ES1* LIME explanation.

given the significantly different training sets used. Even in cases where the training sets are nearly identical (such as in *ES4.1* and *ES4.4*, which differ by only a single generator) the highlighted regions can vary substantially. This indicates that the model's focus during embedding extraction is sensitive not only to the specific classes present but also to the overall composition and diversity of the training data (observation already made in previous sections).

Despite these differences, some consistent patterns emerge across all models. For instance, when generating embeddings for ADM and BigGan, the models consistently focus more on the background of the image than on the fish themselves. This trend aligns with the findings from earlier experiments, reinforcing the importance of background features in characterizing images from these generators.

An additional noteworthy observation is that no real images were included in the training sets of these experiments, yet the highlighted regions in the LIME explanations for real images remain remarkably consistent across all models. This further supports the generalization capability of contrastive models trained exclusively on synthetic data.

A particularly insightful case arises in *ES4.5*, the only experiment where ADM was excluded from the training set. In this setting, the model shifts its focus from the background to the fishes, a behavior not observed when ADM was part of the training data. This shift is accompanied by significant misclassification of ADM-generated images (see Table 4.12). From this, it can be inferred that effective classification of ADM images requires the model to focus on background features, suggesting that these elements are more discriminative than other content in this specific case.

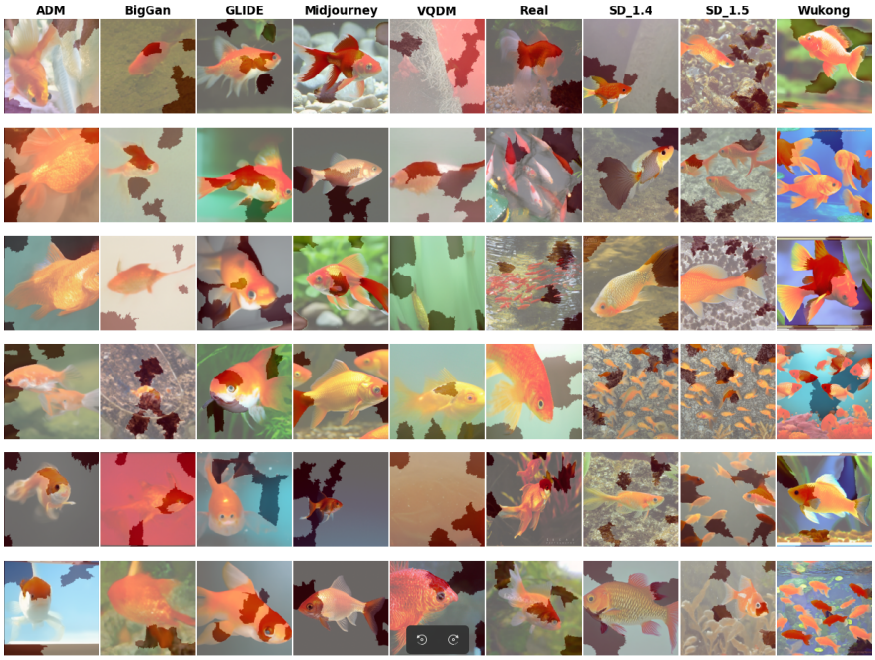


Figure 5.5: *ES4.1* LIME explanation.

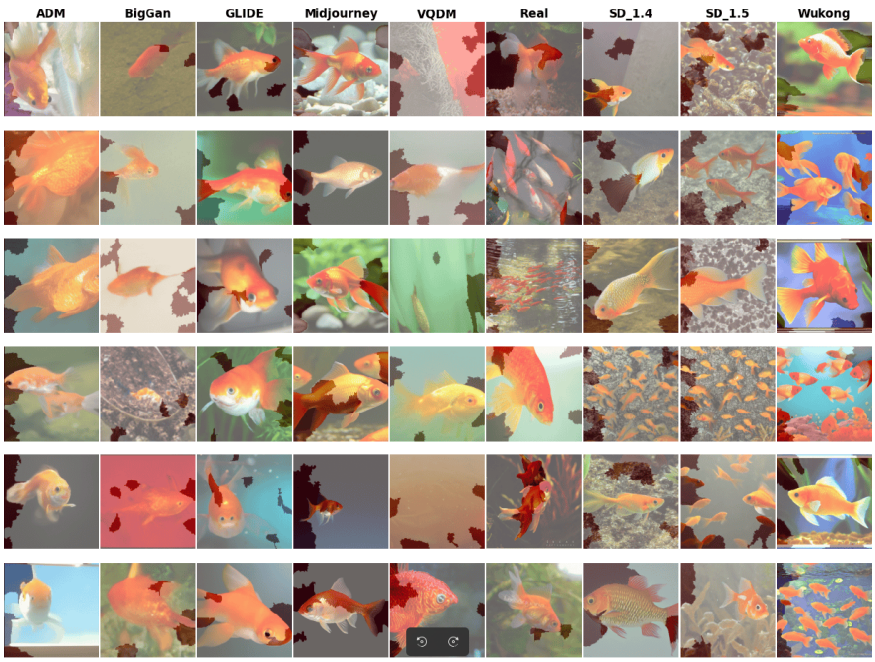


Figure 5.6: *ES4.2* LIME explanation.

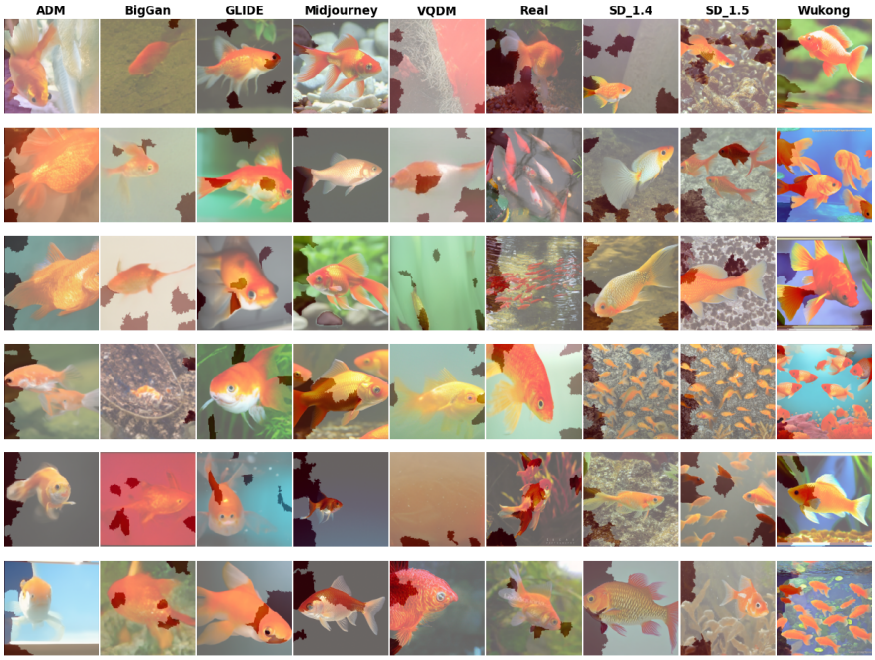


Figure 5.7: *ES4.3* LIME explanation.

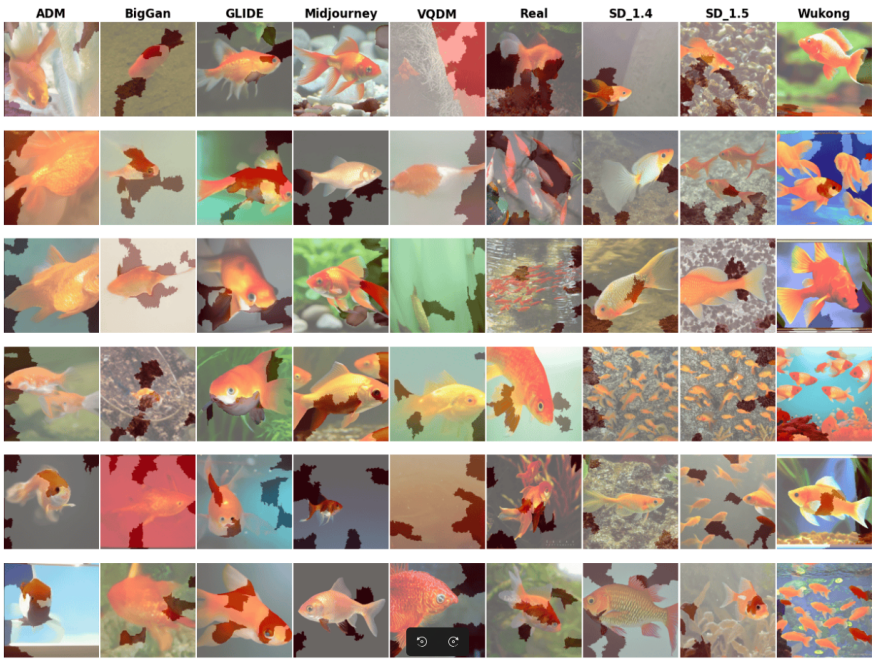


Figure 5.8: *ES4.4* LIME explanation.

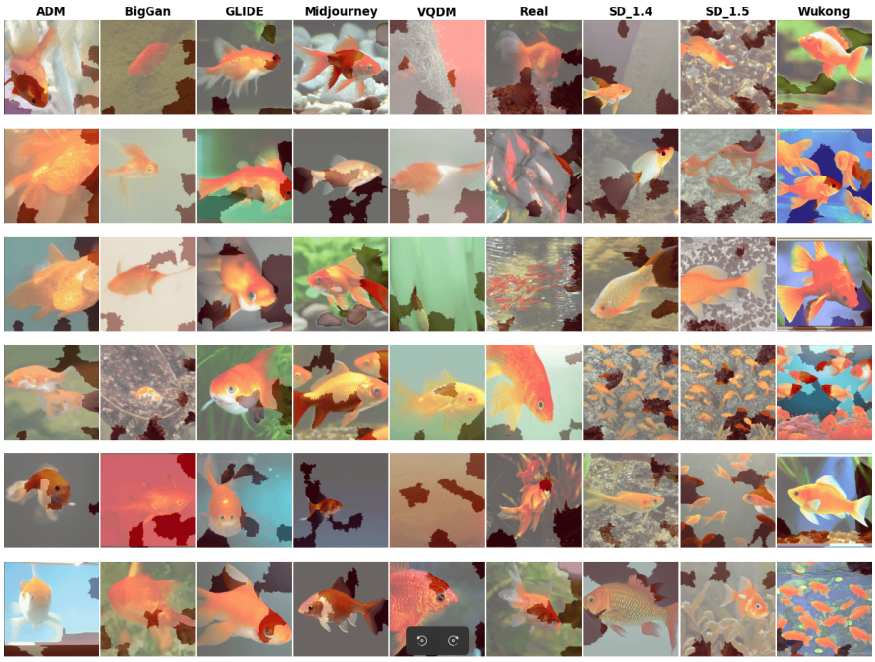


Figure 5.9: *ES4.5* LIME explanation.

Chapter 6

Conclusions

The approach proposed in this project consists in using contrastive learning techniques to train a model so that it is able to create rich and distinguishable embeddings of images generated by different image generators or real ones. Specifically, supervised contrastive learning was used as labels for each image were available (the label for each image was its generator). The reason behind the usage of supervised contrastive learning is to leverage the advantages of each method: contrastive learning encourages the model to separate embeddings in the latent space of the model of images whose generators differ and to pull together the ones belonging to the same generator. This enables to create rich and useful embeddings to then classify them with different ML methods (in this case a KNN was employed). On the other hand, the advantage of using supervised learning along with contrastive learning is the fact that different kind of negative samples are available to the model, as it is not the same to have a negative pair with labels "Midjourney" and "Stable Diffusion 1.4" than "ADM" and "Wukong". To implement the supervised contrastive loss, the SupConLoss was employed. Not only it leverages the fact that several negative pairs within the batch can be made, but also compares all positive pairs within it. For this reason, the bigger the batch size, the more negative and positive pairs there are, enabling more comparisons, hence more information for the model to produce better results and metrics.

Respecting the architecture of the backend network, a EfficientNetB0 was used. EfficientNet is a state-of-the-art CNN that achieves great results being also seven times smaller and 6 times faster than other CNNs. There existed tight constraints regarding VRAM, hence it was necessary to employ a cost-effective network.

Several experiments were conducted regarding the embeddings classifier, including feedforward network prediction with generators' centroids, feedforward network prediction using a single representative per generator, nearest distance to generators' centroids and KNN. The one that best results obtained was KNN with 11 neighbors, being also a fast and cost-effective algorithm that did not need any training previous to inference.

The models trained were used both for the AI-generated image detection and the source attribution problems. Moreover, they were used in a zero-shot approach, due to the fact that new and unknown image generators are developed at a great speed, being unfeasible to

retrain the classifier model each time a new image generator is released. Furthermore, several image generators are privative, and it is unfeasible to gather thousands of its samples.

Regarding the AI-generated image detection problem, when the detection model was trained solely with two classes (real images and images from one generator), the model was not able to generalize and effectively detect samples from other generators as fake images. The reason for this is that when only two classes are present in the training set, the supervised advantages of the SupConLoss function are lost (there is just one type of negative samples). Nonetheless, when more classes were introduced into the training set, great results were obtained. It was proved that a model trained only with synthetic images (from various generators) the model was able to distinguish fake from real image with an almost perfect accuracy (97.8% AUC).

On the other hand, the aforementioned architecture was also used in the source attribution problem. In this case, 5 generators were used for training, while the other 3 and the "Real" class were reserved for the test set. Different generators were included and excluded in different experiments so that a comprehensive comparison could be made. The results showed that depending on the classes used for training the models, different results were obtained. There were models which were usually confused with each other (i.e., Stable Diffusion 1.4 and 1.5 or Wukong). Interestingly, when a subset of those embeddings were redimensioned to 2 dimensions and plotted, it was observed that some models were always plotted near each other (BigGan with Glide, Midjourney with "Real"), meaning that those models produce similar patterns and artifacts. Moreover, a LIME model was trained for each contrastive learning, so that the most important pixels to extract the embeddings for some anchor images were highlighted. It was detected that for some generators, the relevant information was not in the figures themselves but in the background. Furthermore, it was seen that models whose embeddings are always close to each other in the latent space, the model uses the similar regions to extract their embeddings.

The model which best results obtained was the one trained with ADM, Glide, Midjourney, VQDM and Wukong (*ES4.1*), obtaining a F1-score of 60.3% in zero-shot generators (not included in the training set) and 82.3% in non-zero-shot generators (included in the training set). The model trained with all generators but real images (*ES1*) was also able to distinguish real images from synthetic ones (F1-score of the "Real class" was 87.1% and 83.2% for the non-zero-shot generators).

Some ablations were also made in this project, including the ones to set some hyperparameters, such as the dimensions of the embeddings or image resolutions. Furthermore, a series of experiments were made to assess whether SupConLoss behaved better than other distance-based losses, such as Euclidean Distance (with different margins). Some experiments worthwhile to further comment are the ones used to compare the results of the model based on the batch size used during the training phase. The results showed that using a batch size of 182 obtained 0.87% higher accuracy than using a batch size of 39. Batch sizes used in contrastive learning tend to be much larger (thousands of samples per batch). Nonetheless, due to to the strict constraints of VRAM, it was unfeasible to increase the batch size. Actually, the maximum batch size given the available VRAM was 182, which was the one finally used.

To be able to detect images from a generator not seen during the training phase it is necessary

to add some of its embeddings to the training set of the KNN. For this reason, different amount of number of embeddings per generator were tested, to assess the minimum number of embeddings per generator to obtain reasonable results. The results showed that using 162,000 images per generator obtained an accuracy of 72.1%, while using just 50 (a reasonable amount of images) obtained 68.51%, only 3.59% less though using 3,240 times less images. This result proves that this method is feasible, as 50 images can be easily gathered from a brand-new image generator.

6.1 Future work

Albeit the great results obtained in this project and the valuable insights provided, it is possible to further investigate this topic and possibly enhance the results. As commented in previous sections, there were tight constraints in respect to the available VRAM, using a suboptimal EfficientNet model, severe image rescaled and relatively small batch sizes. Given this facts, provided that more VRAM were available, it would be useful to assess whether larger batch sizes, images and architectures produce better results with the proposed method using supervised contrastive learning without any previous feature extraction (the EfficientNetB0 extracts its own features to produce the final embeddings).

Another research worth carrying out is the usage of hard negatives. Hard negatives are negatives pairs that the model confuses greatly (negative pairs whose embeddings are very near in the latent space). Some research states that by training models with lots of hard negatives, the results can be greatly enhanced. A possibility would be to do a warm up training with both of positive and negative pairs. Once the model understands the task, it is retrained with hard negatives, to encourage the model to separate those conflictive samples.

Bibliography

[1] A. M. Ille, C. Markosian, S. K. Burley, M. B. Mathews, R. Pasqualini, W. Arap, Generative artificial intelligence performs rudimentary structural biology modeling, Scientific Reports 14 (2024) 19372. doi:10.1038/s41598-024-69021-2.

[2] B. Chen, Z. Wu, R. Zhao, From fiction to fact: the growing role of generative ai in business and finance, Journal of Chinese Economic and Business Studies 21 (2023) 471–496. doi:10.1080/14765284.2023.2245279.

[3] M. Koohi-Moghadam, K. T. Bae, Generative ai in medical imaging: applications, challenges, and ethics, Journal of Medical Systems 47 (2023) 94. doi:10.1007/s10916-023-01987-4.

[4] J. Su, W. Yang, Unlocking the power of chatgpt: A framework for applying generative ai in education, ECNU Review of Education 6 (2023) 355–366. doi:10.1177/20965311231168423.

[5] Y. Singh, Q. A. Hathaway, B. J. Erickson, Generative ai in oncological imaging: Revolutionizing cancer detection and diagnosis, Oncotarget 15 (2024) 607. doi:10.18632/oncotarget.28640.

[6] P. S. Bist, H. Tayara, K. T. Chong, Generative ai in the advancement of viral therapeutics for predicting and targeting immune-evasive sars-cov-2 mutations, IEEE Journal of Biomedical and Health Informatics 28 (2024) 6974–6982. doi:10.1109/JBHI.2024.3432649.

[7] D. Allen, E. G. Weyl, The real dangers of generative ai, Journal of Democracy 35 (2024) 147–162. doi:10.1353/jod.2024.a915355.

[8] M. R. King, ChatGPT, A conversation on artificial intelligence, chatbots, and plagiarism in higher education, Cellular and molecular bioengineering 16 (2023) 1–2. doi:10.1007/s12195-022-00754-8.

[9] A. Katirai, N. Garcia, K. Ide, Y. Nakashima, A. Kishimoto, Situating the social issues of image generation models in the model life cycle: a sociotechnical approach, AI and Ethics (2024) 1–18. doi:10.1007/s43681-024-00517-3.

[10] E. Sunray, Sounds of science: Copyright infringement in ai music generator outputs, Catholic University Journal of Law and Technology 29 (2021) 185–218.

[11] X. Zhang, S. Dadkhah, A. G. Weismann, M. A. Kanaani, A. A. Ghorbani, Multimodal

fake news analysis based on image–text similarity, IEEE Transactions on Computational Social Systems 11 (2024) 959–972. doi:10.1109/TCSS.2023.3244068.

[12] T. Van Le, H. Phung, T. H. Nguyen, Q. Dao, N. N. Tran, A. Tran, Anti-dreambooth: Protecting users from personalized text-to-image synthesis, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023, pp. 2116–2127. doi:10.48550/arXiv.2303.15433.

[13] R. Liu, A. Khakzar, J. Gu, Q. Chen, P. Torr, F. Pizzati, Latent guard: a safety framework for text-to-image generation, in: European Conference on Computer Vision, Springer, 2024, pp. 93–109. doi:10.1007/978-3-031-73347-5_6.

[14] X. Yang, X. Wang, Q. Zhang, L. Petzold, W. Y. Wang, X. Zhao, D. Lin, Shadow alignment: The ease of subverting safely-aligned language models, arXiv preprint arXiv:2310.02949 (2023). doi:10.48550/arXiv.2310.02949.

[15] X. Li, Y. Yang, J. Deng, C. Yan, Y. Chen, X. Ji, W. Xu, Safegen: Mitigating unsafe content generation in text-to-image models, arXiv e-prints (2024) arXiv–2404. doi:10.48550/arXiv.2404.06666.

[16] M. Elasri, O. Elharrouss, S. Al-Maadeed, H. Tairi, Image generation: A review, Neural Processing Letters 54 (2022) 4609–4646. doi:10.1007/S11063-022-10777-X.

[17] D. Vela, A. Sharp, R. Zhang, T. Nguyen, A. Hoang, O. S. Pianykh, Temporal quality degradation in ai models, Scientific reports 12 (2022) 11654. doi:10.1038/s41598-022-15245-z.

[18] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, PMLR, 2019, pp. 6105–6114. doi:10.48550/arXiv.1905.11946.

[19] M. Zhu, H. Chen, Q. Yan, X. Huang, G. Lin, W. Li, Z. Tu, H. Hu, J. Hu, Y. Wang, Genimage: A million-scale benchmark for detecting ai-generated image, Advances in Neural Information Processing Systems 36 (2023) 77771–77782. doi:10.48550/arXiv.2306.08571.

[20] P. Dhariwal, A. Nichol, Diffusion models beat gans on image synthesis, Advances in neural information processing systems 34 (2021) 8780–8794. doi:10.48550/arXiv.2105.05233.

[21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-resolution image synthesis with latent diffusion models, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10684–10695. doi:10.48550/arXiv.2105.05233.

[22] S. Gu, D. Chen, J. Bao, F. Wen, B. Zhang, D. Chen, L. Yuan, B. Guo, Vector quantized diffusion model for text-to-image synthesis, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10696–10706. doi:10.1109/CVPR52688.2022.01043.

[23] Midjourney, https://www.midjourney.com/, Accessed: 2025-04-21.

[24] A. Q. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. Mcgrew, I. Sutskever, M. Chen, Glide: Towards photorealistic image generation and editing with text-guided diffusion models, in: International Conference on Machine Learning, PMLR, 2022, pp. 16784–16804. doi:10.48550/arXiv.2112.10741.

[25] A. Brock, J. Donahue, K. Simonyan, Large scale gan training for high fidelity natural image synthesis, arXiv preprint arXiv:1809.11096 (2018). doi:10.48550/arXiv.1809.11096.

[26] J. Gu, X. Meng, G. Lu, L. Hou, N. Minzhe, X. Liang, L. Yao, R. Huang, W. Zhang, X. Jiang, et al., Wukong: A 100 million large-scale chinese cross-modal pre-training benchmark, Advances in Neural Information Processing Systems 35 (2022) 26418–26431. doi:10.48550/arXiv.2202.06767.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

[28] Z. Wang, J. Bao, W. Zhou, W. Wang, H. Hu, H. Chen, H. Li, Dire for diffusion-generated image detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 22445–22455. doi:10.1109/ICCV51070.2023.02051.

[29] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Advances in neural information processing systems 33 (2020) 6840–6851. doi:10.48550/arXiv.2006.11239.

[30] A. Q. Nichol, P. Dhariwal, Improved denoising diffusion probabilistic models, in: International conference on machine learning, PMLR, 2021, pp. 8162–8171. doi:10.48550/arXiv.2102.09672.

[31] L. Liu, Y. Ren, Z. Lin, Z. Zhao, Pseudo numerical methods for diffusion models on manifolds, arXiv preprint arXiv:2202.09778 (2022). doi:10.48550/arXiv.2202.09778.

[32] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al., Photorealistic text-to-image diffusion models with deep language understanding, Advances in neural information processing systems 35 (2022) 36479–36494. doi:10.48550/arXiv.2205.11487.

[33] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, M. Chen, Hierarchical text-conditional image generation with clip latents, arXiv preprint arXiv:2204.06125 1 (2022) 3. doi:10.48550/arXiv.2204.06125.

[34] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, J. Xiao, Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop, arXiv preprint arXiv:1506.03365 (2015). doi:10.48550/arXiv.1506.03365.

[35] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of gans for improved quality, stability, and variation, arXiv preprint arXiv:1710.10196 (2017). doi:10.48550/arXiv.1710.10196.

[36] S. Yan, O. Li, J. Cai, Y. Hao, X. Jiang, Y. Hu, W. Xie, A sanity check for ai-generated

image detection, arXiv preprint arXiv:2406.19435 (2024). doi:10.48550/arXiv.2406.19435.

[37] ChatGPT, https://chatgpt.com/, Accessed: 2025-04-25.

[38] N. Keshtmand, R. Santos-Rodriguez, J. Lawry, Understanding the properties and limitations of contrastive learning for out-of-distribution detection, in: International Conference on Pattern Recognition, Springer, 2022, pp. 330–343. doi:10.1007/978-3-031-37660-3_23.

[39] P. H. Le-Khac, G. Healy, A. F. Smeaton, Contrastive representation learning: A framework and review, Ieee Access 8 (2020) 193907–193934. doi:10.1109/ACCESS.2020.3031549.

[40] D. Cozzolino, G. Poggi, M. Nießner, L. Verdoliva, Zero-shot detection of ai-generated images, in: European Conference on Computer Vision, Springer, 2024, pp. 54–72. doi:10.1007/978-3-031-72649-1_4.

[41] D. Cozzolino, G. Poggi, R. Corvi, M. Nießner, L. Verdoliva, Raising the bar of ai-generated image detection with clip, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2024, pp. 4356–4366. doi:10.1109/CVPRW63382.2024.00439.

[42] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al., Learning transferable visual models from natural language supervision, in: International conference on machine learning, PmLR, 2021, pp. 8748–8763. doi:10.48550/arXiv.2103.00020.

[43] J. Xu, Y. Yang, H. Fang, H. Liu, W. Zhang, Famsec: A few-shot-sample-based general ai-generated image detection method, IEEE Signal Processing Letters 32 (2025) 226–230. doi:10.1109/LSP.2024.3511421.

[44] Z. Sha, Y. Tan, M. Li, M. Backes, Y. 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, 2024, pp. 4852–4866. doi:10.1145/3658644.3690297.

[45] G. Cazenavette, A. Sud, T. Leung, B. 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, 2024, pp. 10759–10769. doi:10.1109/CVPR52733.2024.01023.

[46] Z. Meng, B. Peng, J. Dong, T. Tan, H. Cheng, Artifact feature purification for cross-domain detection of ai-generated images, Computer Vision and Image Understanding 247 (2024) 104078. doi:10.1016/j.cviu.2024.104078.

[47] J. J. Bird, A. Lotfi, Cifake: Image classification and explainable identification of ai-generated synthetic images, IEEE Access 12 (2024) 15642–15650. doi:10.1109/ACCESS.2024.3356122.

[48] D. C. Epstein, I. Jain, O. Wang, R. Zhang, Online detection of ai-generated images, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, 2023, pp. 382–392. doi:10.1109/ICCVW60793.2023.00045.

[49] B. Liu, F. Yang, X. Bi, B. Xiao, W. Li, X. Gao, Detecting generated images by real images, in: European Conference on Computer Vision, Springer, 2022, pp. 95–110. doi:10.1007/978-3-031-19781-9_6.

[50] S. Sinitsa, O. Fried, Deep image fingerprint: Towards low budget synthetic image detection and model lineage analysis, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2024, pp. 4067–4076. doi:10.1109/WACV57701.2024.00402.

[51] M. Mao, C. Yan, J. Wang, J. Yang, Leveraging pixel difference feature for deepfake detection, IEEE Transactions on Emerging Topics in Computational Intelligence (2025). doi:10.1109/TETCI.2025.3548803.

[52] N.-K. Ngo, X.-N. Cao, Pixel-wise information in fake image detection, in: International Conference on Future Data and Security Engineering, Springer, 2021, pp. 436–443. doi:10.1007/978-981-16-8062-5_30.

[53] F. Martin-Rodriguez, R. Garcia-Mojon, M. Fernandez-Barciela, Detection of ai-created images using pixel-wise feature extraction and convolutional neural networks, Sensors 23 (2023) 9037. doi:10.3390/s23229037.

[54] Y. Ju, S. Jia, J. Cai, H. Guan, S. Lyu, Glff: Global and local feature fusion for ai-synthesized image detection, IEEE Transactions on Multimedia 26 (2023) 4073–4085. doi:10.1109/TMM.2023.3313503.

[55] Z. Xue, X. Jiang, Q. Liu, Z. Wei, Global–local facial fusion based gan generated fake face detection, Sensors 23 (2023) 616. doi:10.3390/s23020616.

[56] Y. Ju, S. Jia, L. Ke, H. Xue, K. Nagano, S. Lyu, Fusing global and local features for generalized ai-synthesized image detection, in: 2022 IEEE International Conference on Image Processing (ICIP), IEEE, 2022, pp. 3465–3469. doi:10.1109/ICIP46576.2022.9897820.

[57] D. Zhang, J. Chen, X. Liao, F. Li, J. Chen, G. Yang, Face forgery detection via multi-feature fusion and local enhancement, IEEE Transactions on Circuits and Systems for Video Technology (2024). doi:10.1109/TCSVT.2024.3390945.

[58] R. Xu, A. Huang, Y. Hu, X. Feng, Gfft: Global-local feature fusion transformers for facial expression recognition in the wild, Image and Vision Computing 139 (2023) 104824. doi:10.1016/j.imavis.2023.104824.

[59] X. Kang, H. Yin, P. Duan, Global–local feature fusion network for visible–infrared vehicle detection, IEEE Geoscience and Remote Sensing Letters 21 (2024) 1–5. doi:10.1109/LGRS.2024.3375634.

[60] S. Zhu, L. Zhao, Q. Xiao, J. Ding, X. Li, Glffnet: Global–local feature fusion network for

high-resolution remote sensing image semantic segmentation, Remote Sensing 17 (2025) 1019. doi:10.3390/rs17061019.

[61] B. Abhisheka, S. K. Biswas, B. Purkayastha, Hbnet: An integrated approach for resolving class imbalance and global local feature fusion for accurate breast cancer classification, Neural Computing and Applications 36 (2024) 8455–8472. doi:10.1007/s00521-024-09541-0.

[62] Z. Sha, Z. Li, N. Yu, Y. 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, 2023, pp. 3418–3432. doi:10.1145/3576915.3616588.

[63] B. Khoo, R. C.-W. Phan, C.-H. Lim, Deepfake attribution: On the source identification of artificially generated images, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 12 (2022) e1438. doi:10.1002/widm.1438.

[64] T. Yang, Z. Huang, J. Cao, L. Li, X. Li, Deepfake network architecture attribution, Proceedings of the AAAI Conference on Artificial Intelligence 36 (2022) 4662–4670. doi:10.1609/aaai.v36i4.20391.

[65] T. Arora, R. Soni, Chapter 6 - a review of techniques to detect the gan-generated fake images, in: A. Solanki, A. Nayyar, M. Naved (Eds.), Generative Adversarial Networks for Image-to-Image Translation, Academic Press, 2021, pp. 125–159. doi:https://doi.org/10.1016/B978-0-12-823519-5.00004-X.

[66] N. Yu, L. S. Davis, M. Fritz, Attributing fake images to gans: Learning and analyzing gan fingerprints, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 7556–7566. doi:10.1109/ICCV.2019.00765.

[67] S. Hirofumi, K. Fukuchi, Y. Akimoto, J. 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), IEEE, 2022, pp. 1–8. doi:10.1109/IJCNN55064.2022.9892704.

[68] J. Ricker, S. Damm, T. Holz, A. Fischer, Towards the detection of diffusion model deepfakes, arXiv preprint arXiv:2210.14571 (2022). doi:10.48550/arXiv.2210.14571.

[69] K. Xu, L. Zhang, J. Shi, Detecting image attribution for text-to-image diffusion models in rgb and beyond, arXiv preprint arXiv:2403.19653 (2024). doi:10.48550/arXiv.2403.19653.

[70] R. Corvi, D. Cozzolino, G. Zingarini, G. Poggi, K. Nagano, L. 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), IEEE, 2023, pp. 1–5. doi:10.1109/ICASSP49357.2023.10095167.

[71] J. Ke, L. Wang, J. Liu, J. Fu, Sfia: Toward a generalized semantic-agnostic method for fake image attribution, International Journal of Intelligent Systems 2024 (2024) 7950247. doi:10.1155/2024/7950247.

[72] X. He, Y. Zhou, B. Fan, B. Li, G. Zhu, F. Ding, Vlforgery face triad: Detection, localization and attribution via multimodal large language models, arXiv preprint arXiv:2503.06142 (2025). doi:10.48550/arXiv.2503.06142.

[73] Z. Xu, X. Zhang, R. Li, Z. Tang, Q. Huang, J. Zhang, Fakeshield: Explainable image forgery detection and localization via multi-modal large language models, arXiv preprint arXiv:2410.02761 (2024). doi:10.48550/arXiv.2410.02761.

[74] M. Keita, W. Hamidouche, H. B. Eutamene, A. Taleb-Ahmed, A. Hadid, Fidavl: Fake image detection and attribution using vision-language model, in: International Conference on Pattern Recognition, Springer, 2025, pp. 160–176. doi:10.1007/978-3-031-78305-0_11.

[75] H. Wu, J. Zhou, Iid-net: Image inpainting detection network via neural architecture search and attention, IEEE Transactions on Circuits and Systems for Video Technology 32 (2021) 1172–1185. doi:10.1109/TCSVT.2021.3075039.

[76] Y. Zhang, F. Ding, S. Kwong, G. Zhu, Feature pyramid network for diffusion-based image inpainting detection, Information Sciences 572 (2021) 29–42. doi:10.1016/j.ins.2021.04.042.

[77] Y. Li, L. Hu, L. Dong, H. Wu, J. Tian, J. Zhou, X. Li, Transformer-based image inpainting detection via label decoupling and constrained adversarial training, IEEE Transactions on Circuits and Systems for Video Technology 34 (2023) 1857–1872. doi:10.1109/TCSVT.2023.3299278.

[78] F. Xiao, Z. Zhang, Y. Yao, Ctnet: hybrid architecture based on cnn and transformer for image inpainting detection, Multimedia Systems 29 (2023) 3819–3832. doi:10.1007/s00530-023-01184-w.

[79] N. Kumar, T. Meenpal, Semantic segmentation-based image inpainting detection, in: Innovations in Electrical and Electronic Engineering: Proceedings of ICEEE 2020, Springer, 2020, pp. 665–677. doi:10.1007/978-981-15-4692-1_51.

[80] H. Liu, Z. Wan, W. Huang, Y. Song, X. Han, J. Liao, Pd-gan: Probabilistic diverse gan for image inpainting, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 9371–9381. doi:10.1109/CVPR46437.2021.00925.

[81] A. Lahiri, A. K. Jain, S. Agrawal, P. Mitra, P. K. Biswas, Prior guided gan based semantic inpainting, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 13696–13705. doi:10.1109/CVPR42600.2020.01371.

[82] M. A. Hedjazi, Y. Genc, Efficient texture-aware multi-gan for image inpainting, Knowledge-Based Systems 217 (2021) 106789. doi:10.1016/j.knosys.2021.106789.

[83] M. H. Givkashi, M. Hadipour, A. PariZanganeh, Z. Nabizadeh, N. Karimi, S. Samavi, Image inpainting using autoencoder and guided selection of predicted pixels, in: 2022 30th International Conference on Electrical Engineering (ICEE), IEEE, 2022, pp. 700–704. doi:10.1109/ICEE55646.2022.9827427.

[84] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, Y. Akbari, Image inpainting: A review, Neural Processing Letters 51 (2020) 2007–2028. doi:10.1007/s11063-019-10163-0.

[85] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, L. Van Gool, Repaint: Inpainting using denoising diffusion probabilistic models, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 11461–11471. doi:10.1109/CVPR52688.2022.01117.

[86] A. Grechka, G. Couairon, M. Cord, Gradpaint: Gradient-guided inpainting with diffusion models, Computer Vision and Image Understanding 240 (2024) 103928. doi:10.1016/j.cviu.2024.103928.

[87] C. Corneanu, R. Gadde, A. M. Martinez, Latentpaint: Image inpainting in latent space with diffusion models, in: Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2024, pp. 4334–4343. doi:10.1109/WACV57701.2024.00428.

[88] J. Pyo, Y. G. Rocha, A. Ghosh, K. Lee, G. In, T. Kuc, Object removal and inpainting from image using combined gans, in: 2020 20th International Conference on Control, Automation and Systems (ICCAS), IEEE, 2020, pp. 1116–1119. doi:10.23919/ICCAS50221.2020.9268330.

[89] C. Tan, Y. Zhao, S. Wei, G. Gu, P. Liu, Y. Wei, Rethinking the up-sampling operations in cnn-based generative network for generalizable deepfake detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 28130–28139. doi:10.48550/arXiv.2312.10461.

[90] U. Ojha, Y. Li, Y. J. Lee, Towards universal fake image detectors that generalize across generative models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 24480–24489. doi:10.1109/CVPR52729.2023.02345.

[91] C. Tan, Y. Zhao, S. Wei, G. Gu, Y. Wei, Learning on gradients: Generalized artifacts representation for gan-generated images detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 12105–12114. doi:10.1109/CVPR52729.2023.01165c.