SAPIENZA
UNIVERSITÀ DI ROMA

Enhancing Abnormality identification: Robust Out-Of-Distribution strategies for Deepfake Detection

Faculty of Information Engineering, computer science and statistics
Master's degree in Artificial Intelligence and Robotics

Fabrizio Casadei
ID number 1952529

Advisor
Prof. Irene Amerini

Co-Advisor
Dr. Luca Maiano

Academic Year 2023/2024

Thesis defended on 25 March 2024
in front of a Board of Examiners composed by:

Prof. Domenico Lembo (chairman)
Prof. Irene Amerini
Prof. Ioannis Chatzigiannakis
Prof. Comminiello
Prof. Marco Console
Prof. Federico Fusco
Prof. Gabriele Proietti Mattia

Enhancing Abnormality identification: Robust Out-Of-Distribution strategies for Deepfake Detection
Master's degree thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: casadei.1952529@studenti.uniroma1.it

Alla mia Famiglia,
Per il continuo supporto e per avermi dato la possibilità di intraprendere questo percorso.
Alla Prof.essa Irene Amerini,
Per la continua disponibilità e cordialità, l'aver svolto questo lavoro con lei è stata una bellissima esperienza che ricorderò sempre con grande gioia.
A Luca Maiano,
Per gli utili consigli e il continuo aiuto datomi lungo l'intera attività tesistica.
A mia Nonna Elena e ai miei Zii,
Per rendermi orgogliosi di ciò che ho portato avanti e per avermi sempre compreso e motivato.
A Matteo,
Amico presente, pronto a consigliarmi e sostenermi da sempre.
A Giulia
Da poco entrata nella mia vita, fin da subito aiutandomi, incorangianodomi e capendomi.
A tutti i miei Amici
Manuel, Alessandro, Massimiliano, Luigi, Marco, Renata, Chiara, Roberta, Alessia, Daniele, Giorgia, ... Per aver reso questo percorso più spensierato con la vostra presenza.

Abstract

Out-Of-Distribution (OOD) and Deepfake detection are two significant challenges in the field of computer vision and artificial intelligence. In this thesis, we propose a novel architecture for identifying OOD samples in the context of deepfake detection, addressing both these open problems simultaneously into a unique work pipeline.
Neural networks are trained under the closed-world assumption, which holds that the majority of the data they receive will be similar to the training set. Although it is inevitable to come across unknown areas in the dynamic and unpredictable terrain of real-world data. In these kinds of situations, out-of-distribution detection becomes critical as neural networks frequently perform poorly on unknown input. In crucial fields like home robotics and medicine, where even little errors can have disastrous outcomes due to OOD brittleness, this might result in large performance decreases. By giving AI systems the ability to recognize and manage data that deviates from their training distribution, OOD detection attempts to reduce the dangers that come with coming into interaction with OOD data in practical applications.
Furthermore, we explore the deepfake detection domain in the era of advanced AI technology, where the generation of synthetic data raises significant security concerns. Deepfake detection aims to detect altered media produced with advanced artificial intelligence methodologies like Deep Neural Networks (DNNs). These technologies pose real security risks across a range of domains, including identity theft, cyberbullying, privacy violations, and misinformation.
We propose novel architectures and strategies for OOD detection, employing the collaborative efforts of In-Distribution classifiers and Out-Of-Distribution detectors. Our study integrates Convolutional Neural Networks (CNN) and Vision Transformers (ViT), presenting two distinct architectures for OOD detection related to a common pipeline. The first exploits the image reconstruction capabilities of the CNN model, while the second integrates the attention estimation in the study.
We introduce an adaptation of CDDB, the Continual Deepfake Detection Benchmark, preprocessed to facilitate customization and enable the evaluation of OOD detection techniques. Through extensive experimentation, we validate the effectiveness of our approaches and compare them against existing baselines. Moreover, our methodology is tested on a popular OOD benchmark to give a wide comparison between state-of-the-art methods that are currently known in the field. Our models achieve promising results in the deepfake detection domain, and reach the benchmark top ranking levels in several configurations.

Contents

List of Figures

Chapter 1

Introduction

Intelligent recognition systems must be able to recognize and process unfamiliar inputs in order to operate consistently in the real world. The closed-world assumption is based on the idea that a neural network would primarily come across data that is similar to what it was trained on. This data is called In-distribution (ID) data. But what happens when real-world data represent unexplored regions from the model? That is the point at which out-of-distribution (OOD) detection becomes important. Neural networks can perform poorly when faced with OOD data. One of the direct effects is a significant drop in activities' performance that are carried out in real life. Furthermore, the consequences may be catastrophic in significant domains. For example, in medicine, an AI system with OOD weakness might misdiagnose a patient and prescribe the wrong treatment path. A practical example is [105] which highlights the limitations of existing OOD detection methods in 3D medical image segmentation. In figure 1.1 a representation of In-Distribution and OOD data for vestibular schwannoma segmentation task.

In a robotics scenario, a robot can misunderstand and interpret incorrectly an object or an instruction and do unexpected behaviors. Due to these real risks, it's critical to identify and manage OOD data with care.
In autonomous driving, we would like the driving system to issue an alert and give the command to the human driver when it detects unusual scenes or objects that it has never seen during training time and cannot make a safe decision. Thus, OOD detection is used to describe the process of detecting samples or instances that don't belong to a machine learning model's training distribution. In other words, refers to the capacity of a model to identify and treat data that differs significantly from the one of its training set.

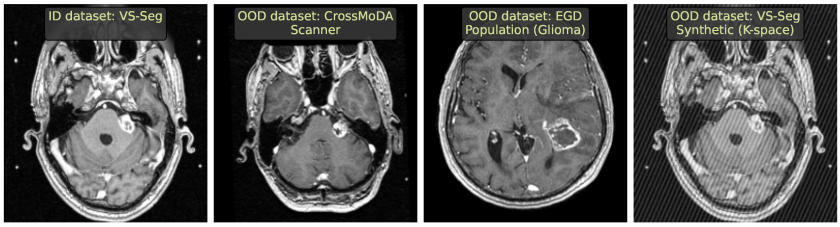

Figure 1.1. Examples of MRI images with ID and OOD sources.

The term, OOD detection, first emerged in 2017 and since then has received increasing attention from the research community. The advantages produced by a reliable OOD detector could be used to improve Deepfake detection performances, a task that received great interest in recent years. The integration of OOD detection system in discriminative models could be used to enhance adaptability, robustness, and effectiveness. For example, given the evolving nature of deepfake techniques, we can understand when re-training is required for new samples that would be wrongly interpreted by the model.
The term "deepfake" was first coined in late 2017 by a Reddit user of the same name. This user posted manipulated media content on Reddit, featuring celebrities with digitally swapped faces. The term itself is a fusion of the words "deep learning" and "fake", which were inspired by the user's account name. The formal definition of "deepfakes" is "AI-generated synthetic data" and denotes the fabrication of data by using deep learning algorithms.
Nowadays, remarkable advancements in Neural Networks and Machine Learning algorithms have been made possible by the quick development in the field of Computer Graphics, notably in GPU computational capacity and tensor processing. Using architectures like Generative Adversarial Networks [30] makes it simple to create and modify text, audio, video, and image files so that the human eye and ear cannot distinguish between authentic and fake content.
The production of synthetic data has several advantages, especially in the entertainment sector, where it is employed to modify and fix visual issues and speed up the generation of visual effects. Other fields that are beneficial to this technology are all the ones involved in visual content like e-commerce, social media management, assets designing, and so on.
However, serious security problems have also arisen as a result of the advances in AI-generated data, which are correlated to the diffusion of false information, lack of privacy, and unauthorized identity usage.
Apart from privacy-related issues like financial fraud and identity theft, social trust is also a potential problem. Misinformation campaigns and the dissemination of harmful content and false news may impact public opinion, particularly when it comes to popular people such as politicians or actors. Cyberbullying, revenge porn, deepfake porn, and blackmail are among the more famous malevolent uses of deepfake data.
In 2021 Korean newscaster Kim Joo-Ha was briefly replaced by a deepfake from MBN news channel, figure 1.2. It was unexpected when the company stated that it will keep using deepfake technology for its newsreaders. This gives rise to genuine worry about the technology use that could become popular in the future, and newsreaders may become obsolete as a job.
In 2019, the CEO off UK-based energy firm got a call from someone who sounded just like his chief executive. His "boss" ordered him to transfer $243,000 to a Hungarian supplier, and he fell into the fraud. Similarly, in January 2020 scammers stole $35 million from a Japanese company using a sophisticated voice clone.
in 2022 a deepfake of Mark Zuckerberg was produced by the progressive activist organization Demand Progress in an attempt to ridicule Democratic legislators for their incapacity and refusal to put controls on powerful digital companies like Facebook and Instagram.

Figure 1.2. Korean newscaster Kim Joo-Ha deepfake.

These events motivate the research community to focus on detection systems to limit the possibility of similar dangerous events, looking at solutions for well-known and new generative approaches.

1.1 Out-Of-Distribution & Deepfake Detection

The double nature of our work comprehends the development of novel techniques to perform Out-Of-Distribution detection in the context of deepfake image identification. This is traduced in two models that operate for the two tasks, one after the other. Both models deal with a binary classification problem, in which instances can be categorized as real or fake and as In or Out of distribution.
Therefore, the models act in a cascade, with the first model's output composed of the deepfake outcomes, and further extracted data serving as input for the OOD detection.
We now turn our attention to the two tasks separately, describing what they concern and how are modeled, starting from the deepfake detection.

Deepfake detection is one of the operations included in the digital media forensics field, which is used to analyze and verify the authenticity of digital content.
Digital media forensics not only analyzes images but also videos and audio sources, in order to find any sign of inconsistency and lack of integrity due to manipulation and tempering techniques applied to generate or modify media.
Different techniques to perform Deepfake detection exist, they vary on the type of data processed, the nature of exploited features, and the overall architecture of the model in action.
While some works use image-based algorithms, like in our case, much research has been done on video frame-based techniques and those that include both audio and video data. Although video-based approaches are often more powerful than image-based ones, their applicability is limited to specific kinds of assaults.
The key aspect to take into account is the nature of deepfake manipulation, since

several generative approaches can be used for the generation of fake content, like face swapping, face expression alteration (face reenactment), face retouching (attribute manipulation), entire face synthesis, non-face related synthesis, and so on.
Despite deepfake detection is not limited to faces, there are several reasons why face alteration gets so much attention in deepfake research and detection techniques. For example, the social and political impact of fake media is a serious risk that needs to be considered and addressed, to avoid scenarios of privacy violation and public defamation.
Getting mastery of new deepfake manipulations is one of the biggest obstacles to deepfake detection. The problem is called the generalization capability problem.
This is more clear if we think about how an effective deepfake detector incorporates a great degree of generalization to work well with all of the manipulation techniques exposed in precedence, as well as those that become accessible in the future.
The overall approach to this task can follow both unimodal and multimodal strategies. In the unimodal approach, techniques focus only on one type of content, image/video or audio. An ensemble approach is a useful tool for combining many unimodal models, such as those for audio or video. However, this approach takes into account each piece of information separately rather than analyzing their relationship.
Both audio and visual modalities are usually required for an important percentage of deepfake detection algorithms. Furthermore, one modality may not be available in real-world multimodal applications. Generally speaking, combining two different unimodal ways is more viable in these kinds of scenarios than using audiovisual detection methods.
It's also interesting to highlight how different clues can be used for the detection of real or fake samples, some of these require the use of both video and audio sources. Other techniques instead can be used for deepfake detection at the image level.
Deepfake detection methods that utilize spatial inconsistencies focus on analyzing visual artifacts, facial landmarks, and intra-frame inconsistencies within deepfake videos.
Exploiting convolutional traces detecting artifacts and traces left behind during the generation process of deepfakes, it's possible to discriminate fake media effectively.
Another common solution involves focusing on biological signs, this is done through the analysis of physiological signals such as eye blinking frequency, eye color, and heartbeat to identify manipulated content. Researchers focus on facial Emotion Alignment as well, detecting poor alignment of facial emotions on swapped faces, which is a clear sign of inconsistency.
Other inferences can be made on video sources by looking at temporal Inconsistencies and Audio-Visual Inconsistencies. The first represents discontinuity between adjacent video frames leading to flickering in most of the cases, while the latter is the multimodal approach expressed before that takes into account both visual and audio inconsistencies.
In the field of deepfake identification, a number of new technologies have been used, including media forensics-based methods and machine learning. Nevertheless, at the moment deep learning-based models are known to perform most effectively in differentiating between fake and real digital material.
This treatment and experimental development address the detection of deepfake images, and it's clear how this solution can be applied also to video, extending

the possibility of more comprehensive inferences like the ones described in precedence.

For what concern the Out-Of-Distribution detection, is important first to give a general definition, and then go deeper with the relative task sub-categories characterized by other distinctive aspects.
The goal of out-of-distribution detection is to identify test samples taken from a distribution that differs from the training distribution. The distribution is designed in a way that is specific to the intended application. In other words, this detection involves the recognizing of instances from a distribution different from the one seen during training.
Following the extremely new guidelines [119] is possible to distinguish between five different problems: Anomaly Detection (AD), Novelty Detection (ND), Open Set Recognition (OSR), Outlier Detection (OD), OOD detection.
These five problems can be seen as special cases or sub-tasks, all under the generalized OOD detection framework. All these types of detection share the goal of detecting Out-Of-Distribution examples under the open-world assumption, which assumes the possibility of testing the application on data coming from an unknown distribution. In contrast, we have the closed-world assumption that states the opposite and assumes the same distribution for test and training data.
The open-world assumption represents better a real-world scenario, in which the data distribution known regarding a certain domain can be incomplete, and we want performative models also in this condition.
All the differences between the presented sub-categories rotate around the concepts of *Covariate shift* and *Semantic Shift*.
When the label space $P(Y)$ stays constant while the marginal distribution of the input space $P(X)$ changes, it is referred to as covariate shift. This affects the features, and domain shifting, stylistic modifications. Adversarial examples are an example. Covariate shift is commonly used to evaluate model generalization and robustness performance.
Instead, Both the label space $P(Y)$ and the input space $P(X)$ are modified when the semantic shift occurs. The introduction of new categories or their modifications indicates a shift in the label space, which affects the type of data observed.
Moreover, we talk of *Semantic overlapping*, when In and Out-Of-Distribution data present samples with the same categories or classes.
Let's briefly describe now each sub-task that composes the **generalized OOD detection**:

- Anamoly detection, identification of anomalous samples respect predefined normality, either with covariate shift or semantic shift.
- Novelty detection, detect any samples not seen by the model during training. Only semantic shift is detected.
- Open Set Recognition, train the model with a subset dataset's classes and detect the unknown samples relative to the remaining classes (no semantic overlapping) exploiting the semantic shift.
  In terms of the detection problem, open set recognition is the same as multi-class novelty detection; the main distinction is that open set recognition

additionally needs ID categorization.

- Outlier detection, detect outlier data deviating from the norm in the training dataset. An outlier can be identified for both semantic shift and covariate shift.
- Out-Of-Distribution detection, detect distribution different from the training one. For most tasks with distribution, is intended the label distribution, so no semantic overlapping between ID and OOD, but some OOD detection works also take covariate shifts into account. These works assert that covariate shifts typically result in a significant decline in model performance and should be recognized and rejected.
  Therefore, this task aims to identify samples to which the model is not able to generalize.

Usually, Anomaly detection and Outlier detection are the only ones implicated in binary classification problems, even if this isn't always the case.
In addition, for both Anomaly detection and Novelty detection the ID classification training is not required. Training is not necessary for Outlier identification, but it is for Open Set recognition and OOD detection.
Different kinds of techniques are used to develop algorithms able to detect OOD samples: Bayesian models, Reconstruction-based models, Gradient-based methods, and so on. A more comprehensive explanation and illustration of these techniques is present in the following chapter 2.
There is a further empirical definition of OOD data based on the level of similarity to the In-Distribution data.
Regarding content and semantics, *Near-OOD* datasets are similar to ID data. They might consist of certain features or patterns in samples that are strongly associated with the ID collection.
Differently, the content and semantics of the *Far-OOD* datasets differ dramatically from those of the in-distribution data, go beyond semantic shifts, and include obvious covariate (domain) shifts compared to the ID datasets.
To conclude, it's important to point out a difference in the typology of methods applied to perform this kind of detection, between the post-hoc inference and training methods.
Out-Of-Distribution detection post-hoc inference techniques include giving to the input data an "OOD score" according to certain base classifier outputs. After that, these scores are thresholded to provide binary predictions. Differently, training methods involve the additional training of an ad-hoc model to enhance OOD detection capability. These methods go beyond inference and focus on improving the model thanks to training.

1.2 Problems and Objectives

Detection of deepfake media is characterized by several intrinsic difficulties, that derive from constant advancement in the generation techniques with complex models

able to synthesize ultrarealistic visual media that are very hard to distinguish from the real captures.
This type of detection should be intended to "filter" over the huge flow of media exchange from the internet (but not only) that today's times present. Precisely for this reason sustainability and practicality are important from a computational point of view, since this kind of operation could potentially be performed countless times, and it should be able to run also on mobile platforms. In addition to this use, some applications like streaming, require real-time deepfake detection. Real-time detection poses a significant challenge due to the computational power needed to analyze massive amounts of data shared online continuously. Further problems come from the reduced, incomplete, and low quality dataset used to train the deepfake detection models. The availability of data present to face this task is limited if compared to the boundless domain of fake media generated with known and unknown techniques. Therefore, Deepfake datasets may not accurately represent the deepfakes shared online, impacting the effectiveness of detection techniques. This aspect is strictly interconnected with the well known problem of generalization, and lack of robustness. This is not only due to the restricted data available and the possibility of training models only on generative deepfake techniques known and available, but also because existing methods are not robust to post-processing operations like compression, noise, light variations, etc.
Manipulation techniques seen at training time can be well discriminated for the detection, but the performances significantly drop when the methods are unknown, and detection models are not able to correctly infer with the trained distribution.
Under the direction of Explainable Artificial Intelligence (XAI) research, we can identify another weakness of these algorithms about the results' lack of interpretability. The lack of explanations for detection outcomes makes complicated to understand why certain decisions are selected by the models, which limits accountability and trust in these systems.
Despite all these difficulties, in last years interesting steps forward in the research have been conducted. Generative deepfake techniques counter these improvements through the creation of adversarial examples. Adversarial perturbations exploit vulnerabilities or weaknesses in the algorithms, leading to a more likely misclassification of manipulated media.

Out-of-distribution detection has its proper difficulties as well, some of these are shared between the two detection tasks.
As for deepfakes, the distribution of OOD data is not static and may change over time, and the detector needs to adapt to these dynamic shifts of data. Furthermore, is not easy to find an exhaustive and diversified collection of OOD data to test (and eventually train) detectors, since the concept of dataset for OOD detection is strictly related to the collection of data not seen from the model at training time.
In order to effectively evaluate in a uniform manner OOD detection approaches, several datasets are integrated into OOD benchmarks [118], which also specify the complete procedure for a consistent evaluation.
Even in this case, the literature highlights the difficulty of designing a solution that is able to generalize across different domains and In-Distributiion data, pursuing the research for more robust and explainable methods that can be effectively used for

this purpose.
Moreover, the complexity of this task is twofold, since achieving dual objectives for classification over ID data and OOD detection simultaneously can pose inherent trade-offs and challenges in shared architectures.
Effective training methods should be also convenient for hardware resources, considering the resources required by the classifier, additional computations have to be as limited as possible.
When exposed to new kinds of OOD data, OOD detection algorithms may experience catastrophic forgetting, a condition in which they lose the ability to identify abnormalities that they have previously seen. Approaches to continuous learning are required to limit this problem.
Other types of problems are more theoretical and relative to terminology, inconsistent evaluation settings, and erroneous practice between researchers' works.
For researchers as well as practitioners in the field, the small variations in sub-topic definitions and settings (anomaly identification, novelty detection, etc.) cause confusion. A clear unification of the tasks is crucial, also considering the important boost in the research which led to several reliable and effective solutions in the last decade, producing an increase of interest year by year on the topic. Works like [119] tend toward this direction.
Considering the previous characterization of generalized OOD detection, our task is represented at best under Anomaly detection, given specifics like binary detection and covariate shift. Although, a pure OOD detection task is a suitable representation of the problem for certain problem scenarios that we have designed, since present indirectly no semantic overlapping. Given the problem statement, it's impossible to avoid a total semantic overlapping in a binary classification task between real and deepfake images, but it's possible thanks to a further characterization of the samples, i.e. the image content, techniques used to generate fake samples, etc. We present a novel architecture for the identification of OOD samples in the context of deepfake detection. In detail, the proposed contributions are the following:

- Designing of an ad-hoc CDDB dataset pre-processing, providing the possibility to customize settings and build different deepfake detection dataset instances, based on the image content, deepfake groups, and random selection of deepfake techniques. This data partitioning is particularly convenient to perform both deepfake and Out-Of-Distribution detection. In this way, the dataset can be used as a new hard OOD benchmark under the philosophy of the dataset splitting from Open Set Recognition tasks. Moreover, these custom sets of data can be used also for multi-class classification over the different deepfake methodologies classes.

- We propose a pipeline composed of an In-Distribution classifier and an Out-Of-Distribution detector, which collaborates and shares auxiliary data. The In-Distribution classifier' aim is to identify deepfake samples, while OOD detector discriminates between images that belong or not to the training distribution. The OOD detection solution respects the reconstruction-based and output-based paradigms.

- Two distinct approaches are presented, involving Convolutional models and

Vision Transformers.

- Three slightly different OOD detector architectures are exposed, and integrated with both the two approaches.
- We train and test our models on the custom CDDB dataset, employing different scenario settings. We propose the results for both Deepfake and OOD detection. OOD baselines are been implemented and tested on this novel task, to make a comparison and validate the effectiveness of the approaches.
- We train and test our models also on a popular OOD benchmark, to give a wider look at the complete panorama of known techniques, and understand better the potentialities of this methodology, analyzing pros and cons.

This study exposition is organized under the following structure.
Chapter 1 provides an overview and a definition of Out-Of-Distribution and Deepfake as well. Talks about potentials, benefits, and problems related to these topics. Gives a first presentation of the proposed work.
Chapter 2 investigates several state-of-the-art approaches in the literature for deepfake generation, Deepfake detection, and Out-Of-Distribution detection. It also describes and presents relative datasets and benchmarks for the tasks.
Chapter 3 represents the core of our work, presenting the work pipeline, the different novel approaches, and the architectures on which this study is based. It describes key OOD detection clues and technical aspects of the models.
Chapter 4 outlines the dataset utilized and how it is pre-processed. It also indicates the metrics employed to evaluate the suggested approaches and the training configuration necessary to replicate our solutions.
Chapter 5 reports the results for both the detection tasks, testing our models on the proposed deepfake task and also on a popular OOD benchmark.
Chapter 6 provides conclusive details and possible future directions of our work.

Chapter 2

Related work

Nowadays, the advent of deepfake technology has introduced unprecedented challenges in the protection of both privacy and security. Numerous studies are being conducted to identify false multimedia material to avoid the diffusion of tempered information, which can consequently lead to negative scenarios.
Another open problem in the literature, is the improvement of reliability and safety in machine learning, to avoid falling victim of overconfident results over data never seen. In other words, we want models that are able to generalize as much as possible, in order to really enhance of AI in our daily routine.
For this purpose, the well-known challenge of finding a robust deepfake detection solution, can be integrated with the reliability degree coming from the Out-Of-Distribution detection task. In this chapter, are highlighted previous works that are task-related to our proposal. We approach the problem first by looking at models designed to generate deepfake content, with a particular emphasis on visual media. Afterward, we introduce various techniques on both Deepfake and Out-Of-Distribution detection, pointing out the current state of the art. Finally, we show an overview of the available datasets to address both tasks, discussing their characteristics.

2.1 Visual Deepfake Generation

Deepfake-related studies represent one of the emerging trends in the last decade in the fields of Artificial Intelligence and Machine Learning, attracting a lot of interest in the new techniques and methods proposed. These algorithms are so effective that make possible for everyone the creation or alteration of media, with very little effort and stunning outcomes. What the user should provide as input to these models depends on the type of deepfake technique that we want to apply, which can vary from source and target media to simple text.
As explained in the previous chapter, with the term "deepfake", we express the process that involves a Deep Neural Network (DNN) in the generation of media like images, videos, audio, and text. The methods that focus attention only on one of these media are called unimodal, while multimodal systems can elaborate different typologies of data, like parallel audio and video generation.
In this section, we focus on the visual content, in particular on image generation. It's

important to highlight that image-based generation often shares common underlying technique with video-based generation, extending the procedure to a sequence of images over time.
Visual Deepfake can be classified into five different types of manipulation. One of the most popular techniques, which has become very popular at the moment, is face swapping [60]. The objective is to substitute the face from a source image with the one of another person in another image, which is called target. Using this technique, the target subject's original facial emotions and characteristics are transferred onto the corresponding parts of the source subject's face.
Instead face reenactment [125] transfers facial expression without changing the aspect, from the source to the target. Altering only a region of the original image, changing facial aesthetic is the main goal of attribute manipulation [66].
Face synthesis [93], whose objective is to generate human faces of non-existing people, making the result as realistic as possible.
Combining together visual and audio or text data, deepfake lip-syncing techniques are involved to generate video with a target face synchronized with lip movements from given sources.
Although face swapping and altering facial features in photos and videos makes deepfake so popular, in reality, the term refers to a broader range of artificial content creation. Deepfakes, in a broader sense, can be defined as any media (not related to content) that is produced or altered through the use of deep neural networks.
Numerous existing techniques are used to generate deepfakes, Among the most common we have Generative Adversarial Networks (GANs) [30], Autoencoders (AEs) [89], Variational Autoencoders (VAEs) [47], Diffusion models [32] and Convolutional Neural networks (CNN) [52].

The Generative Adversarial Networks introduction (2014), represents an important milestone in the field, taking a crucial role in advancing of generative models, proposing a new methodology for the synthesis of realistic and high-quality content.

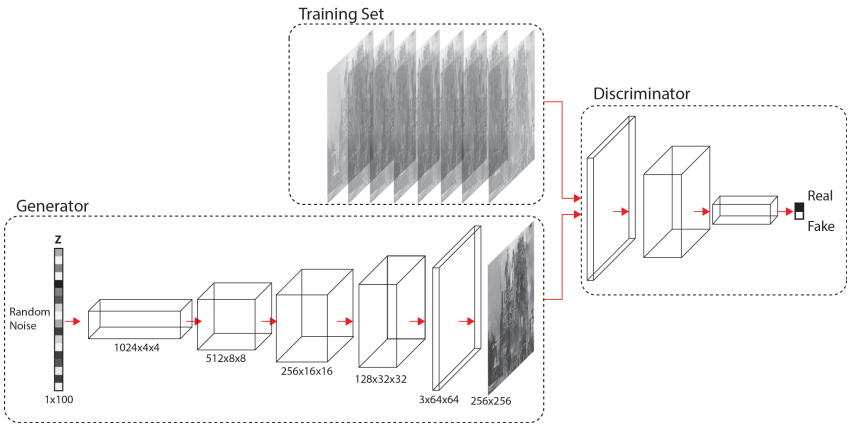

Figure 2.1. Scheme of a basic GANs architecture.

GAN models are designed with two main components, the Generator and the Discriminator, as is possible to appreciate in the image 2.1. The term "adversarial"

comes from the particular mechanism that characterizes these two elements. The Generator's goal is to reproduce images with the highest possible fidelity, while at the same time, the Discriminator's objective is to classify and discriminate between authentic and artificial images, being learned on both real and fake data.
By essentially pushing one model to try to outwit the other, this iterative process creates deep neural networks with amazing representation capabilities.
This is the key idea behind GAN architecture, nevertheless, there are a variety of innovative GAN-based methods in the literature that feature advanced and complicated deepfake generation techniques.
StyleGAN [44] is a GAN architecture that produces high-realistic and detailed images and at the same time, allows the manipulation of various facial features, which is a very useful functionality for the creation of deepfake content. A further step in the synthesis process by StyleGAN 2 [45], that makes changes in generator and discriminator networks leading to a better result and improved training stability.
STGAN [66] is designed to manipulate specific attributes of the face, and at the same time, it doesn't affect the person's identity.
ProGan [43] proposes a new technique that increments the resolution of input images and the number of layers for both the discriminator and the generator, which helps in the training stabilization. To transfer facial features from one individual to another, it's possible to use CycleGAN [130] which is an expert in transfer between two domains. StarGAN [14] enables the modification of facial traits in images, such as gender, the addition of eyeglasses, and the color of skin or hair.
Nvidia proposes its own GAN model called GauGAN[77], which uses a semantic map as input, defined by the user via colors, to generate photo-realistic images based on the map. BigGAN [12] is a type of generative adversarial network meant for high-fidelity, high-resolution image production.

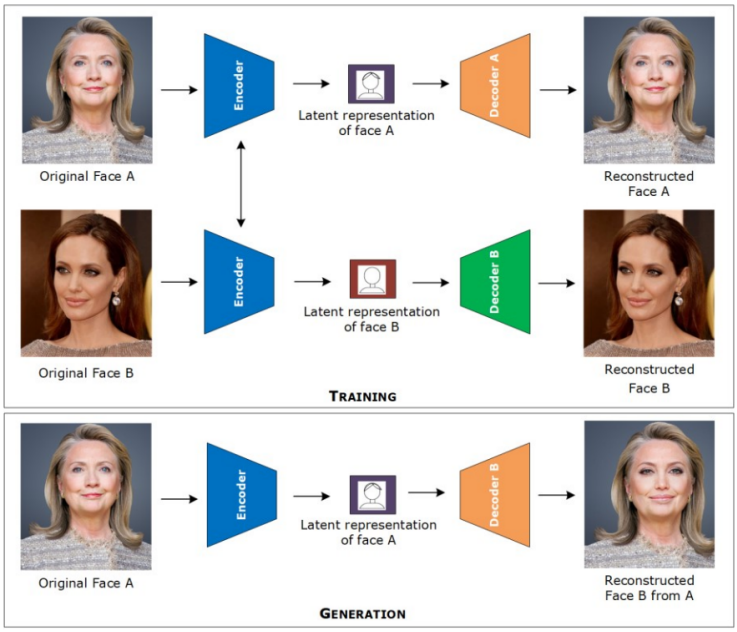

Figure 2.2. Face swapping via Autoencoder.

Facial feature encoding and decoding can be accomplished with AutoEncoders acting as feature extractors. The AutoEncoder achieves the ability to compress an input face image into a lower-dimensional representation while preserving key facial features during training. It is then possible to rebuild the original image using this latent space representation. However, two AutoEncoders are used to generate deepfakes: one is trained on the source's face, while the other is trained on the target one (Fig. 2.2). After training is finished, the decoders switch sides, allowing the target image to be generated using the features of the source image by using the original encoder of the source image and the target image's decoder. The target's facial expressions are preserved, while the source's face appears on the target in the final image. This methodology is at the basis of architectures used from popular applications like FakeApp [26] and Reface [2].
Face2Face [100], FaceSwap [108], FaceSwap-GAN [73] and Fast Face-Swap [52] provide advanced techniques for the manipulation of facial expressions in video. In particular, Face2Face is a real-time tool for manipulating facial expressions that uses a nonrigid model-based bundling technique to project a modified source picture onto a target face.
Different open-source online tools are available online to create deepfakes. Their accessibility, mixed with the simplicity of use for the creation of complex work, which otherwise would require important graphic and artistic skills, makes them popular and widely used for the creation of visual content. Examples are Deepfakes Web [114], DeepBrain [6], DeepFaceLab [78].
We conclude this section by referring to the state of the art in image synthesis. In these architectures, the user inserts a textual input and the tool exploits NLP techniques, like word embedding, to bridge the gap between textual descriptions and visual content. This new contextual information can be used by the generative model to reproduce high-quality images. Examples of these tools are: DALL-E [80]

Figure 2.3. Image generated using Stable diffusion model.

and MidJourney [75]. Stable Diffusion [86] proposed a method for the synthetization of images from texts, through a diffusion process, which refers to the gradual transformation of basic distributions, like noise, into complex data. It is based on a latent diffusion model, which is trained to remove noise from images in an iterative process. Stable Diffusion is an energy-based model that learns to generate images

by minimizing an energy function that represents how well the image matches what the text represents. In figure 2.3 we have an example of fake image realized with this technique.

2.2 Visual Deepfake Detection

Deepfake detection in visual media is a task that can be addressed by following different approaches and exploiting different types of clues and intrinsic data information. Being deepfake generation techniques are always in progress, the necessity to find a robust and general solution to identify real and fake examples is the main goal, but different works approach the problem by reducing the detection for only certain categories of tampering and forgery.
Deepfake detection via Biological/Physiological is present in [62, 37]. Li, Y. et al. adopt the use of a method based on recognizing eye blinking, a biological signal that is difficult to interpret in deepfake footage. As a result, the lack of eye blinking indicates the presence of a deepfake video. A deep neural network model that combines a recursive neural network and CNN, and takes into account temporal information, is utilized to identify open and closed eye states.
Differently, Hernandez-Ortega et al.[37] provide a novel method for identifying deepfake movies that employ remote photoplethysmography (rPPG) to analyze heart rate data. It is possible to determine whether human blood is present underneath the tissues by watching video sequences and seeing slight changes in skin tone.
A Convolutional Attention Network is incorporated into the suggested detection system, DeepfakesON-Phys, to extract spatial and temporal features from video frames and efficiently provide a deepfake detection system for video sources.
Some focus on discrepancies in their behavior, such as differences in the movements of their lips, bodies, head positions, and left and right eyes [33]. Another common approach is about the identification of GAN-originated media is to take into account convolution traces. Huang et al. [40] used a gray-scale fakeness map to take advantage of the artifacts in the upsampling process in GAN-generated deepfakes.
To further increase the robustness of the model, the attention mechanism, partial data augmentation, and individual sample clustering are used.
L. Guarnera et al. [31] presented a technique for extracting a collection of local features meant to mimic the convolutional patterns commonly seen in images using an expectation maximization approach.
To capture spatial irregularities and temporal changes, Waseem et al.[113] presented a dual-stream convolutional neural network technique that incorporates XceptionNet [15] and 3DCNN. MTCNN [129] is first used for input video frame face extraction and detection. XceptionNet and 3DCNN are then used to extract features from face photos. Subsequently, sigmoid and fully linked layers assess that videos are not generated using deep learning models.
In the literature, a lot of different approaches have been employed against deepfake detection tasks, including media forensics techniques [107], biological signs, emotional signs, spectrum analysis [9], etc.
Thanks to the introduction of the dynamic routing algorithm [91] and its variance, namely the expectation-maximization routing algorithm [38], they use a novel kind of CNN that performs remarkably well on computer vision tasks, in contrast to other state-of-the-art detectors that use conventional CNNs with a lot of parameters. They use a face identification method and crop the desired facial area if the task involves identifying a fake face. A small pre-trained VGG16 network [94] is used as feature extractor. Their work pipeline is characterized by a pre-processing part that

depends on the input (they split the frames if the input is a video; then they divide each image into patches if the aim is to identify deepfake images).
The features are then sent into the capsule network, made up of two output capsules (fake and genuine) and several primary capsules. A pooling layer, a 1D convolutional component, and a 2D convolutional section make up the primary capsules. Following the 1D convolutional layer, the output is subjected to the dynamic routing method before being sent to the output capsules, where the ultimate result is established.
Rather than limiting his analysis to facial discrepancies, [74] additionally examines the surrounding environment of the face. The authors specifically note that the deepfake generator adjusts the face area while preserving the context, like in the case of Face Swapping. Thus, it is feasible to easily identify fake media by concentrating on potential differences between faces and backgrounds. In figure 2.4 we give an example of this inconsistency.
Two models compose the architecture: a face identification network that uses a

Figure 2.4. swapped faces and their context.

tight semantic segmentation to identify the face region it surrounds, and a context recognition network that considers facial context such as hair, ears, neck, etc. This produces two distinct identity vectors, which may then be compared to identify any differences. This provides a complementary detection signal that improves the performance of deepfake classifiers. Deep learning-based models, on the other hand, are considered by most the best choice to distinguish between fake and real digital content, reaching remarkable performances. These methods take advantage of neural network models as backbone networks like ResNet [34], VGG [94], EfficientNet [97], Inception [96], ViT [25], etc. used for feature extraction.
Entire face synthesis detection has also been the subject of several research. Several types of infer systems can be identified in the literature, each one focusing on a particular aspect like color [71], GAN fingerprint [123], PRNU (Photo Response Non-Uniformity) [69], self-attention mechanism [61], neuron behaviors [111], and so on.
The authors of [81] present Deep-fakeStack, a deepfake ensemble learning technique that essentially entails combining many cutting-edge DL-based classification models to produce a superior composite classifier. The models are pre-trained on ImageNet and apply transfer-learning and using GLP (Greedy Layer-wise Pretraining) [10] for training. The models included are the following: ResNet101, DensNet121, DenseNet169, InceptionV3, XceptionNet, InceptionReseNetV2 and MobileNet.
The idea of ensemble learning is also applied in [76], where the authors first investigate deepfake detection using popular deep learning models like MobileNet and

Xception, and then they utilize an ensemble of the two models to do the prediction. Both models are built on convolutional neural networks (CNNs) and employ depthwise and pointwise convolutional layers; the difference is that MobileNets incorporates fewer features to increase the model's efficiency.
The usage of a Convolutional Vision Transformer (CViT) to detect Deepfakes is introduced by the authors in [116]. The CViT can be divided into two main segments: The Convolutional part, known as Feature Learning (FL), and the ViT for the final classification. The convolutional portion's structure is very similar to the VGG architecture and is distinguished by the absence of the last fully connected layer and a stack of convolutional layers.
This since rather than classifying images, this section is utilized to extract facial image attributes. After the features have been extracted, the input is passed into the ViT, which transforms it into a sequence of image pixels in preparation for the final detection stage. In contrast to the traditional Transformer design, the ViT architecture just makes use of an encoder composed of MLP and MSA blocks.
One of the common problems in the detection of forgeries is the overfitting of the model on the training data, and the consequent lack of generalization across different datasets and deepfake techniques. Korshunov et al. [51] investigate techniques to improve the generalization of deepfake detection methods, and propose a methodology that includes data augmentation and few-shot tuning methods.
To increase the detection algorithms' accuracy in a cross-database scenario, we can perform aggressive augmentation during few-shot learning.
Few-shot learning models can quickly adapt to new classes with only a limited set of new examples, making them more versatile and adaptable.
The augmentation technique proposed is called data farming, and involves augment-

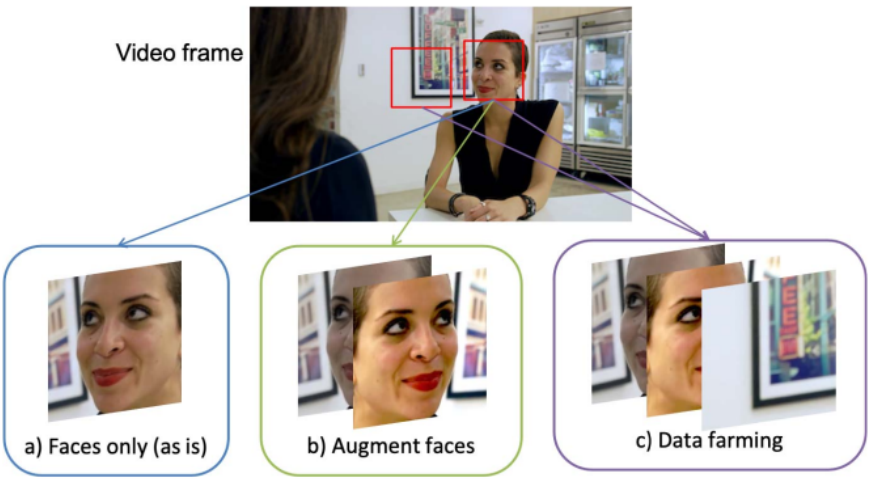

Figure 2.5. Data farming process.

ing training data by mixing together background patches in equal proportion with faces from genuine samples. This approach aims to force neural networks to learn the visual artifacts of deepfakes, reducing overfitting of facial features. By doubling the genuine data portion through the addition of background patches, data farming helps balance the training data, which is often significantly smaller compared to the

amount of deepfake data.
The paper proposes an investigation about the effectiveness of few-shot tuning methods in improving the generalization of deepfake detection systems. It explores and evaluates different approaches among no model tuning, tuning the first convolutional layer of the neural network, and tuning the last fully connected layer. According to the evaluation conducted by this work, tuning the neural network's first convolutional layer is the most effective method among the three few-shot tuning techniques for enhancing the generalization of deepfake detection system.
The work in [7] proposes DDT (Deep Distribution Transfer) a novel zero and few-shot transfer learning method for facial forgery detection that models data with a multimodal distribution. Still, the objective is to generalize the forgery detection using pre-trained models. The approach focuses on modeling data with a multimodal distribution and incorporates a spatial mixup augmentation strategy to enhance model generalization in unseen scenarios. Figure 2.6 points out the complete ap-

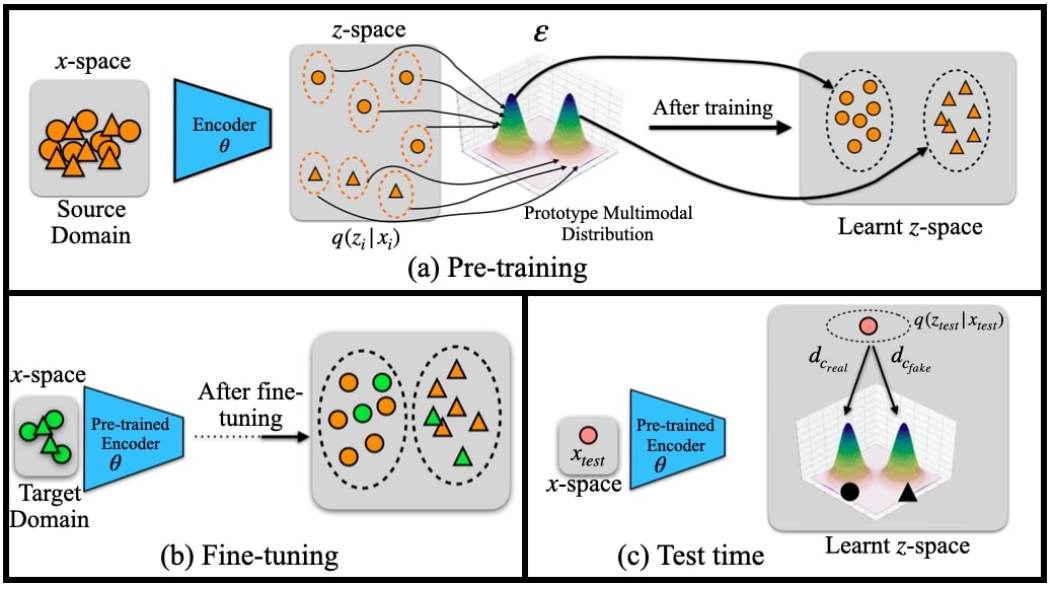

Figure 2.6. Deep Distribution Transfer method overview.

proach.
A single multi-modal distribution ϵ is learned with non-overlapping means. This distribution consists of several unimodal distributions ϵ_c, each representing a specific class C. Each ϵ_c is characterized by the mean m_c and the covariance matrix Σ.
Each class $c \in 0, ..., C-1$ is associated with a Gaussian distribution ϵ_c in the mixture model. Means $m_0, m_1, ..., m_{C-1}$ of these Gaussian distributions are learned to ensure non-overlapping representations for each class.
The model learns to project data points into a latent space (via an encoder) where each data point is mapped to a specific component of the multi-modal distribution. By enforcing non-overlapping means in the multi-modal Gaussian distribution, each class is effectively represented as a distinct unimodal distribution, facilitating accurate classification and detection tasks.
With this technique, class distributions are effectively modeled without overlap, since

each class is guaranteed to be uniquely described by a Gaussian distribution with specific mean and covariance.
After the pre-training the model is fine-tuned and relative metrics are estimated.
The results of this work demonstrate the effectiveness of the proposed Deep Distribution Transfer method for facial forgery detection, outperforming past works in the literature by a significant margin in both zero-shot and few-shot settings.

2.3 Deepfake Detection Datasets

Several datasets are available for deepfake detection studies. Here we enlist the most relevant, briefly describing their characteristics. The list does not include only pure visual deepfake detection works but it poses more importance to them.
The performance, generalizability, and resilience of deepfake image detection algorithms are boosted by these datasets.

- UADFV [120]: The first dataset released for deepfake detection. It proposes 98 videos, totaling 32,752 frames, evenly split between 49 real videos and 49 fake ones. The deepfake videos are generated from the real ones by applying a basic deepfake technique like FakeApp [26].
  Videos have an average duration of 11.14 seconds and an average resolution of 294x500 pixels. Nevertheless, movies have extremely poor visual quality, making the resulting alteration visible and simple to spot.

- Celeb-DF [63]: this popular deepfake dataset is a compilation of real and artificially generated deepfake videos with characteristics that match those commonly shared online. It offers better quality videos and attempts to address the issue of obvious source artifacts prevalent in earlier data collection, using an enhanced synthesis [83, 8] process. It Comprehends 5,639 fake videos and 590 real ones, with clips coming from YouTube interviews with 59 celebrities. Videos last an average of 13 seconds and have a frame rate of 30 frames per second. Identification of fake faces among numerous real faces in scenes taken in nature is a significant difficulty.

- FaceForensics or FF[90]: over 500,000 frames, mostly of frontal faces from 1004 films that were obtained from YouTube, constitute the FaceForensics dataset. Its frames are combined into short, continuous sections. The FaceForensics dataset is only FaceForensics++'s earlier iteration.

- FaceForensics++ or FF++ [88]: this dataset was presented in 2019, one year after the publication of FF. This set of data is built upon 1000 video sequences with audio channels, all collected from YouTube-8M dataset [5], and 3000 videos edited.
  It doesn't only include videos with altered expressions and it's made up of four subsets representing each one a different manipulation technique: FaceSwap [60], DeepFake, Face2Face [100], and NeuralTextures [99].
  Every video has an occlusion-free, trackable, and mostly frontal face, which makes possible for automated tampering techniques to create realistic fakes. Furthermore, the vast majority of the talks are given in languages different

from English.
Because the dataset is provided in two distinct quality levels, it may be used to test deepfake detection algorithms on both compressed (H264 format) and uncompressed videos.
Nonetheless, the FF++ dataset is limited in the generalization of lip-sync deepfakes, and several clips show uneven color around the altered faces.

- FakeAVCeleb [46]: is a multimodal dataset for deepfake detection that contains cloned deepfake audio and deepfake videos. It includes celebrities that differ in terms of gender, age, and race, with source videos taken from the VoxCeleb2 dataset[16] by selecting videos of 500 celebrities. Specifically, it has 19,500 deepfake videos and 500 genuine ones. The techniques used to generate fake samples include face-swapping [60] and facial reenactment [125]. Every video is clear and shows just one person's front face without any type of occlusion. 11 distinct deepfake detection methods, including multimodal, ensemble-based, and unimodal techniques, were used to evaluate the dataset.

- WildDeepfake [132]: is a dataset that is well known to be challenging for deepfake identification. It has 7314 high-quality face sequences taken from 707 Deepfake videos that were collected from the internet. Its inclusion of both real and deepfake samples from the internet sets it as unique from other available datasets. Different lighting, positions, facial expressions, and indoor/outdoor locations characterize videos. It is a small dataset that includes a variety of body types and should be used in addition to other datasets, to improve the generalization.

- CDDB [57]: to face this detection problem in a real-word scenario, The CDDB (continual deepfake detection benchmark) dataset proposes a continual learning [110] approach over sequences of easy, hard, and long deepfake detection tasks. This set of data is consistent to be used in a purely binary fashion, or can be adapted to work also for multiclass task format.
The objective is to measure detectors' ability to learn detection tasks without forgetting. Existing benchmarks are limited to known generative models, while CDDB includes diverse types of deepfakes for real-world scenarios. The data comprehends samples from GAN based models [30], non-GAN based models, and unknown sources coming from the Internet. It also incorporates the data from the FaceForensics++ [88] and WildDeepfake [132]. CDDB covers a wide range of deepfake generation methods, enhancing the variety of data for evaluation, and offering multiple experimental setups to handle deepfake detection methods. Moreover the different sources, CDDB reduces the gap between existing benchmarks and real-world scenarios.
On the other hand, researchers may need to adapt this data due to the unique challenges posed by CDDB, potentially requiring additional time for understanding and implementation. In addition, Working with a dataset that includes known and unknown deepfake sources may require more computational resources for analysis.

- DFD [11]: also known as DeepFakeDetection, is a dataset that was developed by Jigsaw and Google. It includes a variety of situations, including over 363 real sequences with 28 hired actors spread over 16 distinct scenes. It also has more than 3000 manipulated videos.

- DFDC [24]: the deepfake detection challenge dataset launched by the Facebook community represents the most extensive collection of face swap videos available and openly accessible. It contains over 100,000 video and audio samples collected from over 3,426 paid actors, of several genders, ages, and ethnic groups. Unlike other datasets, videos are produced in adverse visual conditions with problems of high or low light levels and no professional makeup or lighting. The dataset is generated by exploiting different augmentations, like geometric and color transformation, varying frame rate, etc.
  Approximately 48,000 high-quality video clips, each lasting roughly 70 seconds, made up the source material. The videos last roughly 15 seconds each with an fps of 30, so there are 300 frames per video.
  To be noticed, the high level of unbalancing between samples' classes, there is a nearly 5:1 ratio between falsified and authentic videos.

- DeeperForensics [41]: another Large-Scale dataset for deepfake detection that contains 50,000 original clips and 10,000 forged ones. Differently from the DFDC, it contains videos recorded in a controlled studio setting. These altered videos are created with the use of DF-VAE, a conditional autoencoder. The actor appearance samples in the dataset are quite distinct. To further better reflect the real-world settings, a variety of distortions and disturbances, such as compression, blur, noise, etc., are introduced. Videos are characterized by a resolution of 1920x1080 pixels recording 100 hired actors with different skin tones, genders, ages, and emotions from 26 different countries.

- DeepfakeTIMIT [49]: a standard dataset for deepfake detection which was introduced in 2018. It comprises 620 videos where faces were swapped using GAN-based techniques. It was designed by taking 16 pairs of people that looked similar and assigning each pair one of two quality levels (64×64 and 128×128) from the VidTIMIT database [50].
  There are 640 face-swap videos in the collection, and each of the 32 subjects is related to ten clips. The original audio information is unaltered (only visual manipulation).

2.4 Out-Of-Distribution Detection

The importance and potential of Out-of-Distribution techniques could play a key role in real-world applications.
The first work that we associate with OOD detection in the literature, is the MSP (Maximum Softmax Probability) baseline [36]. Even if the prediction probability from a softmax distribution has a poor direct correspondence to confidence, this work shows that prediction probabilities of incorrect and Out-Of-Distribution examples

tend to be lower than the prediction probability of correct examples.
The approach involves taking the maximum softmax probability of each inference from a neural network model to discriminate between correctly classified examples and misclassified or out-of-distribution examples.
It's demonstrated through extensive experimentation in different tasks relative to Computer Vision, Natural language processing, and the automatic speech recognition field, that simple statistics derived from softmax distributions provide a surprisingly effective way to determine whether an example comes from a different distribution from the training one.
For computer vision, the tasks formulated include image classifications using datasets such as MNIST [22], CIFAR-10 and CIFAR-100 [1].
furthermore, the authors not only proposed a simple method to discriminate between In and Out of distribution, but also suggested the application of the same technique to separate correctly and wrongly classified samples.
The terminology used in this study was maintained in this thesis too, referring to **normality** whether examples fall within the training distribution, and **abnormality** for samples that are outside the training distribution.
ODIN (Out-of-DIstribution detector for Neural networks)[64] exploits gradient information to perform OOD detection. In particular, ODIN aims to detect out-of-distribution data samples in neural networks without the need for retraining. Starting from the baseline evidence [36], authors propose the usage of temperature scaling in the softmax function (formula 2.1), and adding small controlled perturbations to inputs (formula 2.2).

$$S_i(x;T) = \frac{\exp\left(\frac{f_i(x)}{T}\right)}{\sum_j^N \exp\left(\frac{f_j(x)}{T}\right)} \tag{2.1}$$

$$\tilde{x} = x - \epsilon \cdot \text{sign}(\nabla_x \log S_{\hat{y}}(x;T)) \tag{2.2}$$

With f the neural network, T the temperature scaling, ϵ the perturbation magnitude, x the input that is elaborated by f and assigns a label $\hat{y}(x) = \arg\max_i S_i(x;T)$. The perturbed input is then forwarded through the network, and the threshold-based detector discriminates between In and Out of distribution. It was discovered that the perturbations increased the difference between the softmax scores of the ID and OOD inputs, making them easier to distinguish and enhancing OOD detection results.

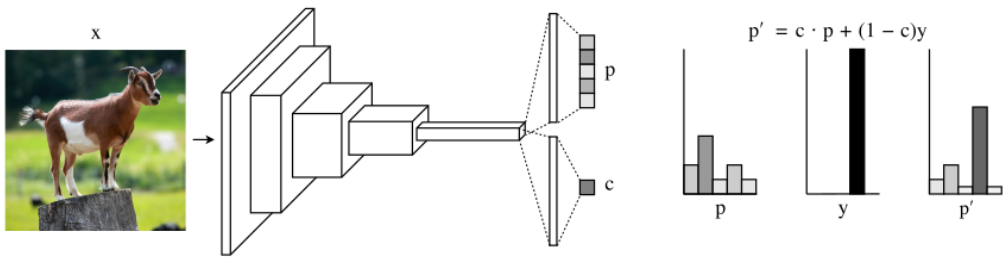

Figure 2.7. Use of confidence branch to detect abnormal examples.

The confidence estimating branch [23] method in figure 2.7, shows how is possible to perform OOD detection without accessing the output distribution and the necessity of having extra labels. This approach incentivized the neural network to produce confidence estimations that correctly reflect the model's ability to produce a correct prediction for any input. In parallel to the output classification layer, is incorporated a confidence estimation branch that produces one single value per input. Usually, the branch is added after the penultimate layer of the original classifier architecture, and it receives the same input as the prediction or classification layer. After the application of the sigmoid activation function, the confidence estimation value should output values close to one when the network is confident, while values toward zero if is not confident. This work also proposes a custom loss function and other techniques to stabilize training (hints, budget hyperparameter, etc).
After training, we can perform a threshold-based classification only based on the confidence estimation.
The results of this work have shown an important improvement in the OOD metrics. In a multi-label context, the techniques proposed by Want et al. [109] called JointEnergy for out-of-distribution (OOD) detection, aggregating label-wise energy scores across all labels to establish a new OOD score. It establishes an important technique to efficiently apply OOD detection also with multi-label classifiers. JointEnergy can be theoretically interpreted from a joint likelihood perspective. The joint likelihood allows separability between in-distribution vs Out-Of-Distribution data, since OOD data is expected to have lower joint likelihood.
Having a lot of dominant (high likelihood) labels is indicative of an in-distribution input, which is the key aspect that JointEnergy captures. It's demonstrated how summing labels' scores is less performative rather than summing labels' energies, emphasizing the reasons behind JointEnergy. A similar approach can be applied also to a multi-class task, as proposed by Liu et al. [67].

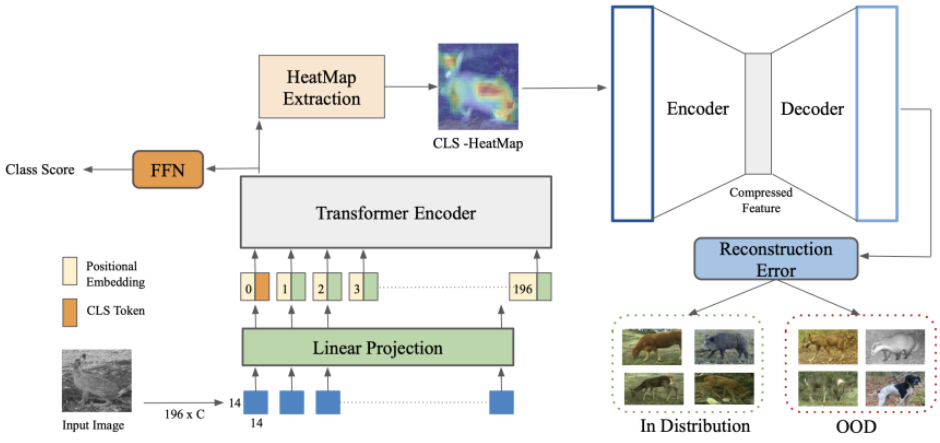

Figure 2.8. Attention Heatmap based Out-Of-Distribution scheme.

It's also possible to exploit visual attention heatmap [19] from the Vision Transformer (ViT) classifier to perform OOD detection.

OOD detection models should not incorporate too high computational complexity and also require as little supervision as possible. This approach suggests the extraction of the attention map combined with an additional autoencoder (AE) to reconstruct this map. The AE learns to encode meaningful and salient features of the input images, enabling accurate image reconstruction. Authors use the reconstruction error as discriminative information for the classification of OOD samples and ID examples. Figure 2.8 represents the overall architecture for this approach.
We also mention the possibility of including the CutMix [124] augmentation technique to improve training generalization, which has been proven to have a positive effect also for a consequent OOD detection test. This method proposed the following strategy: patches are cut and pasted among training images where the ground truth labels are also mixed proportionally to the area of the patches. It makes efficient usage of training pixels and retains the regularization effect of regional dropout.
Among all the techniques that don't require additional OOD data, we mention the possibility of using Gram matrices [92] to detect samples out of the distribution. Analyzing feature correlations in Gram matrices at different layers of neural networks, is possible to identify anomalies in activity patterns without the need for access to OOD examples, working across various architectures and datasets.
Deep neural networks (DNNs) use gram matrices to calculate pairwise feature correlations. Through the capture of relationships between different features, they encode information like patterns. Gram matrices are expanded in this context to calculate feature correlations' class-conditional limits at various network layers.
These techniques fall under different categories based on the type of method exploited to infer In and Out of distribution examples. It's important to highlight that a specific approach can belong to more than one of the following classifications.
Output-based methods in Out-of-Distribution (OOD) detection focus on utilizing the model's output. This category can be further characterized by Post-hoc detection and training-based methods. Post-hoc methods don't change the training procedure and objective [36, 64, 109]. The most recent approach, SHE [128], is a hyperparameter-free and computationally efficient algorithm, it measures the discrepancy of unseen data using stored patterns that represent classes.
Instead, training-based methods modify the training of the classifier at different levels like model structure, loss function, augmentation, normalization, etc. One example is G-ODIN [39], which extends ODIN [64] using a specific training loss and the selection of hyperparameters on ID data, including perturbation magnitude. G-ODIN is considered a training-based method since necessitates model retraining.
Bayesian models are employed in this kind of problem [104], they represent uncertainty through inference based on Bayes' rule.
Bayesian neural network [72] represents the most known method of this category, based on Markov chain Monte Carlo [29] to draw samples from the posterior distribution of the model. These techniques like Laplace methods [27] and variational inference [79], lead to inaccurate prediction and high computational costs.
Recent studies have attempted several less rigorous approximations, such as deep ensembles [53] and MC-dropout [28], for faster and better modeling of the uncertainty, nevertheless, they are less performative with respect to other techniques in the estimation of OOD uncertainty.
Density-based techniques for OOD identification utilize some probabilistic models

to predict the In-distribution and classify test data in low-density areas as OOD. Examples of methods for this class involve multivariate Gaussian distribution [56], mixed Gaussian distribution [82], and Poisson distribution [103].
Another set of methods falls under the category of distance-based methods. Distance-based methods for Out-of-Distribution detection operate on the principle that OOD samples should be significantly distant from the centroids or prototypes of In-Distribution classes. For instance, the detection thanks to minimum Mahalanobis distance [55] to all class centroids for detection. Further work exploits this idea through a series of more complex operations and pre-processing [84]. Other works use also different distance functions like the cosine similarity [98].
Reconstruction-based methods imply training of encoder-decoder model on In-Distribution data [19]. The intuition suggests a higher reconstruction error for samples that don't belong to the ID, due to the difference between the trained distribution and the one unknown.
They differ in pixel-level or image-level error computations and how they handle this error [121, 42, 58].
To conclude, we expose a further class of methods that includes outlier exposure, with real or generated outliers.
Methods with real outlier exposure involve using a set of collected out-of-distribution input, or "outliers," during training to help models learn the discrepancy between in-distribution and OOD data. MixOE [126] proposes to mix ID and OOD images to obtain informative outliers for better regularization.
Whether OOD data are not available which is very common in practice, it's possible to use outlier data generation. One possibility is to use GAN models to generate OOD samples [54, 95] and force to have a uniform prediction or achieve high confidence for OOD samples.
We finally mention the popular tendency of adopting transfer learning as a common base for the classification models used in OOD detection research work. A key factor in their success is the massive pre-training they acquired on large datasets.

2.5 Out-Of-Distribution Datasets

We conclude this chapter by presenting and describing a list of significant datasets that can be used to deal with Out-Of-Distribution detection. As before we also give a short description, highlighting the key points.
In the Out-of-Distribution detection context [119], are utilized pre-existing datasets built for different purposes (like classification, and segmentation), typically separating data into two categories.
The first category is the In-Distribution Datasets, which are datasets used for training the model, and represent the familiar data distribution that we want to learn.
The Out-of-Distribution Datasets represent the other set, these data are never seen from the model, and they contain other samples that are not included in the ID set.
The OOD set can be characterized from very different categories or partially related to the ones in the ID.
The combination of different datasets, and assigning arbitrarily roles of ID and OOD

is the common practice in this task and allows the possibility to explore and evaluate the detection systems in many ways.
Datasets choice combined with well-known metrics used in the evaluation procedure helps in the standardization of tasks, defining the so called benchmarks.
The benchmarks help in the reproducibility of the experiments and comparison between models. The purpose of benchmarks is to offer a fair and consistent framework for evaluating the efficacy of different OOD detection strategies.
The presence of a lack of uniformity and standardization between the different benchmarks that the literature suggests, poses a significant challenge in the field of OOD detection.
The recreation of the same experimental setting to correctly compare different techniques is a complex aspect that works like OpenOOD [118, 127] tries to mitigate.
Follows a list of datasets for classification that are used both in training and evaluation of OOD detection techniques in existing benchmarks.

- CIFAR-10 and CIFAR-100 [1]: is a popular set of data composed of 60,000 color images, each with a spatial resolution of 32 $\times$ 32 pixels. The ten classes are represented by 6,000 samples for each category. The classes represent a limited range of objects like cars, trucks, ships, frogs, dogs, cats, deer, and airplanes.
  CIFAR-10 (Canadian Institute For Advanced Research) is employed for image classification tasks; it offers a standardized framework for assessing how well machine learning models can identify and classify objects in tiny, low-resolution pictures. A more challenging upgrade of this dataset is CIFAR-100. In this case, the classifier has to learn to distinguish between a larger set of classes, including very similar categories. This upgraded version is particularly helpful in estimating how robust and adaptive machine learning models are in various classification situations.
  The most important limit of this work concerns the simplicity of CIFAR images, that not fully represent the huge complexity of real-world scenarios. Moreover, the reduced spatial dimensions could limit the possibility of capturing consistent semantic context.

- SVHN [3]: this dataset for digit classification comprises 600,000 32$\times$32 RGB images of printed digits (numbered from 0 to 9) that have been clipped from photos of house number plates. The reduced pictures retain surrounding numbers and other sources of distraction, but they are concentrated around the digit of interest. The real-world images are captured from Google Street View.
  SVHN (Street View House Numbers) comprises three sets: training, testing, and an additional set of 530,000 easier-to-understand pictures that may be utilized to improve the training process. The samples present variable resolutions and qualities, which can be an additional difficulty for the model generalization.

- ImageNet [21]: based on the WordNet hierarchy in which each node of the

hierarchy is depicted by hundreds and thousands of pictures, 14,197,122 images have annotations in the ImageNet collection.
Since 2010 the dataset has been used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).
In addition to image classification, also includes data to solve the object detection task, encouraging the development of more advanced algorithms. ImageNet contains a large collection of images covering a wide range of object categories, spanning from animals and plants to everyday objects, vehicles, and scenes. This variety is also extended to resolutions, sizes, and qualities of the images representing real-world scenarios.
It's important to highlight the important contribution of this dataset to transfer learning since commonly several computer vision pre-trained models are based on this work. ImageNet 1K is a well-known subset of ImageNet With 1,000 classes and at least 1,000 total images in each class, ImageNet 1K has a training set of somewhat more than 1.3 million images. Larger subsets of the whole ImageNet dataset, called ImageNet 5K, 9K, and 21K, comprise the most prevalent 5,000, 9,000, and 21,000 image classes.

- TinyImageNet [101]: is a reduced-scale version of the original ImageNet dataset that is used to train and evaluate image classification and object detection techniques. TinyImageNet is smaller and more manageable for research purposes but maintains the hierarchical structure of ImageNet, which consists of millions of images over thousands of categories.
  includes 100,000 images of 200 classes, 500 for each class, all reduced to 64×64 pixel color pictures. This dataset represents a valid alternative for researchers, more computationally efficient but with a reduced diversity, and a consequent lack in the generalization for the models.

- MNIST [22]: is a widely used benchmark in the field of machine learning, this dataset is a large composition of handwritten numbers (0 through 9). It has 10,000 instances for testing and 60,000 examples for training. It is a subset of a larger NIST Special Database 3. Samples are centered in 28×28 grayscale images. MNIST (Modified National Institute of Standards and Technology database) represents a milestone among all the Machine learning datasets, but most of the time we refer to this as a toy dataset that can be used to train simple classifiers.

- F-MNIST or Fashion-MNIST [117]: comprises 28×28 grayscale images of 70,000 fashion products from 10 categories, with 7,000 images per category. The training set has 60,000 images and the test set has 10,000 images. The structure of this dataset is similar to the one of the MNIST, but is created to constitute a more challenging alternative. It provides a diverse set of clothing items for training and testing models. In a similar way to MNIST, the literature refers to F-MNIST as a simple dataset that can be used for educational purposes.

- LSUN (Large-scale Scene UNderstanding Challenge) [122]: is a dataset designed for scene understanding and representation learning in computer vision. The LSUN (Large-scale Scene UNderstanding Challenge) contains 10 scene categories, such as dining room, bedroom, chicken, outdoor church, etc. for training data, each category contains a huge number of images, ranging from around 120,000 to 3,000,000. The validation data includes 300 images, and the test data has 1000 images for each category. It includes different subsets too, each focusing on specific aspects of scenes. Moreover, Images are typically high-resolution, providing detailed visual information for training and evaluation.

- DTD [17]: is a library of textures with 5,640 images arranged into 47 classes derived from human perception. Every category has 120 pictures with a resolution between 300×300 and 640×640, and each image includes at least 90% of the surface that corresponds to the category associated. The photos were gathered from Flickr and Google.
  One remarkable feature of DTD (Describable Textures Dataset) is that each texture class is associated with descriptive text annotations. These annotations provide text descriptions of the textures, making the dataset suitable for tasks involving both visual and textual information.

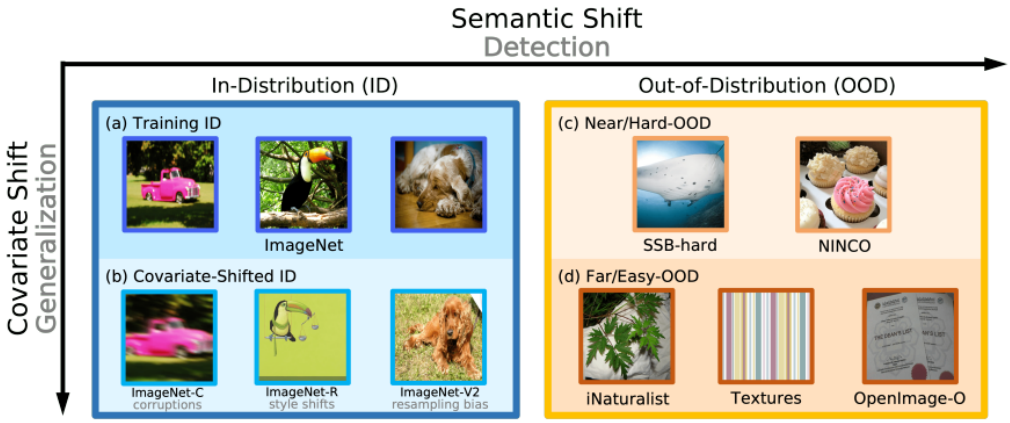

Figure 2.9. Semantic vs Covariate shift from Openood [127].

One popular OOD benchmark [119] proposes the interchangeable assigning of ID and OOD roles between CIFAR datasets [1]. This type of benchmark is called Near-OOD since the two distributions only have semantic shifting and present a reduced level of covariate shifting. This means they only differ slightly in terms of content or characteristics.
One example of Far-OOD benchmarks is CIFAR-10 or CIFAR-100 as ID and MNIST ad OOD. Clearly, switching between the ID and OOD roles causes the classifier to learn a different distribution and produce an alternative OOD detection evaluation.
On the other hand, Far-OOD datasets not only have semantic shifts but also

contain obvious covariate (domain) shifts, making them significantly different from ID datasets. To avoid overlapping semantic OOD samples can be removed, one example is the OOD detection benchmark composed by CIFAR-100 as ID and TinyImageNet as OOD. In this case of Near-OOD distribution, are removed 2502 images.
For a Far-OOD case can be used the SVHN dataset as OOD, with no need for image removal.
Figure 2.9 illustrates how different types of OOD detection can be modeled via data. An ideal system should be robust to the non-semantic covariate shift, improving generalization, while being able to identify semantic shift effectively and performing OOD detection.

Chapter 3

Methodology

This crucial chapter is a comprehensive guide to the methodologies shaping of our research work. In the treatment are involved both Convolutional-based and Transformer-based solutions, adapting these models to a similar inference pipeline for the Out-Of-Distribution detection.
CNNs are used for a variety of applications, including object identification, image classification, and abstract visual reasoning. Their ability to explore and examine spatial locality features, which may be essential for spotting anomalies in picture regions, makes them a popular choice for this kind of task.
Vision Transformers (ViTs) are gaining popularity in the field of computer vision because of their self-attention mechanism that allows them to capture long-range dependencies and global context within an image.
Considering the double nature of our research, each Solution is characterized by an ID classifier (Module A) followed by the OOD detector (Module B), which makes use of different types of ad-hoc exploited data coming from the classifier.
We first present a solution that includes The U-Net [87] model as an ID classifier, defining custom OOD detection modules, and then we move to a solution based on ViT [25] that modifies the precedent one.

3.1 U-Net based approach

The convolutional experimentation, before designing a U-Net solution, lays the foundation on top of the research begun with a ResNet [34] ID classifier. Specifically, employing a ResNet50 that was defined from scratch at first, and subsequently opting for transfer learning with a pre-trained ResNet50 on ImageNet, which is designed for 224p squared resolution images.
ResNet architecture has proven to be highly effective and has achieved state-of-the-art results in various computer vision tasks, including image classification. Key concepts about this architecture concern the introduction of skip connections and a bottleneck structure of convolutional layers within its residual blocks. Skip connections or shortcut connections, are introduced to mitigate the Vanishing gradient problem and speed up the training of very deep neural networks.
Mathematically, given an input x to a residual block, the output y is computed as $y = F(x) + x$ as shown in figure 3.1.

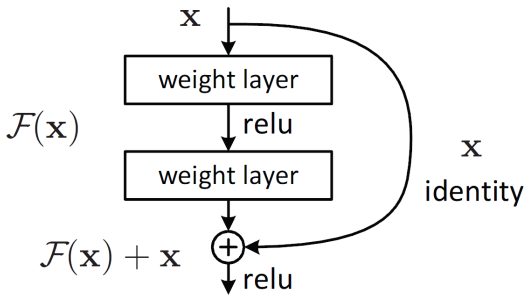

Figure 3.1. ResNet Residual connection.

This model is fine-tuned over all layers using our custom CDDB data to effectively discriminate between real and fake samples from deepfake methods. The model is not only trained using basic augmentation techniques like vertical/horizontal image flip and RandAugment [18], but also includes an interesting technique called CutMix [124]. It involves generating a new training sample by combining two images through a process that comprehends cutting and pasting patches from one image onto another, consequently, the labels are adjusted based on the class mix and in which percentage. This technique helps in improving model performance by enhancing localization ability, increasing training efficiency, and boosting robustness against input corruption. In figure 3.2 we have an example, two images over the same batch are mixed and their labels are adapted in consequence based on the image regions representing the class "Dog" and the class "Cat". This augmentation technique is applied over batches with the 50% of probability.

Figure 3.2. CutMix augmentation for classification.

The ID classifier can produce useful additional data to exploit Out-Of-Distribution detection, and our approach poses its foundation on the interesting intuition from Softmax baseline [36], called **Abnormality module**.
This module in figure 3.3, is a component integrated into neural networks to detect abnormal or out-of-distribution examples. In the foundational work, it's tested on MNIST and is composed by simple fully connected or linear layers. It works by comparing the outputs of the neural network for original examples with the outputs for distorted and altered examples.
The abnormality module is trained to identify instances that deviate from the norm

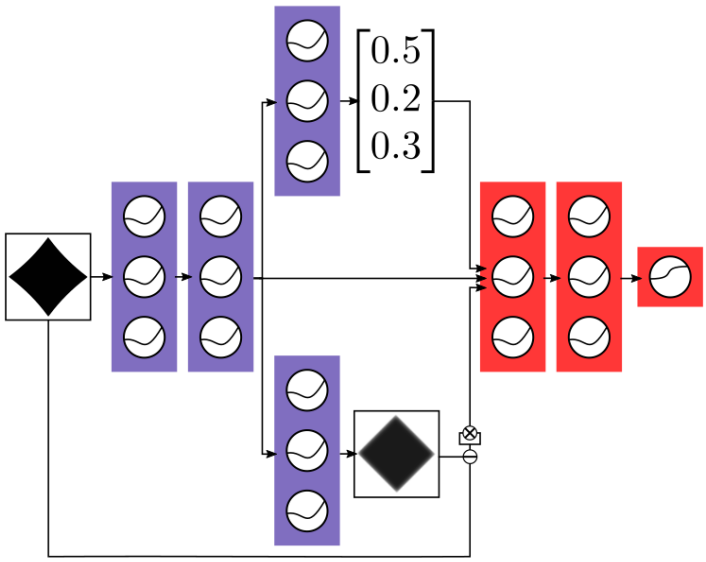

Figure 3.3. Abnormality module by MSP baseline [36].

or are not well-represented in the training data. By detecting abnormal examples, the abnormality module aims to enhance the out-of-distribution detection in machine learning models. The full architecture is composed of two modules, the first (Module A) includes the encoder, a classification head, and a decoder. The decoder is used to reconstruct the input image, then a squared residual is computed. This residual both with the input encoding and the softmax probabilities vector, is passed to the second model (Module B) that stacks these data and discriminates between ID and OOD.
The idea is to re-adapt this technique, used for a toy OOD detection in a more complex task like the one introduced by this work, exploiting as a first attempt the ResNet50 model for In-Distribution classification. This typology of solutions falls under the family of Output-based and Reconstuction-based methods.
From the classic structure of the ResNet50, is extracted the image encoding at the bottleneck just before the final fully connected layer (FC). This model should not only provide the logits for the classification or detection, but its additional goal is to reconstruct the input tensor provided as input following the guidelines introduced. In figure 3.4 we have a clear representation of this structure that we name ResNetEDS (EDS stands for Encoder-Decoder-Scorer). The encoding or latent representation is utilized as input Tensor from the decoder Component.
The decoder comprises six iterations of 2D convolutional transpose layer, GELU activation function, and decoder block. The decoder block in turn is composed of two sequences that include the Convolutional layer, 2D batch normalization, and GELU again. Besides the ResNet backbone, this custom model is made up of the classification head and reconstruction block. The first is initialized assigning values to weights sampled from the Gaussian distribution, while the second, given the presence of convolutional layers, exploits the Kaiming Normal initialization [35].
The classic target function for binary classification problems is the Binary Cross

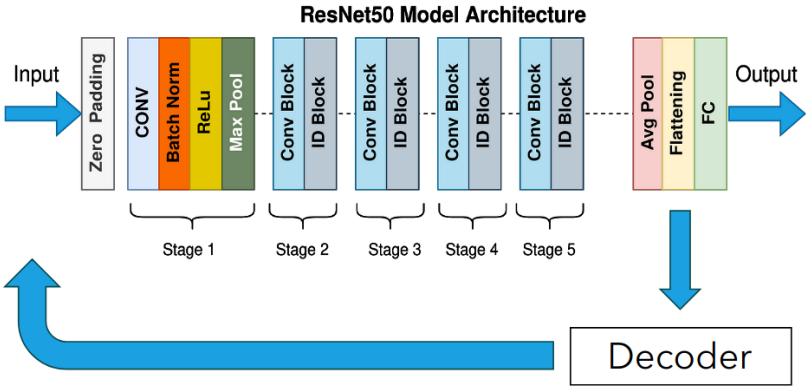

Figure 3.4. ResNet50EDS overall structure.

Entropy or BCE, but in our case, the problem is not only about classification. The goal of this model is double, not only to correctly discriminate between Real and Deepfake images but also to reconstruct with high fidelity the input image. A weighted linear interpolation-like loss function 3.1 is defined for this purpose and is expressed in the following way. Symbols x and y represent respectively the input sample and its label, $\hat{x}$ the reconstructed image, and p the classification prediction.

$$\mathcal{L} = \alpha\,\mathcal{L}_c + \beta\,\mathcal{L}_r = \alpha\,\mathrm{BCE}(y,p) + \beta\,\mathrm{MSE}(x,\hat{x}) \tag{3.1}$$

$$\mathcal{L}_c = \mathrm{BCE}(y,p) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot \log(p_i) + (1-y_i)\cdot\log(1-p_i)\right] \tag{3.2}$$

$$\mathcal{L}_r = \mathrm{MSE}(x,\hat{x}) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{3.3}$$

MSE is the Mean Squared Error loss, the classic function used for regression problems. Whether are respected the following conditions 3.4 the loss function becomes an exact interpolation (LERP formula 3.5).

$$\alpha + \beta = 1, \quad 0 \le \alpha \le 1, \quad 0 \le \beta \le 1 \tag{3.4}$$

$$\mathcal{L} = w\,\mathcal{L}_c + (1-w)\,\mathcal{L}_r \tag{3.5}$$

Anyway, even if in our setup the conditions are respected this is not mandatory, α and β are two new hyperparameters of the learning algorithm. Taking into account that we want to model the classification as a task at a higher priority, and the reconstruction as a secondary/auxiliary target, are assigned the values 0.9 and 0.1 respectively to α and β.
This prototypal network despite achieves satisfactory results in the detection of fake content, is far to be enough accurate in the reconstruction of the image. Important visual information from the input signals is lost during the feedforward process, and results impossible to generate back the images in input. Even Trying different loss weights the outcomes remain pretty similar. Examples of these reconstructions are

Figure 3.5. Reconstructed images by ResNet50EDS.

pointed out in the figure 3.5 where is clearly visible the complete inability in the reconstruction, losing a lot of visual information also related to the orientation.

From the crucial necessary of enhancing the reconstruction capabilities of Module A, and maintaining the overall architecture, we propose the introduction of the U-Net [87] architecture in the pipeline process.
The U-Net model is a convolutional neural network architecture, primarily designed for semantic image segmentation tasks, and is widely used in biomedical image segmentation but has also found applications in other various domains. This model is made up of three main components:

1. Encoder: it consists of several 2D convolutional layers with max-pooling operations, which gradually reduce the spatial resolution while increasing the number of channels. Each Encoder block provides a skip connection for the Decoder at the same dimension level.

2. Bottleneck: block that connects Encoder and Decoder section, it contains multiple 2D convolutional layers to extract high-level features.

3. Decoder: consists of up-sampling layers, usually characterized by 2D transposed convolutions followed by regular 2D convolutions and skip connections. These connections provide fine-grained details and are designed to address challenges such as information loss, and vanishing gradients during training.

Encoder blocks start from the higher spatial dimension and reduce, while the Decoder blocks starting from the lower dimensionality, do the opposite. The final Convolutional layer just maps a higher number of features to the one required by the task. In figure 3.6 is represented a clear diagram of the network, composed of four upsampling/downsampling levels. From now on with the term "levels", we refer to the number of Encoder and Decoder blocks, and we use the level directly in the name of the model to specify this information, i.e. U-Net4 is composed of 4 Encoders and 4 Decoders.

Regarding the development experiments, several types of U-Net networks are designed, changing two main aspects among these models' specifics: the features dimensionality, and the number of levels.

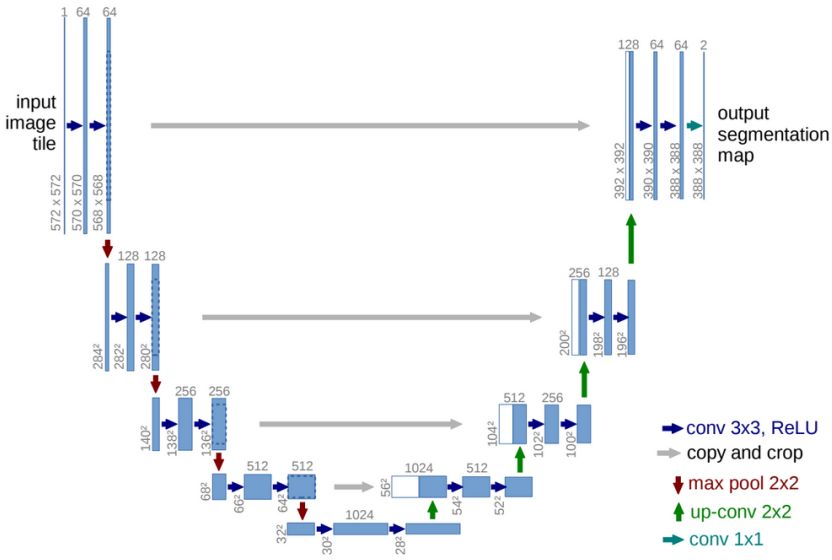

Figure 3.6. Classic U-Net network architecture.

The encoder block consists of a convolutional block followed by a 2D max-pooling (downsample). While the decoder is made up of a transpose 2D convolutional layer (upsample) followed by a convolutional block. This convolutional block is designed only as a features extractor and doesn't perform any spatial transformation, is defined as a double repetition of 2D convolution layer, 2D batch normalization, and ReLU activation function.
All the U-Net based models designed respect two conditions. The first regards the number of features in the bottleneck, considering l the U-Net level, they are in number equal to 2^{4+l}. The second is about the spatial dimension resize after each level, which is always by a factor of 2. Thus, a Decoder block reduces the dimension by half and the Encoder increases the spatial resolution to double.
Considering these relations it's simple to understand the Architecture behind U-Net4 and U-Net5 custom networks. Upon this model categorization, is incorporated the classification head, placed after the bottleneck. This additional branch is composed of a sequence of five linear layers, interspersed with 1D batch normalizations, dropout layers with 30% of dropout probability, and RelU activation functions. We refer to this type of model with the name U-Net Scorer.
Furthermore, a slightly different implementation that introduces the concept of residual connections from ResNet into the encoder block is proposed and included in the research, placing the skip connection after the max pooling operation. U-Net Residual Scorer is the terminology used to mention this architecture.
The target function used to train this model is the one exposed for ResNetEDS, with the exchange of Mean Squared Error (MSE) with the Mean Absolute Error (MAE) for the reconstruction loss 3.6.

$$\mathcal{L}_r = \text{MAE}(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^{N} |x_i - \hat{x}_i| \tag{3.6}$$

This new module A is capable of reaching good performances in the detection task and can reconstruct input images with high accuracy as we can appreciate in figure 3.7. The first row contains the original samples and the second their reconstruction. What is suggested for the usage of these architectures, is to accordingly choose

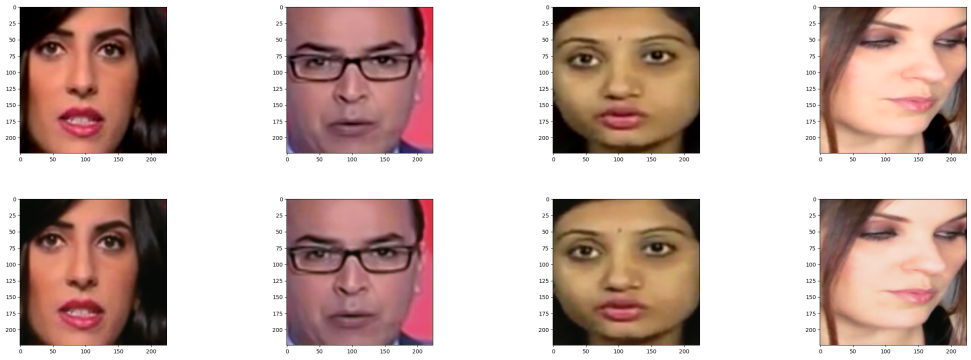

Figure 3.7. Reconstructed images by U-Net5 Scorer.

the network level based on the input image resolution. To avoid problems with higher dimensionality of the encoding, it's advisable to utilize U-Net 4 Scorer with 112p images, and U-Net 5 Scorer with 224p images. This is not strictly required but an oversized encoding would lead to different problems for module B like data unbalancing, long convergence time, etc.

Taking into account the OOD detection research present in [23] we propose an additional ID classifier, that mixes the U-Net Scorer with the confidence estimation. This work introduces a technique for learning confidence estimates without requiring labels, showing improvements in out-of-distribution detection tasks. Confidence represents the neural network's belief or certainty in its predictions. It is quantified as a scalar value between 0 and 1, where 0 indicates low confidence (uncertainty) and 1 indicates high confidence (certainty) in the model's prediction for a given input. For what concern the alteration of the model structure, the confidence estimation branch is added after the penultimate layer of the original network, such that both the confidence branch and the prediction branch receive the same input. This branch is simply built from a dense layer with a single neuron output which is managed through the sigmoid activation function.
Figure 2.7 from chapter 2 shows clearly this architecture. Confidence estimation requires an adaptation of the training algorithm since with this configuration we have a triple target for classification, reconstruction, and producing accurate confidence prediction. Following the direction of the original work, the training is characterized by "hints" to the network, used for an effective learning of the confidence concept. Given c the confidence estimation, p_i the probability inferred by the model for class i, y_i its ground truth, the new interpolated prediction probability p'_i is computed according to formula 3.7.

$$p'_i = c \cdot p_i + (1 - c) \cdot y_i \tag{3.7}$$

These hints are provided under the Bernoulli distribution with success probability

at 50%. The confidence loss is simply defined with the expression 3.8. It prevents the networks from minimizing the loss by always choosing $c = 0$ and receiving the full ground truth value.

$$\mathcal{L}_{conf} = -\log(c) \tag{3.8}$$

Thus, this new U-Net based model version produces 4 different outputs: classification probabilities, image reconstructions, image encoding, and prediction confidence. The full target takes the form 3.9.

$$\mathcal{L} = \alpha\,\mathcal{L}_c + \beta\,\mathcal{L}_r + \lambda\,\mathcal{L}_{conf} \tag{3.9}$$

Confidence loss is weighted by λ hyperparameter, which balances the previous task loss and the confidence one. Using an additional hyperparameter γ called budget, λ is updated during training, to avoid convergence of c to a unity for all samples.

After this comprehensive description of Module A, we will now examine Module B, which builds upon the functionalities of Module A and applies its data to perform Out-Of-Distribution Detection. Our work introduces several architectures for this purpose.
As said previously, The different learned meta-data is exploited from the abnormality module, customizing the primary idea introduced in [36]. The name comes from the term "abnormal", which is used about samples that belong to a data distribution unknown from the classifier (OOD data).
This novel module takes advantage of all the information inferred during the training of the deepfake detector. Along the module B discussion, we assume the use of inputs such as classification logits, input encoding, and reconstructed image, omitting the solution with the inclusion of confidence, since can be trivially included in the overall scheme. Logits are converted to a probability distribution thanks to Softmax output activation function. This distribution incorporates information about the biased model's confidence in its prediction, which has been proven to be insufficient to effectively discriminate between ID and OOD samples. For a confident prediction the expected behavior is characterized by probability values at the extreme of the range, so closer to 0 or 1.
Instead, the image reconstruction is employed in the computation of the residual, squaring its difference with respect to the original input image. The clue behind the residual data is about the reconstruction capability of the decoder, which is expected to decrease whether encounters data not relative to the known distribution.
Finally, the encoding gives a latent representation of the image from the classifier's point of view and can be a significant discriminant for very different input samples.
To be noticed, a simple consistent operation for confidence inclusion is to stack this estimation directly in the softmax probability vector, keeping the following intuitions valid.
A crucial aspect to discuss, concerns data applied to train the OOD detector. Both ID and OOD data are necessary to train this module, and we can distinguish between two training methodologies that can be utilized by this model. In both cases, we consider the ID train set as the one employed by the classifier, but they differ in the nature of OOD data. The first methodology uses five techniques to synthesize OOD samples, altering the ID images. Specifically: Blurring from Gaussian filter, JPEG

compression at high loss (10% of quality), inclusion of normal noise and normal noise with random standard deviation (distortion effect), and inclusion of noise with $\frac{\pi}{2}$ random image rotation. The results are presented in figure 3.8. Using these data, we exploit an unsupervised technique to train the abnormality module.

In contrast, The second methodology integrates available OOD samples with the

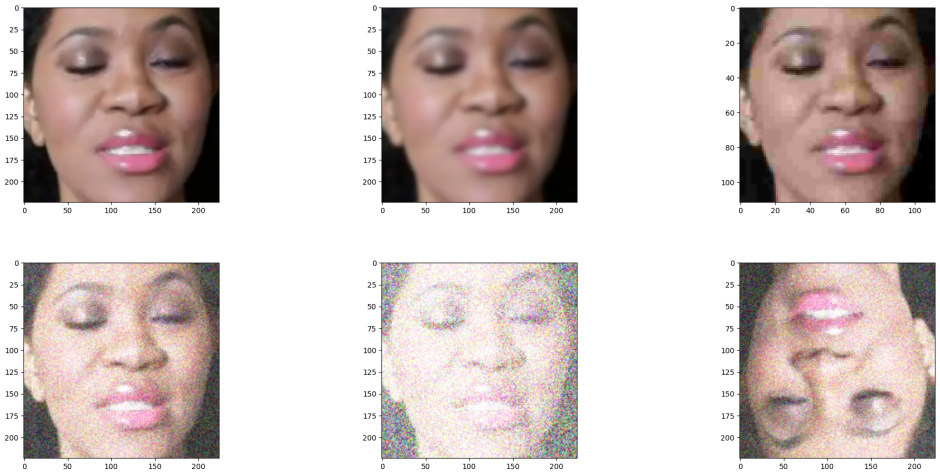

Figure 3.8. Alteration techniques for synthesized OOD data.

From left to right and top to bottom: original sample, blur effect, JPEG compression, normal noise, normal noise distorted, normal noise and random rotation

role of real outliers, defining a supervised procedure.In the Results chapter 5 is also involved a mix of the two OOD sets.

3.1.1 Abnormality modules

The first prototypal version of the abnormality module, which we will refer to by the name **basic**, implements the simplest solution possible, quite similar to the original module from the Softmax baseline [36]. The auxiliary data from module A comes as input for this network, which flattens the residual and stacks all together in a single tensor. Subsequently, this new data is elaborated through a multilayer perception (MLP) composed of 5 fully connected layers, 1D batch normalization (BN), and GELU activation function. Finally, the single outcome logit is fed into the sigmoid activation function, producing the risk value. Risk value is directly associated with the abnormality of the sample. In this section, the dimensions come from the consideration of a U-Net4 Scorer as Module A.

The general architecture of the basic abnormality module is schematized in figure 3.9, and in table 3.1 we have a summary of the network for 112p resolution input images.

This model is trained using the Binary Cross Entropy target 3.2. Looking at the table we find out the total number of input components after the concatenation, which is around 50k. In the case of U-Net5 Scorer and 224p images, the total amount

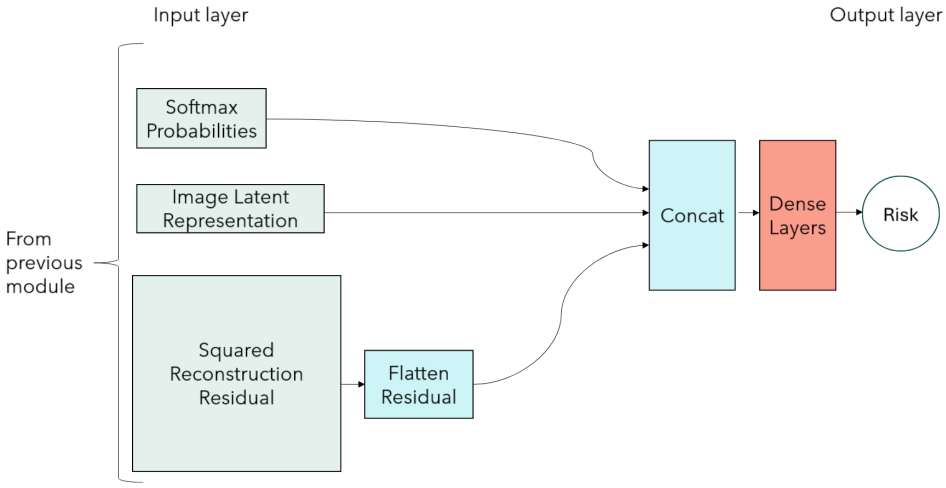

Figure 3.9. Basic Abnormality module: scheme.

Layer/Module	Output Shape	Branch
Input 1	(batch size, 2)	Softmax
Input 2	(batch size, 12544)	Encoding
Input 3	(batch size, 3, 112, 112)	Residual
Flatten	(batch size, 37632)	Residual
Concatenation	(batch size, 50178)	Risk
FC 1 + BN + GELU	(batch size, 4096)	Risk
FC 2 + BN + GELU	(batch size, 1024)	Risk
FC 3 + BN + GELU	(batch size, 512)	Risk
FC 4 + BN + GELU	(batch size, 128)	Risk
FC Final	(batch size, 1)	Risk

Table 3.1. Abnormality module Basic: architecture table.

of elements in the concatenated tensor becomes about 175k.
Given l the U-Net level, R the image resolution (assuming squared resolution, an equal number of pixels for width and height), and C the number of input channels, we can compute the following sizes in general, by considering these relations.

$$\text{Encoding dimension} = R^2 \cdot 2^{4-l} \tag{3.10}$$

$$\text{Flatten Residual dimension} = R^2 \cdot C \tag{3.11}$$

The huge dimensionality after concatenation which characterized this model implies a high computational cost and a very slow convergence to the target. Moreover, the significant difference in sizes between each single auxiliary data tends to overshadow the different typologies of data exploited. Softmax probability suffers the most from this problem.
Three evolved modules arise from this basic version, exploiting the usage of additional encoders, to make the model more efficient and obtain greater performances balancing data.

What is proposed as the first variant **V1**, slightly modifies the basic abnormality module reducing the residual size before the concatenation with the other inferred data from the ID classifier. This is achieved by involving a new sub-module called Encoder block, whose goal is to reduce spatial dimensionalities and increase the number of features. This block is composed of a triple sequence of 2D Convolutional layer, 2D batch normalization, and GELU activation function. Each block defines an output number of feature maps, which is used to augment the number of input channels from the first layer. Instead, the second conv layer is responsible for downsampling using a stride equal to 2. All the convolutional layers are characterized by a 3×3 kernel and a single extra pixel for padding.
Three encoding blocks are used to reshape the residual information, then follow the same approach explained before with the concatenation and the inclusion of several sequences of fully connected layers, 1D batch normalizations, and GELU activation functions in the MLP block. Actually, this architecture contains a sequence more in the MLP block with respect to the basic version.
In figure 3.10 and table 3.2 its overall structure.

Using this encoding in the abnormality module, the residual information is re-

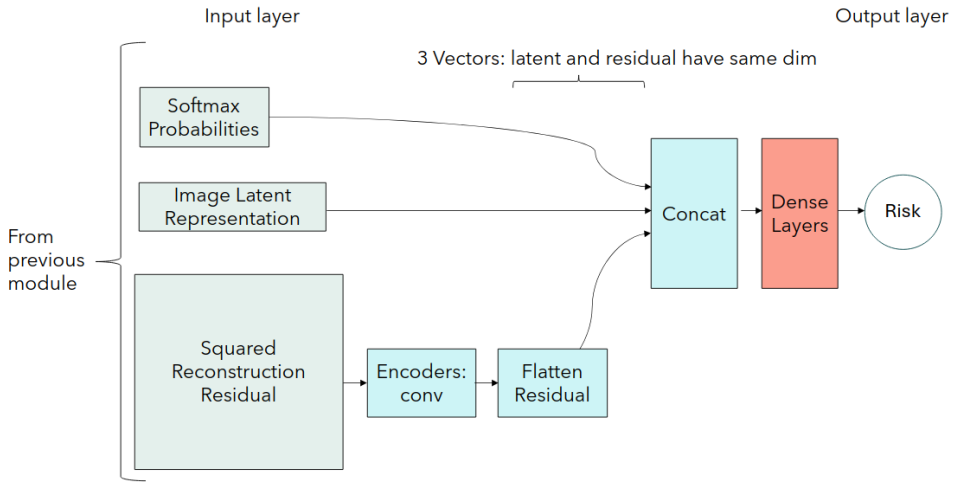

Figure 3.10. Abnormality module Encoder V1 and V2: scheme.

duced to the same dimension as the ID classifier encoding. The unbalancing respect softmax probabilities still exists but is eliminated between encoding and residual. Computational complexity and training time are decreased, such as the number of model parameters which have been halved to 113 million.

An advanced version named **V2** pushes further on the residual encoding, adding no blocks to downsample but changing the encoding directly on the U-Net Scorer model. This modification comes for free extracting the encoding not from the bottleneck as before, but rather from the first convolutional layer present in the classification/scorer branch. In order to keep the same size for encoding and residual, is necessary to include an additional encoding block in the residual branch. The overall architecture remains identical to the one in figure 3.10, the layers and internal

Layer/Module	Output Shape	Branch
Input 1	(batch size, 2)	Softmax
Input 2	(batch size, 12544)	Encoding
Input 3	(batch size, 3, 112, 112)	Residual
Encoder block 1	(batch size, 16, 56, 56)	Residual
Encoder block 2	(batch size, 32, 28, 28)	Residual
Encoder block 3	(batch size, 64, 14, 14)	Residual
Flatten	(batch size, 12544)	Residual
Concatenation	(batch size, 25090)	Risk
FC 1 + BN + GELU	(batch size, 4096)	Risk
FC 2 + BN + GELU	(batch size, 2048)	Risk
FC 3 + BN + GELU	(batch size, 1024)	Risk
FC 4 + BN + GELU	(batch size, 512)	Risk
FC 5 + BN + GELU	(batch size, 128)	Risk
FC Final	(batch size, 1)	Risk

Table 3.2. Abnormality module Encoder V1: architecture table.

dimensionalities change according to table 3.3. The number of model parameters decreases to around 22 million.

Layer/Module	Output Shape	Branch
Input 1	(batch size, 2)	Softmax
Input 2	(batch size, 1568)	Encoding
Input 3	(batch size, 3, 112, 112)	Residual
Encoder block 1	(batch size, 4, 56, 56)	Residual
Encoder block 2	(batch size, 8, 28, 28)	Residual
Encoder block 3	(batch size, 16, 14, 14)	Residual
Encoder block 4	(batch size, 32, 7, 7)	Residual
Flatten	(batch_size, 1568)	Residual
Concatenation	(batch_size, 3138)	Risk
FC 1 + BN + GELU	(batch size, 1024)	Risk
FC 2 + BN + GELU	(batch size, 512)	Risk
FC 3 + BN + GELU	(batch size, 128)	Risk
FC Final	(batch size, 1)	Risk

Table 3.3. Abnormality module Encoder V2: architecture table.

The last abnormality module version **V3** takes the concept of balancing between the auxiliary data to the extreme. This is performed by introducing 2 MLP blocks both on the Encoding and Residual branches. Starting from Abnormality module V1, so with the large encoding in input and 3 Encoding blocks for the residual, the new MLPs have the same structure. They are built repeating 6 times the following sequence: dense layer, 1D batch normalization, and GELU activation function.
After the concatenation, the size of the resulting tensor is highly reduced, such that

the risk MLP block is constituted just from a single dense layer and the sigmoid activation function. Figure 3.11 and table 3.4 summarize the general picture and add other information.
This new network is made up of 125 million parameters. The huge reduction of encoding and residual before concatenation to 10 elements, is chosen to give a strong balance between the 3 inputs. However, a more mild reduction would be an interesting additional experiment to investigate further.

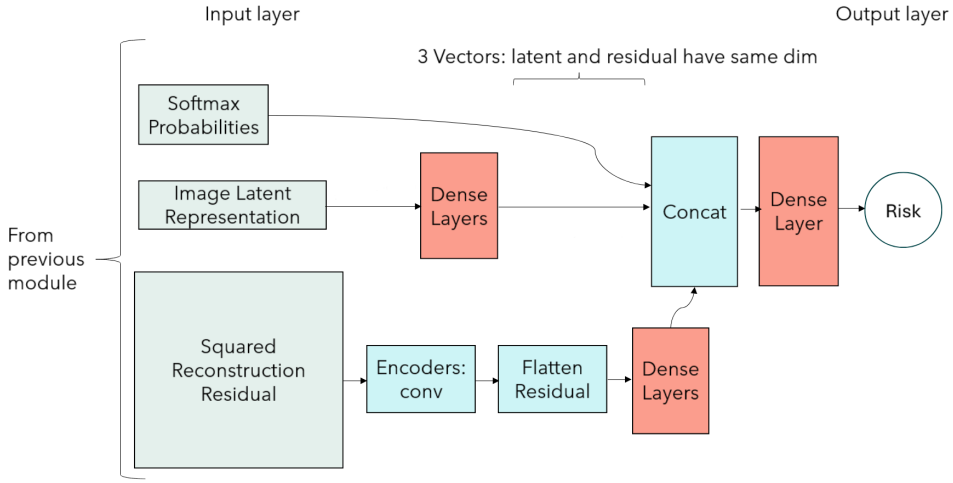

Figure 3.11. Abnormality module Encoder V3: scheme.

It's important to notice, and is valid for all abnormality modules exposed, that whether we use images at higher resolution, i.e. passing from 112p to 224p, is necessary to include an additional encoder block for the residual to keep all the tensors shapes. The encoding remains at the same dimensionality whether the initial assumption of using 112p images with Unet4-like and 224p with Unet5-like models is respected.

Layer/Module	Output Shape	Branch
Input 1	(batch size, 2)	Softmax
Input 2	(batch size, 12544)	Encoding
Input 3	(batch size, 3, 112, 112)	Residual
Encoder block 1	(batch size, 16, 56, 56)	Residual
Encoder block 2	(batch size, 32, 28, 28)	Residual
Encoder block 3	(batch size, 64, 14, 14)	Residual
Flatten	(batch size, 12544)	Residual
FC R1	(batch size, 4096)	Residual
FC R2	(batch size, 2048)	Residual
FC R3	(batch size, 1024)	Residual
FC R4	(batch size, 512)	Residual
FC R5	(batch size, 128)	Residual
FC R6	(batch size, 10)	Residual
FC E1	(batch size, 4096)	Encoding
FC E2	(batch size, 2048)	Encoding
FC E3	(batch size, 1024)	Encoding
FC E4	(batch size, 512)	Encoding
FC E5	(batch size, 128)	Encoding
FC E6	(batch size, 10)	Encoding
Concatenation	(batch_size, 22)	Risk
FC Final	(batch_size, 1)	Risk

Table 3.4. Abnormality module Encoder V3: architecture table.

3.2 Vision Transformer based approach

In the preceding sections, we extensively explored the full pipeline composed of Module A and Module B, respectively by the U-Net based ID classifier and one of the Abnormality modules proposed.
Building upon the strengths of this structure, we now introduce a novel approach inspired by the work proposed by Cultrera et al. [19]. The core behind this new methodology, lies in the substitution of the Convolution network in Module A, with a Vision Transformer (ViT) model. The Transformer is a neural network Encoder-Decoder architecture introduced in the paper "Attention is All You Need" by Vaswani et al. [106] that revolutionized natural language processing tasks. Self-attention mechanism is the essential component, that enables the model to effectively capture long-range relationships by allowing it to weigh different parts of the input sequence differently.
The Vision Transformer [25] extends the Transformer architecture to computer vision tasks. Instead of processing sequences of words, ViT processes images as sequences of patches. It divides an input image into fixed-size patches (tokens), linearly embeds them, and elaborates them through a Transformer Encoder. Its schematic model is shown in picture 3.12. The latent representation extracted can be used from a feedforward block for the classification, i.e. the classification task is carried out including at the beginning of the input sequence the [CLS] token, processing its

transformed output with a scorer head.
The attention mechanism allows the model to selectively focus on different parts of the input sequence when making predictions. Rather than considering every aspect equally, attention gives each element an individual weight, either highlighting or de-emphasizing it according to how relevant it is in the actual context.
Considering the potential of such networks in the computer vision field a new work

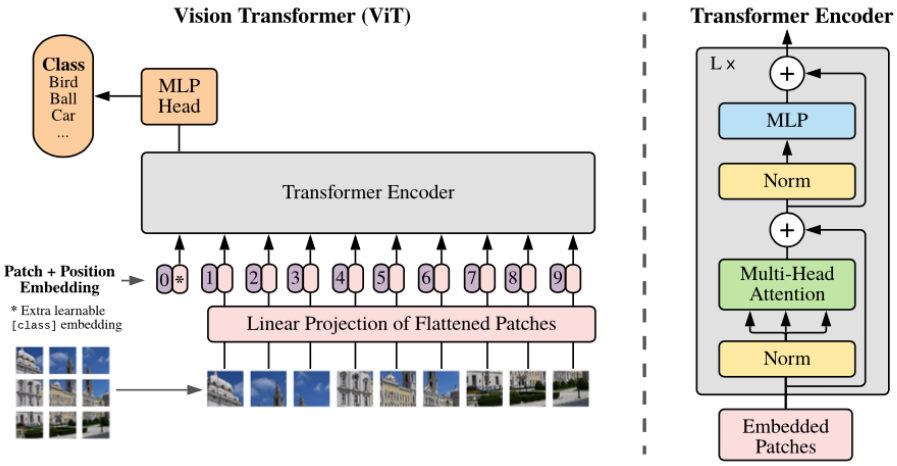

Figure 3.12. Vision Transformer (ViT) network.

pipeline is proposed following the general structure exposed using the U-Net. The crucial clue resides in the employment of the attention map or heatmap information to perform OOD detection. Keeping the same structure ID classifier + OOD detector, the intuition is to substitute the image residual computation via the Decoder with the extraction of the attention map. Successively train an Autoencoder [89] model to learn the distribution of these maps and reconstruct them in the most similar way to the original maps. What follows is the same procedure exposed before which exploits the residual calculus from the heatmap and its reconstruction, together with the latent representation of the image and the Softmax probabilities from the classification head. These OOD metadata are used to feed the abnormality module. The full development of this ViT solution is restricted to images with 224×224 resolution.
Based on the accumulated experience gained with the previous approach, we have tested several ViT-based networks to perform deepfake detection effectively.
After several attempts in the definition and training of ViT base and lighter versions from scratch, transfer learning and consequent fine-tuning are adopted, as done for ResNet. This choice speeds up the training of the ID classifier. In the specific, the pre-trained models like "vit-base-patch16-224" are obtained from TIMM [115] module and are pre-trained on ImageNet.
The ViT base model reaches good performances in the detection of deepfake examples, but during training tends to uniformly assign the same attention value for all the patches, losing significant information about which patch is really important to perform the detection.
To mitigate this difficulty a different model called Data-efficient Image Transformer

or DeiT [102], is integrated in the pipeline substituting the ViT base. DeiT is a transformer-based model architecture designed to achieve strong performance in image classification tasks keeping computational efficiency and requiring a reduced number of labelled data. It maintains the same procedures as ViT and employs a training technique called knowledge distillation. The model is trained with a teacher model (typically a larger and more accurate model) to transfer knowledge from the teacher to the smaller student model, the Deit itself. This leads to improved generalization and competitive performance with lower processing requirements for the smaller model. The network is trained including the teacher loss usually a combination of Cross entropy loss and KL divergence between the teacher and the model distribution. In figure 3.13 we can appreciate a scratch of this model.

Also for this case, pre-trained models are employed. The development focuses on the

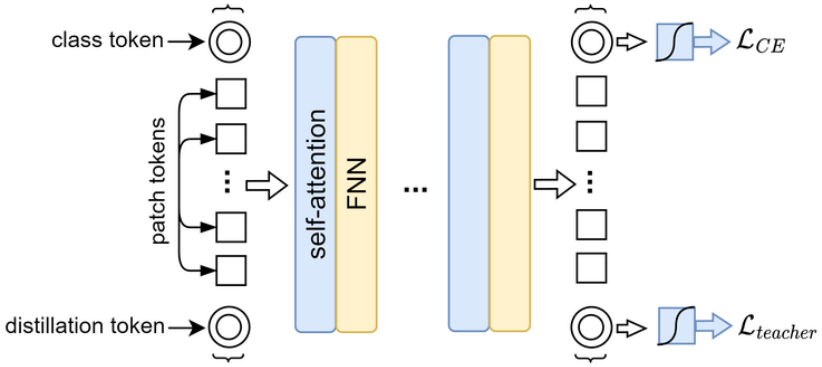

Figure 3.13. Data Efficient Image Transformer (DeiT) network.

usage of computationally non-heavy DeiT networks in tiny and small configurations. Their technical details are exposed in the table 3.5. From now on, we consider the

	DeiT Small	**DeiT Tiny**
Embedding Size	384	192
Input image	$3 \times 224 \times 224$	
Patch Size	16	16
Num Heads	6	3
Num Layers	12	12
Dim Head	64	64
Dropout %	10	10

Table 3.5. DeiT Small and DeiT Tiny Configurations.

solution with the DeiT tiny configuration, which is particularly advantageous for faster training and convergence.

The image encoding is extracted from the transformer encoder before the MLP block, with features relative to the [CLS] token. Softmax vector is the application of the activation function on the classification logits, as usual.

The attention map is computed by breaking down the multi-head self-attention in

the last transformer block. This phase begins with the embedding of the penultimate block, query Q, key K, and value V matrices are first separated. The full attention matrix A is computed as $A = Q \cdot K^T \cdot s$, with s the block scaling factor, a scalar used to normalize the dot product. This attention matrix presents a heatmap for each token. Thus, it's extracted and saved the attention map relative to the [CLS] token, removing attention values about the classification token itself and the distillation token. The resulting tensor has shape batch size × n.heads × (n. tokens -2). Subsequent operations consist of the meaning computation over the attention heads and a reshaping to obtain a tensor of the form $1 \times \sqrt{\text{n. patches}} \times \sqrt{\text{n. patches}}$, representing a small grayscale attention map. As last step, the map is upsampled to obtain the original image shape, in this case to 224p resolution through bilinear interpolation.
Figure 3.14 shows an example of an extracted attention map over a sampled image. Moreover, to allow the possibility of additional tests, this modified feedforward of the last block is designed to memorize the attention value relative to all the patches, organizing a grayscale image and interpolating it as before. Differently from

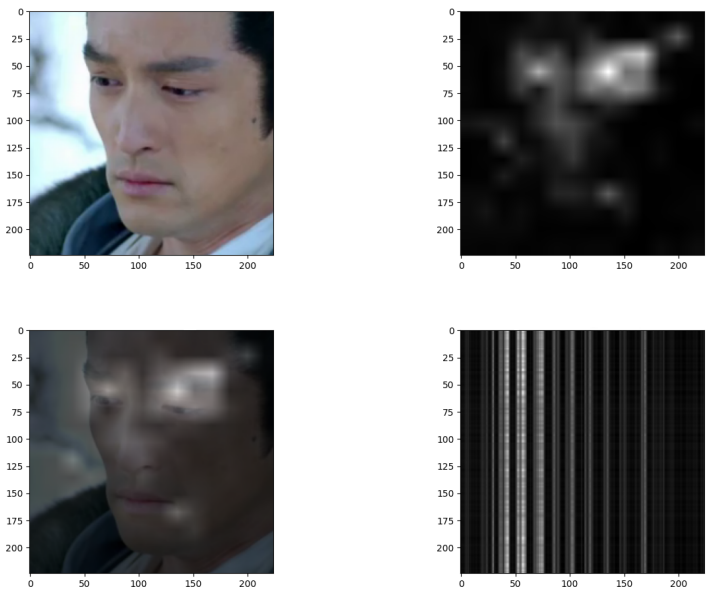

Figure 3.14. Attention heatmap from DeiT model.

From left to right and top to bottom: the original sample, its attention map, original sample blended with the attention, attention map of tokens about patches.

U-Net solution, this new approach trains in a separate fashion the classifier and the reconstruction model. The network architectures proposed for the reconstruction procedure are the classic AutoEncoder (AE) and the Variational AutoEncoder (VAE), represented in figure 3.15.

Tables 3.6 and 3.7 display the two models structure. All the convolutional layers are defined by a squared 4×4 kernel, stride equal 2, and a single pixel of padding. In both cases, the output activation function is the Sigmoid and the slope of the

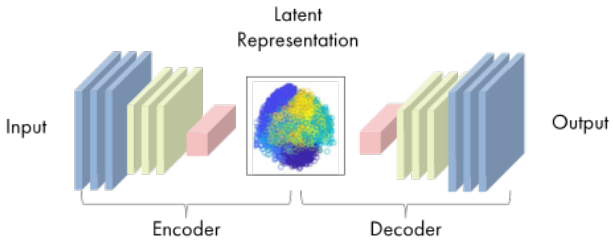

Figure 3.15. AutoEncoder architecture.

Leaky ReLU is 0.2. The number of filters can be simply inferred by looking at the second dimension of the output shape.
The AE is trained using the classic MAE 3.6 loss function. Instead, the VAE loss function is defined in the equation below, which represents the reconstruction loss via MAE summed to KL divergence. In the specific N are all the input components, x_i is the input sample, $\hat{x}_i$ its reconstruction, σ and μ are respectively the mean and standard deviation for the reparametrization trick.

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}|x_i - \hat{x}_i| + -0.5 \times \sum(1 + \log(\sigma^2) - \mu^2 - \sigma^2) \tag{3.12}$$

We conclude this section, by presenting the reconstruction results of the two models in figure 3.16, in reference to the previous sample in 3.14.

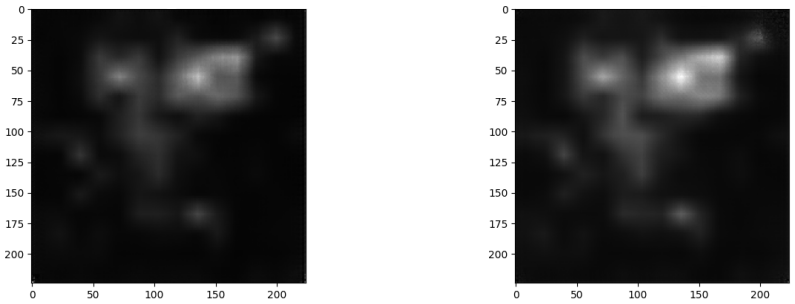

Figure 3.16. Reconstructed heatmaps.

From left to right: AE & VAE reconstructions.

Layer/Module	Output Shape	Block
Input	(batch size, 1, 224, 224)	Encoder
Conv2D + BN + ReLU	(batch size, 32, 112, 112)	Encoder
Conv2D + BN + ReLU	(batch size, 64, 56, 56)	Encoder
Conv2D + BN + ReLU	(batch size, 128, 28, 28)	Encoder
Conv2D + BN + ReLU	(batch size, 256, 14, 14)	Encoder
Conv2D + BN + ReLU	(batch size, 512, 7, 7)	Encoder
ConvTranspose2D + BN + ReLU	(batch size, 256, 14, 14)	Decoder
ConvTranspose2D + BN + ReLU	(batch size, 128, 28, 28)	Decoder
ConvTranspose2D + BN + ReLU	(batch size, 64, 56, 56)	Decoder
ConvTranspose2D + BN + ReLU	(batch size, 32, 112, 112)	Decoder
ConvTranspose2D	(batch size, 1, 224, 224)	Decoder

Table 3.6. AutoEncoder architecture.

Layer/Module	Output Shape	Block
Input	(batch size, 1, 224, 224)	Encoder
Conv2D + BN + Leaky ReLU	(batch size, 32, 112, 112)	Encoder
Conv2D + BN + Leaky ReLU	(batch size, 64, 56, 56)	Encoder
Conv2D + BN + Leaky ReLU	(batch size, 128, 28, 28)	Encoder
Conv2D + BN + Leaky ReLU	(batch size, 256, 14, 14)	Encoder
Conv2D + BN + Leaky ReLU	(batch size, 512, 7, 7)	Encoder
Flatten	(batch size, 25088)	Encoder
Linear	(batch size, 1024)	Encoder
Linear + Leaky ReLU	(batch size, 25088)	Decoder
Unflatten	(batch size, 512, 7, 7)	Decoder
ConvTranspose2D + BN + Leaky ReLU	(batch size, 256, 14, 14)	Decoder
ConvTranspose2D + BN + Leaky ReLU	(batch size, 128, 28, 28)	Decoder
ConvTranspose2D + BN + Leaky ReLU	(batch size, 64, 56, 56)	Decoder
ConvTranspose2D + BN + Leaky ReLU	(batch size, 32, 112, 112)	Decoder
ConvTranspose2D	(batch size, 1, 224, 224)	Decoder

Table 3.7. Variational Autoencoder architecture.

3.2.1 Abnormality modules

As anticipated, abnormality modules follow the same structure introduced before for the U-Net approach. We now exploit the reconstruction nature of our technique, computing the squared residual from the attention map and the reconstruction from the VAE model. The residual is flattened and stacked with softmax probabilities and image encoding. Using the tiny DeiT as ID classifier, the encoding length is equal to the embedding size of the model, thus, a vector of 192 elements. Using such small-size encoding, the only compatible abnormality module is the V2. Besides this, further experiments involve the use of V3 in this ViT pipeline. Still holds what was previously stated, working with 224p images and no more with 112p images, the abnormality architecture exposed for the U-Net solution, includes an additional

Encoder block in the residual branch.
Since the variations are just in terms of input image encoding, passing respectively in V2 and V3 from 1568 and 12544 to 192 components, and the additional block to encode the residual, the abnormality modules are not explained in detail again here. The application of this new Module A causes one property loss of the previous OOD detection architecture. In this approach, the equality in size between the encoding and residual information in the concatenation phase is not respected. For this reason, V2 and V3 are the solutions that best suit this case, mitigating as possible any related problem.

Chapter 4

Experimental settings

The settings in which we trained and tested our algorithms are shown in this chapter. In detail, we provide the necessary knowledge for a consistent understanding of the data used to train the models, with a description of the starting dataset and pre-processing operations, and then we move on to the setup description for the learning phase. To conclude, we prepare the reader for the next chapter introducing the metrics used for evaluating the proposed method.

4.1 Dataset

The dataset choice is one of the fundamental bases in any machine learning-related task. There's a quite large quantity of deepfake detection datasets available for research, as we have seen in chapter 2. Our work, proposed a novel usage of CDDB (Continual Deepfake Detection Benchmark) Dataset [57] by Huang et al. Adapting the full dataset to the requirements of our task.
As suggests its name, this dataset provides a benchmark designed to evaluate deepfake detection methods in a continual learning setting. It offers a realistic and challenging environment for testing deepfake detection algorithms. Moreover, CDDB introduces an exhaustive evaluation procedure categorizing evaluation protocols into easy, hard, and long sequences, and offers a set of appropriate measures for assessing detection performance for a continual learning approach. The original paper of CDDB goes into details of CIL (classic Incremental Learning) studying three prominent categories of methods: gradient-based, memory-based, and distillation-based.
Every technique is examined in the framework of continuous learning for deepfake detection, giving particular attention to how they improve detection performance over several classes and deal with the problem of catastrophic forgetting. Also, deepfake detection performances in a continual learning setting are reported for each technique, on existing methods.
We now focus on how the dataset is structured, exposing which kind of data contains, and how is divided. This dataset is built by joining a diverse collection of deepfakes from both known and unknown generative models.
CDDB includes 11 deepfake models divided into 3 different groups by source: GAN models, non-GAN models, and unknown models. A graphical representation of the dataset is shown in figure 4.1. Previous opensource deepfake works included in

this collection are: CNNDetection [112] (which in turn include examples from face forensics++[88] and WhichFaceReal [4]), GanFake Detection [70] and WildDeepFake [131]. The 3 categories contain the following techniques and datasets:

Figure 4.1. CDDB data collection.

In Green the real samples, while in red fakes.

1. GAN models, this group includes fake images synthesized by 6 GAN models, using ProGAN [43], StyleGAN [44], BigGAN [12], CycleGAN [130], GauGAN [77], StarGAN [14]. These models are trained on ImageNet [21], LSUN [122], COCO [65], CelabA [68].

2. Non-GAN models, comprehends 8 different techniques: Generative flow [48], Cascade refinement Networks (CRN) [13], Implicit Maximum Likelihood Estimation (IMLE) [59], Second Order Attention Networks [20], and from faceforensics++ we have: Deepfake [60], Face2Face [100], FaceSwap [108], and NeuralTexures [99]. These models are trained using CelabA [68]and GTA [85] datasets.

3. Unknown models, last group containing images from two unknown generative models, that are collected from WildDeepFake [131] and WhichFaceReal [4], which both contains real and deepfake images/videos from internet.
   This set of examples is provided to more closely mimic real-world scenarios where it may not be possible to identify the source model of observed deepfakes.

The number of real and fake samples is balanced, which represents an optimal property to face binary classification on these two classes. Instead, the support of each sub-category can be summarized in the below table 4.1.
The total number of images contained in this dataset version is around 82k.
This dataset represents an advanced work in the field of deepfake detection, Incorporates deepfakes from both known and unknown sources, providing a varied collection to simulate real-world scenarios. Furthermore, Allows us to test and compare existing deepfake detection methods.
The data nature of this collection may require advanced deepfake detection algorithms to achieve high accuracy, also considering the complexity of handling deepfakes from unknown sources.

Group	Deepfake sources	# samples
GAN model	StyleGAN	9.58k
	BigGAN	3.2k
	CycleGAN	2.1k
	GauGAN	8k
	StarGAN	12k
Non-GAN model	Glow	12k
	CRN	10.2k
	IMLE	10.2k
	SAN	352
	Deepfake	4.3k
Unknown model	WhichFaceReal	1.6k
	WildDeepFake	8.27k

Table 4.1. CDDB samples by technique.

4.2 Data Preprocessing

CDDB [57] is an overall balanced dataset, considering each deepfake technique included, real and fake samples are almost perfectly equal in number. This collection shows up as a directory-based or directory-structured dataset. In this type of representation, the dataset is organized into directories and subdirectories, with each directory corresponding to a specific class or category, and the data samples belonging to that class stored within that directory. The complete organization is represented in figure 4.2. CDDB is only predisposed for training and testing, the validation set is not present in the default version of this collection. The train data make up about 70% of the whole set, with 57,790 samples, the remaining for testing is around 30% of the full CDDB, with 24,057 elements.
In the table 4.2 is shown the exact number of samples for these two sets, and the contribution of each technique.

Deepfake sources	# samples train	# samples test
StyleGAN	7,188	2,394
BigGAN	2,400	800
CycleGAN	1,572	524
GauGAN	6,000	2,000
StarGAN	7,200	4,800
Glow	7,200	4,800
CRN	7,658	2,552
IMLE	7,656	2,552
SAN	260	90
Deepfake	3,248	1,082
WhichFaceReal	1,200	400
WildDeepFake	6,208	2,063

Table 4.2. CDDB train and test sets.

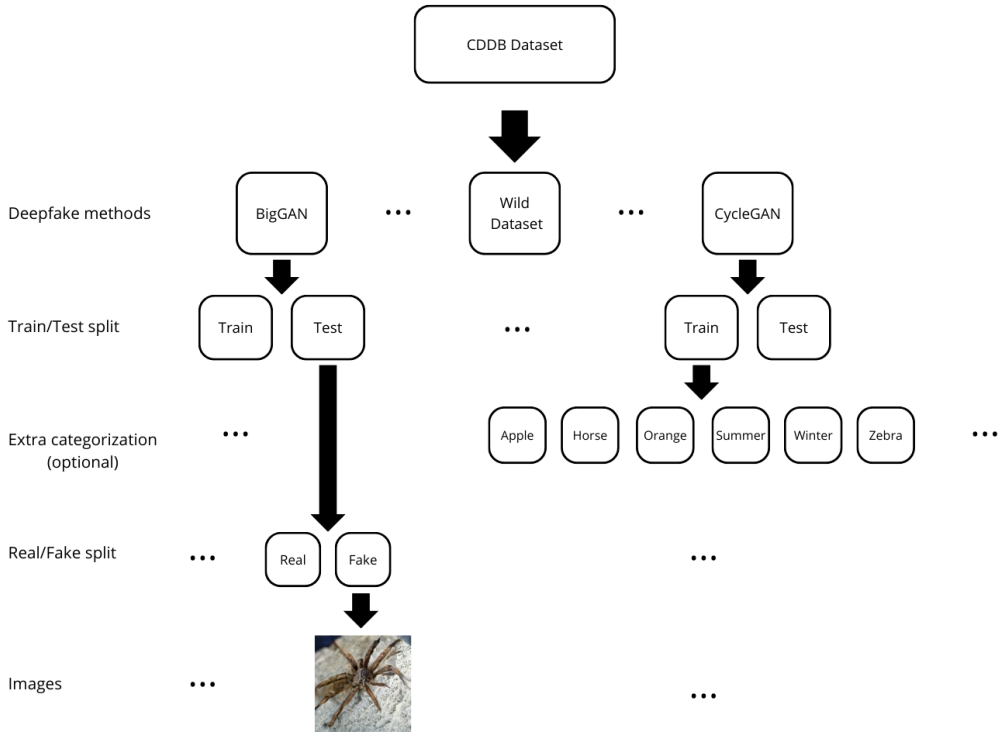

Figure 4.2. Directory Structure of CDDB dataset.

The main goal of this data processing step is to transform a continual learning dataset, with different binary classifications, into a single classification dataset. Given the structure of the original records, has been not difficult to define four different versions of the CDDB dataset. For the pure Deepfake detection task, the binary and binary-partial CDDBs are designed, the first includes the whole set of data available, while the second proposes a division between In-Distribution and Out-Of-Distribution.
ID and OOD split is necessary to effectively face Deepfake and OOD detection in the same work pipeline. ID data is used to train the deepfake detector, while OOD data is used for the OOD detection analysis. This data separation is thought to follow three different strategies. In the **content scenario**, the semantic content of the data is the discriminant to move data in the In or Out distribution set. For each Deepfake technique, the original dataset in most cases separates well the data also by the image's subject. For example are present real and fake sets of faces, apples, oranges, horses, cats, virtual environments, winter, etc. The first study case that we define, is using the faces content as ID and the rest of the set as OOD. It's important to notice that several DF techniques are not organized by content but mix the presence of different subjects. To avoid semantic overlapping between ID and OOD these sets have been removed from this study. Table 4.3 summarizes the custom binary CDDB in the faces (content) scenario. The **group scenario** uses as key of separation, the

Deepfake sources	# samples train	# samples test
In-Distribution (ID)		
Deepfake	3,248	1,082
Glow	7,200	4,800
StarGAN	7,200	4,800
WhichFaceReal	1,200	400
WildDeepFake	6,208	2,063
	25,056	13,145
Out-Of-Distribution (OOD)		
CRN	7,658	2,552
CycleGAN	1,572	524
IMLE	7,656	2,552
StyleGAN	7,188	2,394
	24,074	8,022

Table 4.3. CDDB Content scenario.

DF methods groups that we have seen in the previous section. Indeed, we have methods belonging to GAN, non-GAN, and unknown models. Our choice to define another this study case, is to use GAN models as ID data and the rest as OOD data. As before we use a table to wrap up this case. The last scenario defined, called

Deepfake sources	# samples train	# samples test
In-Distribution (ID)		
BigGAN	2400	800
CycleGAN	1,572	524
GauGAN	6,000	2000
StarGAN	7,200	4,800
StyleGAN	7,188	2,394
	24,360	10,518
Out-Of-Distribution (OOD)		
CRN	7,658	2,552
Deepfake	3,248	1,082
Glow	7,200	4,800
IMLE	7,656	2,552
SAN	260	90
WildDeepFake	6,208	2,063
WhichFaceReal	1,200	400
	33,430	13,539

Table 4.4. CDDB Group scenario.

mix scenario, separates samples based on techniques, assigning randomly to each Deepfake method either the ID or the OOD set. The assignment even if random, is capable of balancing the two partitions to avoid insufficient data for the Deepfake classifier, in the specific for the ID data we have: BigGAN, GauGAN, StarGAN,

Deepfake dataset, Glow, CRN, and WildDeepFake. In table 4.5 the other specifics. The same type of work is extended also for the multi-class classification task. In this

Deepfake sources	# samples train	# samples test
In-Distribution (ID)		
BigGAN	2400	800
CRN	7,658	2,552
Deepfake	3,248	1,082
GauGAN	6,000	2000
StarGAN	7,200	4,800
Glow	7,200	4,800
WildDeepFake	6,208	2,063
	39,914	18,097
Out-Of-Distribution (OOD)		
CycleGAN	1,572	524
IMLE	7,656	2,552
SAN	260	90
StyleGAN	7,188	2,394
WhichFaceReal	1,200	400
	17,876	5,960

Table 4.5. CDDB Mix scenario

case, each deepfake technique represents a label, while real images can be organized following different implemented criteria:

- Single, a unique label that plays for each real sample, like performed for the binary case (Still we have multi labels for fake examples).
- Categories, different categories, highlighting the content of real images. The content is the one discussed in the first scenario, like car, cat, dog, etc.
- Models, the grouping of real samples is based on the source of the real and fake images, so real ones like fake samples are divided for each model or technique.

After this process, three possible custom versions of the CDDB for the binary classification are available, and nine for multi-class. However, more possible versions can be customized, selecting different types of ID content, ID group/s, and ID techniques (mix scenario) in the configuration file. For example, another possible setup would be the choice of Zembra and Cat content as ID and the rest as OOD. Following this principle, a high number of study cases are available with this custom implementation of the CDDB dataset, considering the ID-OOD scenarios, ID data configuration, and whether pursuing the multi-class task, the real label grouping strategy as well.

Given the lack of a set for the validation, useful in the training phase of the model, has been defined two different policy to extract this set. In both cases, the validation set represents the 10% of the whole dataset in analysis. It's possible to

extract this validation data only from the test set, reducing it to 20%, or from both train and set, which respectively become the 65% and 25% of the data.
For the whole treatment, the selection went to the first option.

ID and OOD partitions are then used to build up another unsupervised dataset, in which we automatically assign the ID label (0) and the OOD positive label (1) to the samples. Before the assignment, the dataset (Binary or multi, Partial or Full) is shuffled to avoid problems with the position correlation.

Finally, we conclude this section by talking about data transformation and augmentation. Samples from CDDB don't respect a fixed resolution, and vary from squared 256p and 512×256, to high resolutions like 1024×1024. In addition, the file extension is different over the whole set including both .PNG and .JPEG images. Therefore images are resized to a fixed squared resolution, in this treatment 224p or 112p, using bilinear interpolation for a trade-off between speed and quality. Images are then converted to tensors, passing to pixel 8-bit int values in the range 0-255, to 32-bit float values in the range 0-1. This pro-processing pipeline includes also the possibility to use a float range from -1 to 1, like in common computer vision works. Whether request data fetching for training sessions, this module also provides base augmentation techniques such as horizontal and vertical image flip (50% and 10% of probability) and the RandAugment [18] technique with a single operation, magnitude 7 and 51 bins.

4.3 Training setup

This section describes the training setup applied for each developed task. We train all models on the Nvidia 1060 with 6GB of VRAM. The following important parameters need to be adjusted to get the best results on both deepfake and OOD detection tasks:

- batch size: number of training examples utilized in one step before updating, used to parallelize the calculation of the network. A neural network is trained by dividing the dataset into smaller batches, and after processing each batch, the model adjusts its weights. The choice of this hyperparameter introduces an important trade-off between overfitting, training time, and convergence. Small batches could lead to faster convergence and reduced training time with the risk of overfitting, while large batch size improves the training generalization but slows down the convergence of the training process requiring higher memory cost.

- Number of epochs: number of times the learning algorithm will work through the entire training dataset. The model's parameters are modified through each epoch, which comprises many iterations or steps, using the gradients calculated from a batch of training. A smaller number of epochs might lead to underfitting, which would make the model unable to effectively extract and interpret the input-output relationship. Alternatively, a larger number of

epochs might lead to overfitting, making the model over-trained on excessive epochs employing the same samples (w/o considering augmentation).

- data augmentation: a machine learning method that applies several modifications to the available data to artificially expand the size of a training dataset. Exposing a model to a wider variety of data variations, data augmentation aims to increase the model's robustness and generalization. Common transformations include altering brightness or contrast, flipping, shifting, shearing, and rotating,

- learning rate or η: indicates the size of the step taken during the descent of a gradient or training of the model.
  The Gradient Descent Update Rule can be described by the following formulae:

$$\theta' = \theta - \eta \cdot \nabla \mathcal{L}(\theta) \tag{4.1}$$

  With θ' the updated model's parameters, θ model's parameters before the update and $\nabla \mathcal{L}(\theta)$ the gradient of the loss function $\mathcal{L}$ respect θ.
  It plays a key role in controlling the convergence and optimization performance of a learning algorithm, increasing learning speed based on current batch estimations. Thus, The assignment of its value is crucial because variations on it can have a big effect on training.

- learning rate scheduler: an algorithm to adjust the learning rate during the training process dynamically. The learning rate scheduler helps fine-tune η over the course of training. When working with complicated optimization scenarios or trying to find the best possible balance between stability and rapid convergence, learning rate schedulers are very helpful.

- Optimizer: the optimization algorithm applied during training that adjusts the learning rate and weights of the neural network to reduce loss. Since it affects training speed and convergence, it's very important. Examples are Stochastic Gradient Descent (SGD), which exploits the Gradient Descent Update Rule 4.1, and Adaptive Moment Estimation (ADAM), which extends the idea of SGD by incorporating both momentum and adaptive learning rates. ADAM's Momentum term acts as a moving average of past gradients, helping to smooth the variations in the gradient updates.

- Weight decay: also referred to as L2 regularization is a regularization technique commonly used in machine learning and deep learning to prevent overfitting. Weight decay penalizes large weights during the training process, encouraging the model to use smaller weights for its parameters.

- Loss function: or cost/objective function, measures the performances of the models between the predicted outcomes and the real values from the ground truth. The goal during training is to minimize this loss function. The value of the loss function is reduced during back-propagation step in order to improve the neural network. This function depends on the typology of the task handled, is possible to use well-known functions like Cross-Entropy loss for classification

or custom loss functions.
In chapter 3 we have introduced custom loss functions, that combine well-known target functions. In the specific:

  - α, weights the importance of classification objective (BCE).
  - β, regulates the contribution of the reconstruction cost function (MSE or MAE).
  - λ, quantifies the penalization of the confidence estimation.
  - γ, used to dynamically update λ under [23] guidelines.

- Early Stopping: regularization techniques used to prevent overfitting and improve the generalization performance of the model on unseen data. It involves monitoring the model's performance on the validation set during training and stopping the training process once the model stops its improvement. The early stopping performance metrics can be different, for example, the loss or the accuracy of the model. The patience parameter is used to represent the number of consecutive epochs during which the monitored metric is allowed to not improve.

Concepts related to models' structure and forwarding, like layers, activation functions, dropout percentages, etc., are been already described in the chapter 3.
What follows are the training setups for models: ResNet50, ResNet50EDS, U-Net Scorer, DeiT tiny, AE or VAE, and all the Abnormality modules (basic, v1, v2, v3). All the models use L2 regularization with weight decay fixed to 1e-3, and Adam Optimizer with 0.9 momentum value.
With basic augmentation is intended image flipping (horizontal and vertical with respectively 0.5 and 0.1 probabilities) and RandAugment [18].

ResNet50 involves basic data augmentation. The model is trained using a batch size of 32 elements, with $\eta = 1e-3$ for 40 epochs, with early stopping based on accuracy and patience equal to 5. The learning rate exploited is the One Cycle Learning Rate with a starting percentage of 30%.
ResNet50EDS introduces the combined usage of base and cutmix augmentation[124]. The learning rate is decreased to 1e-4 opting for the Reduce learning rate on Plateau scheduler, with patience equal to 5, no cooldown, and a minimum learning rate of 1e-5. The scheduler is updated in max mode since it uses accuracy metric from validation.
U-Net Scorer setup modifies the previous one with a variable number of epochs (depending on the experiment) equivalent to 75 or 100, with also the early stopping activated from half the epochs using a patience value varying from 7 to 10. The learning rate is set to 1e-3 and the minimum scheduler learning rate is 1e-4. The custom loss function is characterized by $\alpha = 0.9$ and $\beta = 0.1$. The same combination of augmentation techniques and batch size as before is employed.
U-Net Scorer + Confidence includes in the target function also the following hyper-parameters $\lambda = 0.1$ and $\gamma = 0.3$.
The tiny DeiT classifier utilizes the same training setup as U-Net Scorer, changing the augmentation to the base version and a batch size of 64 samples.

Autoencoder models are trained with basic augmentation, 64 batch elements, 25 number of epochs, One Cycle Learning Rate scheduler with a starting percentage of 30%, and a learning rate equal to 1e-4. The loss function is selected depending on whether a classic AutoEncoder or Variational AutoEncoder is involved. In the first case, MAE is chosen, otherwise MAE + KD Divergence.
All the abnormality modules are characterized by training without augmentation, which is partially performed during the synthesis of OOD samples. One more time, the One Cycle Learning Rate scheduler is utilized for training, with a variable number of epochs equal to 20 or 50, and $\eta = 1e-3$. The loss function is the weighted BCE to solve the problem of unbalanced support in certain data configurations.

4.4 Metrics

The metrics used for the evaluation of our proposed method, follow the evaluation procedure used for state-of-the-art works. Basically, Deepfake detection and Out-Of-Distribution detection are two binary classification tasks, thus they share different metrics. Most metrics have value estimated as a number between 0 and 1, avoiding the expression in percentage. The definition of these metrics and a technical explanation are provided below.
Metrics used in the deepfake detection are Accuracy, Precision, Recall, F1-Score, AUROC, AUPR, Jaccard Score, and Confusion matrices. Instead for what concert the OOD detection we have: AUROC, AUPR, and FPR95. This set of metrics is exhaustive to give a representation of the OOD detection model, as suggested in the baseline [36] and following work like [23].
Different classification metrics are described in these terms: True positive (TP), True Negative (TN), false positive (FP), and false negative (FN). In details, True Positive are all the samples labeled and correctly predicted as True, True Negative are all the examples labeled and correctly predicted as False, False Positive are all the elements labeled as False and predicted as True, and False Negative are all the element labeled as True and predicted as False.
In problems with classification, accuracy is a frequently used metric that assesses the model's overall correctness. It is the ratio of the correct predictions over the whole predicted samples. In formal terms, accuracy is the probability that the model's prediction for a certain sample will be accurate. It's computed using the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.2}$$

Hence, accuracy is the sum of correct prediction of the total number of samples in the test The accuracy of positive predictions is the main emphasis of the precision metric. It calculates the proportion of accurately predicted positive observations to all positive predictions. It's computed as follows:

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

Precision is representative of the confidence degree to which we may rely on a model's prediction of a positive result.
Recall measures the ability of a model to capture all the relevant instances of a

positive class. It is the ratio of correctly predicted positive observations to all actual positives. It can be estimated in this way:

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

It represents the ratio of the number of true positives over the number of samples that effectively belong to the True class. Recall helps to identify how well the model captures positive instances.
F1 Score is the harmonic mean of precision and recall. It provides a balance between precision and recall, and is expressed in this manner:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.5}$$

This metric is very useful in situations where there is an uneven class distribution, and the equal contribution between Precision and Recall provides an optimal balance between the two metrics.
The Jaccard Score is a metric that's used to assess how similar two sets are and it's sometimes referred to as the Intersection over Union, or IoU. The computation in the binary classification case is the following:

$$Jaccard\ Score = \frac{TP}{TP + FP + FN} \tag{4.6}$$

Jaccard Score ranges from 0 to 1, where 1 indicates a perfect overlap between predicted and true positives, moreover, it can be used when dealing with unbalanced datasets.
An illustration of the distribution of predictions within the class is provided by the Confusion matrix, which is a cross table. It provides a detailed breakdown of the model's predictions and actual class labels. This matrix has an equal number of rows and columns, and this value is defined by the number of classes in our task. Therefore in this binary case, the matrix is composed by two rows and columns. In figure 4.3 we can appreciate the structure and the meaning behind each cell.
By maintaining the same order between elements through rows and columns, it is

		Predicted	
		Negative **(N)** -	Positive **(P)** +
Actual	Negative -	True Negative **(TN)**	**False Positive (FP) Type I Error**
	Positive +	**False Negative (FN) Type II Error**	True Positive **(TP)**

Figure 4.3. Confusion matrix structure in the binary classification scenario.

possible to visualize the number of correct predictions in the main diagonal from left to right of the matrix. Whether the matrix is normalized, this is a percentage value. Other cells of the diagonal represent misclassified samples. Comprehending and analyzing the confusion matrix helps researchers in identifying the model's strengths and potential areas for growth, directing future improvements.
The next two metrics are the ones shared between OOD detection and deepfake detection.
The ROC curve (receiver operating characteristic curve) is a graphical representation that illustrates the discriminative ability of a binary classification model across different thresholds. It incorporates the true positive rate (TPR) against the false positive rate (FPR) for various threshold values. TPR and Recall are the same concept, representing sensitivity. TPR and FPR are computed in the following way:

$$TPR = \frac{TP}{TP + FN} \tag{4.7}$$

$$FPR = \frac{FP}{FP + TN} \tag{4.8}$$

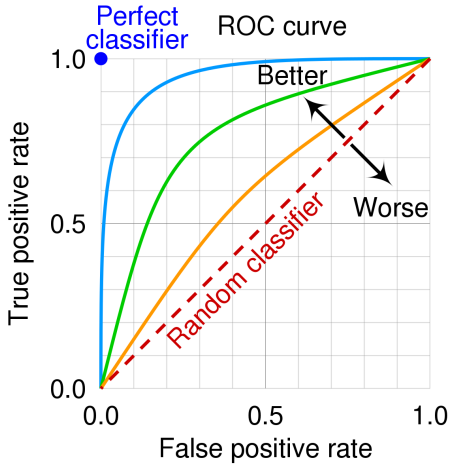

Figure 4.4. ROC curve representation.

An example of AUROC curve is in figure 4.4. The curve shows the trade-off between sensitivity and specificity, and a higher curve indicates a better model. In particular, a triangular shape plot represents a random classifier, like a coin flip, while a squared shape indicates the best possible classifier.
The threshold-independent nature of this curve is useful for understanding the overall discriminatory power of the model.
It's also possible to compute a scalar metric based on this curve to estimate the overall performances of the binary classifier, called AUROC or Area Under the ROC curve. AUROC ranges from 0 to 1, where 0.5 represents a random classifier, and 1 represents a perfect classifier. Higher AUCROC values indicate better discrimination between positive and negative instances.

Similarly, we can define the Precision-Recall (PR) curve and AUPR (Area under PR curve) metric. Like the ROC curve, the PR curve is threshold-independent and graphically represents the trade-off between precision and recall for different threshold values in a binary classification model. We can appreciate an example in figure 4.5. Still, the ideal model is represented by a squared shape plot, while the

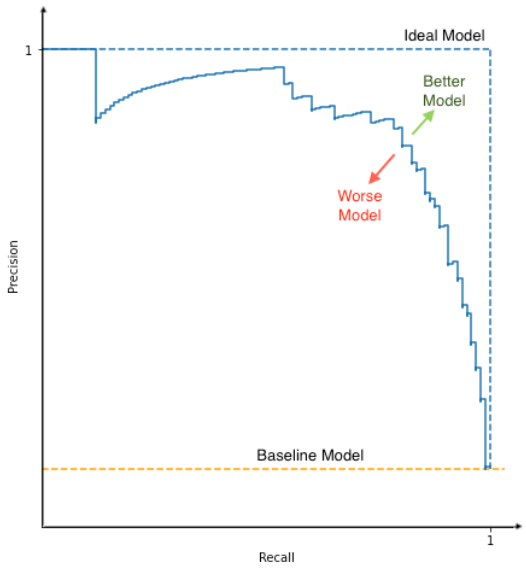

Figure 4.5. PR curve representation.

model with no skill is represented as a line on the horizontal axis of this cartesian plot. PR curves are particularly useful when dealing with imbalanced datasets. AUPR quantifies the area under the PR curve. Similar to AUROC, it provides a single scalar value to measure the overall performance of a binary classification model. AUPR ranges from 0 to 1, where 0 represents a model with no predictive power, and 1 represents a perfect classifier. We conclude this section with a metric commonly used in OOD detection tasks. FPR95 Measures the false positive rate (FPR) when the true positive rate (TPR) is equal to 95%. It's particularly relevant in applications where maintaining a high sensitivity (low false negatives) is important, and there's a specific requirement to limit the false positive rate. the FPR95 metric corresponds to a specific operating point on the Receiver Operating Characteristic (ROC) curve.

Chapter 5

Results

In this chapter, we explore the outcomes coming from our extensive investigation of Out-Of-Distribution detection for what concerns our Deepfake task. Results are not only divided by approach but also refer to different deepfake detection scenarios, re-implementing a few OOD detection baselines for a comparative analysis of methods on this new CDDB dataset usage. Therefore we consider metrics from both the detection tasks since OOD results are dependent also on the ID performances.
Finally, a reimplemented benchmark from [127] is included in the study, to take into account the whole set of methods present in the literature for a complete comparison of the methodology developed. In all the following metrics we always consider the positive label as the fake images in deepfake detection, and the OOD samples in the OOO detection task.

5.1 Deepfake Detection

Following the development process exposed in the methodology chapter 3, as first are exposed the evaluation performances recorded for the ResNet50 model. Under the pre-processing step performed on CDDB dataset, three scenarios are defined: Content, Group, and Mix. Besides this, we provide the ResNet50 evaluation metric over the whole dataset, to give an idea about the classification capabilities of a pure classification model trained only to discriminate between real and fake samples.
ResNet50 is trained for 20 epochs, and requires 2,970 MB of VRAM with a batch size of 32 elements, its loss trend can be appreciated in figure 5.1. Whether is not differently stated, we always assume this batch size for all the memory data.
Classification plots and confusion matrix are shown in 5.2. Areas under ROC and PR curves are respectively equal to 0.993 and 0.994. This model reaches a 0.972 of accuracy, with precision equal to 0.976, recall to 0.969, and a Jaccard score of 0.947. This leads to an F1-score of 0.972.

ResNet50EDS confirms similar results also including the reconstruction target, with a slight degradation of its classification abilities. The F1-score goes from 0.972. to 0.968. Although this model has a huge impact on the computation resources required, with about 4,530 MB of memory necessary to forward the batch.

U-Net Scorer results are available for all three scenarios. Before showing these

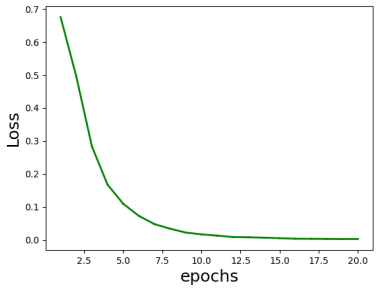

Figure 5.1. ResNet50 Training loss over epochs, full CDDB dataset.

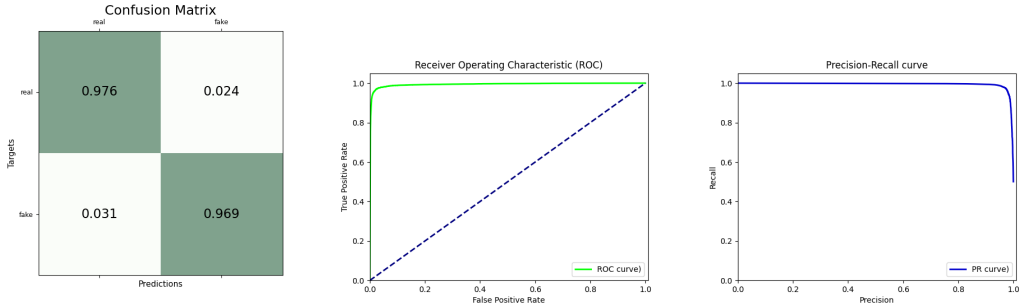

Figure 5.2. Confusion matrix, ROC and PR curves from ResNet50 evaluation.

evaluations and their specifics, we propose two interesting comparisons. First are introduced the U-Net4 metrics trained with 112p images. These results are then compared with the ones coming from the U-Net4 Scorer + Confidence model, showing how the learning of confidence estimation can impact also the classification performances. In addition, by training U-Net5 Scorer with 224p images we want to demonstrate how the downsampling of input information could be critical, removing key information useful for correct inferences. Is important to notice how the inclusion of the confidence branch has an insignificant impact from the computational point of view. Similarly, U-Net4 Scorer with 112p input images and U-Net5 Scorer with 224p images, have the same memory impact of around 2,760 MB. Things change whether we apply 224×224 images with U-Net4 Scorer, incrementing the latent encoding size processed from the classification head, leading to a memory request equal to 3,602 MB. Motivated by this 30% memory increase, it's suggested the utilization of U-Net5 Scorer with the standard 224 pixels of spatial dimensions. Additionally, we have recorded a training speed for U-Net5 Scorer model equal to 6.08 epochs per hour.

Figure 5.3 displays the learning plots relative to the three models under the same Content scenario, with real and deepfake faces as dataset samples. Confusion matrices in 5.4 and table 5.1 give a complete comparison of the metrics.

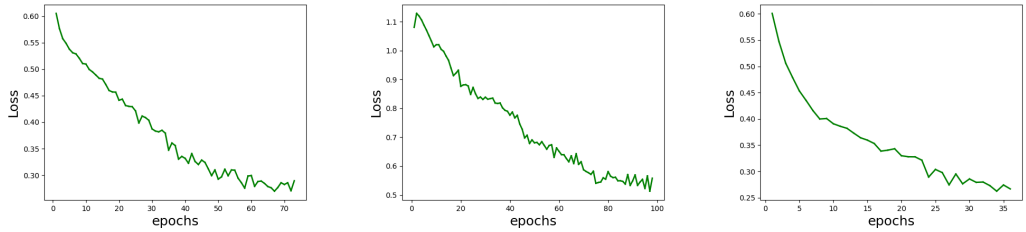

Figure 5.3. Loss plots of U-Net Scorer models, Content scenario.

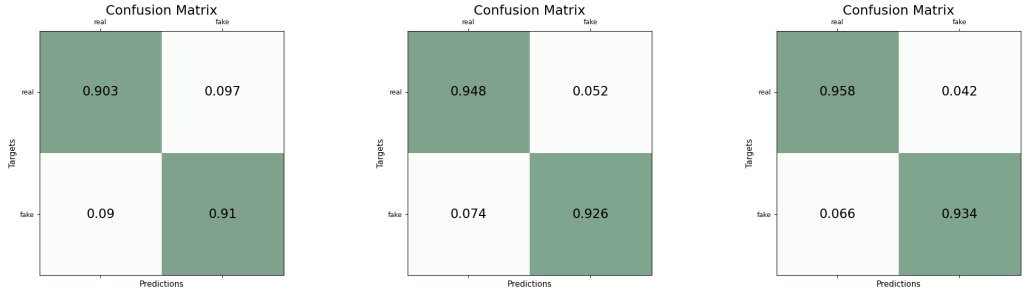

Figure 5.4. Confusion matrices of U-Net Scorer models, Content scenario.

From left to right: U-Net4 Scorer, U-Net4 Scorer + Confidence , U-Net5 Scorer.

Metric	U-Net4 Scorer	U-Net4 Scorer + C.	U-Net5 Scorer
Accuracy	0.906	0.937	0.946
Precision	0.904	0.947	0.957
Recall	0.910	0.926	0.934
F1-Score	0.907	0.937	**0.945**
AUROC	0.968	0.988	0.991
AUPR	0.968	0.988	0.991
Jaccard Score	0.830	0.881	0.897

Table 5.1. Performance Metrics for U-Net Models, Content scenario.

Group and Mix scenarios have proven to be more complex for detection. The group scenario contains classification samples with real and GAN-based generated deepfake images. While the Mix scenario randomly takes a subsection of the available deepfake techniques/sources for the In-Distribution data, specifically: BigGAN, GauGAN, StarGAN, deepfake dataset, Glow, CRN, and WildDeepFake. Below are shown the evaluation results involving U-Net5 Scorer model, in figure 5.5 the confusion matrices and in table 5.2 the metrics summarized.
The model was trained for 71 epochs in the Group scenario task, while for 51 in the Mix. The number of epochs is different due to early stopping.

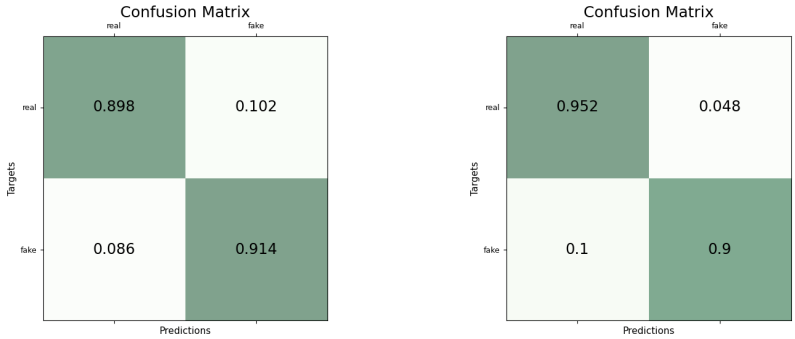

Figure 5.5. Confusion matrices of U-Net5 Scorer model over Group and Mix scenarios.

From left to right: Group and Mix scenarios.

Metric	Group scenario	Mix Scenario
Accuracy	0.906	0.926
Precision	0.902	0.950
Recall	0.914	0.900
F1-Score	0.908	0.924
AUROC	0.973	0.988
AUPR	0.973	0.988
Jaccard Score	0.831	0.860

Table 5.2. Performance Metrics for U-Net5 Model, over Group and Mix scenarios.

We finally conclude this section on the results from pure deepfake detection as our In-Distribution classification problem, exposing the performances of the ViT-based solution which involves the employment of the tiny DeiT model for the detection. The evaluation is proposed for the Content scenario. This model has a reduced memory impact if compared to the previous model, with only 1,118 MB of video memory necessary. This leads to a reduction of almost %57 of memory usage with respect to the 2,760 MB from U-Net5 Scorer. Instead, the training speed remains similarly stable at 6.17 epochs per hour. Figure 5.6 shows the loss trend over epochs in the training phase, while in 5.7 are exposed its confusion matrix, ROC and PR curves. This model reaches an accuracy and precision of 0.96, with a recall of about 0.956. The resulting F1-score corresponds to 0.96. Moreover, the Jaccard score is 0.92, with AUROC and AUPR both 0.993.

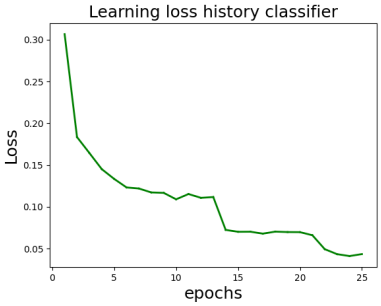

Figure 5.6. Tiny DeiT training loss over epochs.

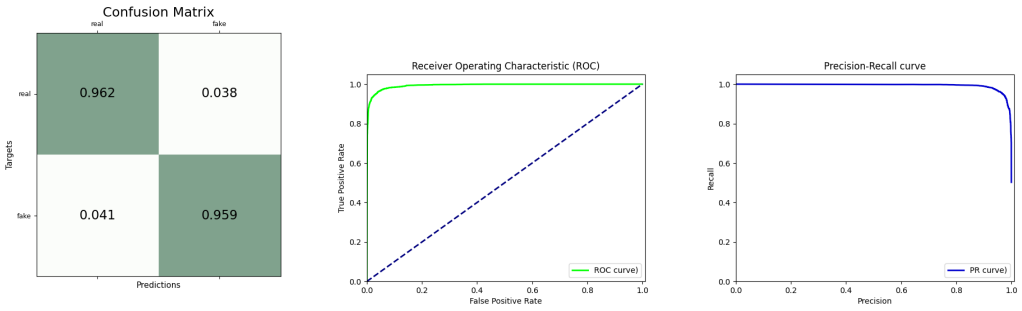

Figure 5.7. Confusion matrix, ROC and PR curves from tiny DeiT evaluation.

5.2 OOD detection

Based on the results obtained in the deepfake detection exploiting the different approaches and models, it's possible to discuss the Out-Of-Distribution detection performances of the proposed work. We mainly focus on the results obtained with our deepfake detection custom dataset, involving different scenarios and training methodologies of the abnormality modules. Three different well-known OOD techniques are been implemented and tested directly on our novel data to have a baseline for comparison.
Finally, is conducted an evaluation work on the CIFAR10 OOD benchmark from [127], comparing the proposed method with respect to a wide range of techniques in the field.

5.2.1 Baselines

The baselines proposed are the MSP [36], the ODIN [64], and the confidence branch [23]. For the confidence method results are available only for the Content scenario. This is because differently from MSP and ODIN which are post-hoc methods, it requires an ad hoc training instance, which takes a long time to be accomplished. Therefore, we prefer to invest time in other types of training and evaluations.
The confidence branch solution employ U-Net4 Scorer + Confidence with 112p images as ID classifier, while both MSP and ODIN use U-Net5 Scorer upstream with 224p images. Follows three tables, 5.3, 5.4, 5.5 for each scenario.

Metric	MSP	ODIN	Conf. branch
AUROC	77.85	**82.67**	78.38
AUPR	71.61	**80.03**	68.17
FPR95	0.61	0.56	0.61

Table 5.3. Baselines performances on Content scenario.

Metric	MSP	ODIN
AUROC	62.06	60.3
AUPR	58.9	54.31
FPR95	0.84	0.83

Table 5.4. Baselines performances on Group scenario.

Metric	MSP	ODIN
AUROC	69.74	62.58
AUPR	61.6	55.05
FPR95	0.74	0.823

Table 5.5. Baselines performances on Mix scenario.

It's clear how ODIN outperforms other methods in the Content scenario, but at same time in the other cases the MSP baseline results in better performances. Moreover, from these poor evaluation values, we can understand the complexity of the proposed task and how is challenging to achieve good results in the discrimination of ID and OOD data here.

5.2.2 Custom Abnormality module

Starting from the three custom Abnormality modules with encoders defined, the three different CDDB scenarios, and the various training methodologies proposed, a large amount of results and metrics were recorded. For this reason, only a subset of all the evaluation data in possession are exposed, including the most interesting results. In fact, the module can be trained with synthesized OOD data, real OOD samples, or mixing them. The following acronyms are used to indicate this specific: **S** for synthesized, **R** for real, and **M** for mixed data.
Given the clear problems and limitations, the Basic version of the Abnormality module is not included in this treatment. Besides this, we use its memory impact as a reference for a comparison with the other versions. The basic Abnormality module requires 1084 MB of free Video RAM, which is halved from both V1 and V3 versions, respectively around 514 and 564 MB. V2 module is the lightest with only 106 MB of memory necessary for forwarding 32 batch elements. Training speed is almost equal for all three versions and is about 15 epochs per hour.

The first comparison we propose is between the three Abnormality Modules, trained and tested on the Content scenario with U-Net4 Scorer as ID classifier. The classifier is equal to 0.906. The table below represents the different training methodologies.

Metric	V1S	V2S	V3S	V1R	V2R	V3R	V1M	V2M	V3M
AUROC	77.79	**91.6**	64.13	99.97	100	99.89	99.93	99.95	97.45
AUPR	80.06	**93.55**	66.65	99.96	100	99.9	99.91	99.88	95.98
FPR95	0.83	**0.49**	0.94	2e-4	0	2e-4	8e-4	8e-4	0.04

Table 5.6. Custom U-Net Abnormality modules performances on Content scenario.

V1, V2, and V3 modules trained under the following training methodologies: only synthesized OOD samples (S), only real OOD samples (R), and with mixed synthesized and real OOD samples (M).

The most interesting outcome is the one from V2S since produces good results using only altered ID data treated as OOD samples in the training phase, providing an interesting unsupervised approach. Notable are the perfect classification metrics from V2R, which validates the potentiality of the V2 module above the others. Loss trend during training and the ROC curve of V2S are displayed in figure 5.8.

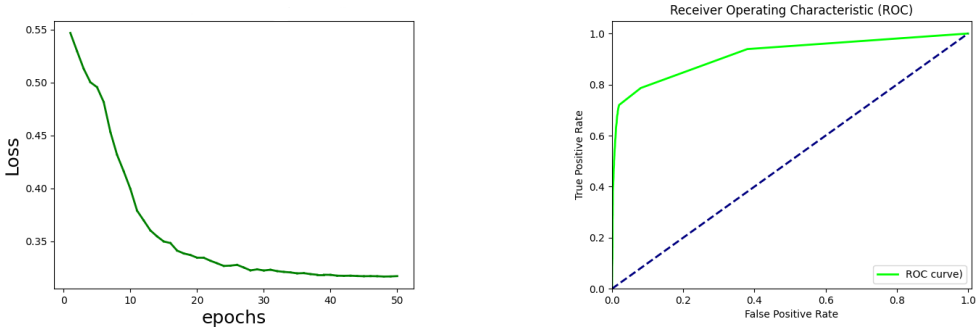

Figure 5.8. U-Net Abnormality module Encoder V2S: training loss plot and ROC curve.

Moving toward a more complex task, other results on the Group and Mix scenario are proposed in table 5.7. This time the ID classifier is the U-Net5 Scorer which produces an accuracy of 0.906 for the Group and 0.926 for the Mix scenario. The results are exposed only for the module that better performs, which as demonstrated is V2.

Metric	V2S Group	V2M Group	V2S Mix	V2M Mix
AUROC	63.35	92.72	60.14	80.0
AUPR	64.21	95.1	65.48	86.19
FPR95	0.90	0.59	0.94	0.87

Table 5.7. Custom U-Net Abnormality modules performances on Group and scenario.

It is clear that these scenarios are truly complex, despite this, good results are obtained by using real outliers.

Afterward the demonstration of results for U-Net approach, we move on evaluating how the ViT-based technique performs. This approach requires an additional model for the reconstruction of the attention maps. This can be performed from both AutoEncoder and Variation AutoEncoder models. The first requires 444 MB of memory while the second almost 900 MB. Training speed in both cases is pretty similar, AE is trained elaborating 22.03 epochs per hour, while 22.03 epochs per hour for VAE. Training both for 25 epochs on the same data coming from the Content scenario, we can give an evaluation in terms of MAE and MSE which are organized in the table 5.8.

Metric	AE	VAE
MAE	9.43e-6	**1.63e-7**
MSE	2e-4	2e-4

Table 5.8. AE and MAE metrics.

In a similar manner to what has been shown for the U-Net approach in the Content scenario, it's proposed the same evaluation environment with this different inference pipeline for Abnormality modules V2 and V3. All data are inserted in the table 5.9.

Metric	V2S	V3S	V2R	V3R	V2M	V3M
AUROC	81.96	**93.77**	100	99.98	99.97	99.94
AUPR	85.23	**94.55**	100	99.98	99.98	99.92
FPR95	0.81	**0.41**	1e-4	2e-4	3e-4	3e-4

Table 5.9. Custom ViT Abnormality modules V1 and V2 performances on Content scenario.

In this case, V3 performs surprisingly better than V2 module beating the best result of the U-Net approach on the training with only synthetic data. This section concludes with the training loss history and ROC curve about this model 5.9, which represents the best result achieved in the ViT-based approach, but also the best compared to the U-Net based approach. Considering instead real OOD samples involved in the supervised training, results from V2 and V3 are practically at the same level.

5.2.3 OOD Benchmark

The following tables present the results of our technique applied to CIFAR10 benchmark from [127]. Each table is divided into three blocks: the first represents the Post-hoc inference methods which don't require any training, the second block is related to trained methods without the necessity of OOD samples, while the third includes techniques that require OOD data as outliers.
Our methods named "ABN Module V2S" and "ABN Module V2M" are written in

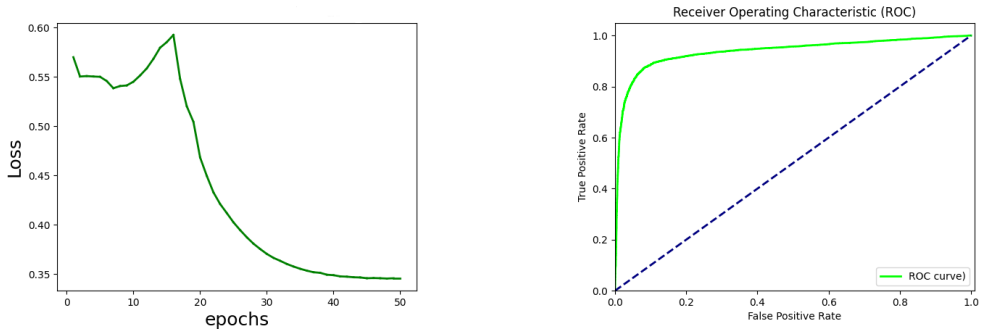

Figure 5.9. ViT Abnormality module Encoder V3S: training loss plot and ROC curve.

bold as well as the best performance in the same table.
The approach involved is the ViT-based, reaching an ID accuracy of 97.5%. As before, the name **V2S** represents the abnormality module V2 trained using only synthesized OOD data, while **V2M** refers to the same module with a mix of synthetic and real OOD samples. Real OOD are data sampled from near OOD datasets as outliers. Thus, V2S belongs to the second block, while V2R is in the third one.
Moreover, the evaluations are divided between Near and Far OOD, an empiric separation proposed by the benchmark itself. In details for Far-OOD are assumed: MNIST [22], SVHN [3], Texture [17] datasets. While for Near-OOD are included the CIFAR100 [1] and TinyImageNet [101].
Respectively, in table 5.10 and 5.11, we have AUROC metric for Far and Near OOD. Tables 5.12 and 5.13 are the Far and Near OOD results in terms of AUPR. Furthermore, for the other techniques, the tables present whether is available the average number and the corresponding standard deviation of the metrics obtained from 3 training runs.

Technique	MNIST	SVHN	Textures	ID ACC
OpenMax	90.50 ± 0.44	89.77 ± 0.45	89.58 ± 0.60	95.06 ± 0.30
MSP	92.63 ± 1.57	91.46 ± 0.40	89.89 ± 0.71	95.06 ± 0.30
TempScale	93.11 ± 1.77	91.66 ± 0.52	90.01 ± 0.74	95.06 ± 0.30
ODIN	95.24 ± 1.96	84.58 ± 0.77	86.94 ± 2.26	95.06 ± 0.30
MDS	90.10 ± 2.41	91.18 ± 0.47	92.69 ± 1.06	95.06 ± 0.30
MDSEns	99.17 ± 0.41	66.56 ± 0.58	77.40 ± 0.28	95.06 ± 0.30
RMDS	93.22 ± 0.80	91.84 ± 0.26	92.23 ± 0.23	95.06 ± 0.30
Gram	72.64 ± 2.34	91.52 ± 4.45	62.34 ± 8.27	95.06 ± 0.30
EBO	94.32 ± 2.53	91.79 ± 0.98	89.47 ± 0.70	95.06 ± 0.30
OpenGAN	56.14 ± 24.08	52.81 ± 27.60	56.14 ± 18.26	95.06 ± 0.30
GradNorm	63.72 ± 7.37	53.91 ± 6.36	52.07 ± 4.09	95.06 ± 0.30
ReAct	92.81 ± 3.03	89.12 ± 3.19	89.38 ± 1.49	95.06 ± 0.30
MLS	94.15 ± 2.48	91.69 ± 0.94	89.41 ± 0.71	95.06 ± 0.30
KLM	85.00 ± 2.04	84.99 ± 1.18	82.35 ± 0.33	95.06 ± 0.30
VIM	94.76 ± 0.38	94.50 ± 0.48	95.15 ± 0.34	95.06 ± 0.30
KNN	94.26 ± 0.38	92.67 ± 0.30	93.16 ± 0.24	95.06 ± 0.30
DICE	90.37 ± 5.97	90.02 ± 1.77	81.86 ± 2.35	95.06 ± 0.30
RankFeat	75.87 ± 5.22	68.15 ± 7.44	73.46 ± 6.49	95.06 ± 0.30
ASH	83.16 ± 4.66	73.46 ± 6.41	77.45 ± 2.39	95.06 ± 0.30
SHE	90.43 ± 4.76	86.38 ± 1.32	81.57 ± 1.21	95.06 ± 0.30
GEN	93.83 ± 2.14	91.97 ± 0.66	90.14 ± 0.76	95.06 ± 0.30
ConfBranch	94.49 ± 0.77	95.42 ± 0.35	91.10 ± 0.41	94.88 ± 0.05
RotPred	97.52 ± 0.14	98.89 ± 0.09	97.30 ± 0.06	95.35 ± 0.52
RotPred + PixMix	98.00 ± 0.34	99.33 ± 0.05	99.21 ± 0.02	95.91 ± 0.25
G-ODIN	98.95 ± 0.53	97.76 ± 0.14	95.02 ± 1.10	94.70 ± 0.25
CSI	92.55 ± 1.15	95.18 ± 0.45	90.71 ± 0.44	91.16 ± 0.14
ARPL	92.62 ± 0.88	87.69 ± 0.97	88.57 ± 0.43	93.66 ± 0.11
MOS	74.81 ± 10.05	73.66 ± 9.14	70.35 ± 3.11	94.83 ± 0.37
VOS	91.56 ± 2.37	92.18 ± 1.65	89.68 ± 1.32	94.31 ± 0.64
LogitNorm	99.14 ± 0.45	98.25 ± 0.41	94.77 ± 0.43	94.30 ± 0.25
LogitNorm + PixMix	98.53 ± 0.11	98.36 ± 0.27	98.74 ± 0.05	94.60 ± 0.25
CIDER	93.30 ± 1.08	98.06 ± 0.07	93.71 ± 0.39	/
NPOS	92.64 ± 1.59	98.88 ± 0.09	94.44 ± 0.90	/
ABN Module V2S	93.15	95.42	94.88	97.5
OE	90.22 ± 1.31	**99.60 ± 0.14**	97.58 ± 0.27	94.63 ± 0.26
MCD	84.22 ± 2.10	93.76 ± 2.30	93.35 ± 1.30	94.95 ± 0.04
UDG	95.81 ± 1.52	94.55 ± 2.27	93.92 ± 0.44	92.36 ± 0.84
MixOE	91.66 ± 2.21	93.82 ± 1.27	91.84 ± 0.51	94.55 ± 0.32
ABN Module V2M	**99.24**	98.06	**99.43**	97.5

Table 5.10. AUROC metric far-OOD detection on CIFAR10 Benchmark.

Technique	CIFAR-100	TIN	ID ACC
OpenMax	86.91 ± 0.31	88.32 ± 0.28	95.06 ± 0.30
MSP	87.19 ± 0.33	88.87 ± 0.19	95.06 ± 0.30
TempScale	87.17 ± 0.40	89.00 ± 0.23	95.06 ± 0.30
ODIN	82.18 ± 1.87	83.55 ± 1.84	95.06 ± 0.30
MDS	83.59 ± 2.27	84.81 ± 2.53	95.06 ± 0.30
MDSEns	61.29 ± 0.23	59.57 ± 0.53	95.06 ± 0.30
RMDS	88.83 ± 0.35	90.76 ± 0.27	95.06 ± 0.30
Gram	58.33 ± 4.49	58.98 ± 5.19	95.06 ± 0.30
EBO	86.36 ± 0.58	88.80 ± 0.36	95.06 ± 0.30
OpenGAN	52.81 ± 7.69	54.62 ± 7.68	95.06 ± 0.30
GradNorm	54.43 ± 1.59	55.37 ± 0.41	95.06 ± 0.30
ReAct	85.93 ± 0.83	88.29 ± 0.44	95.06 ± 0.30
MLS	86.31 ± 0.59	88.72 ± 0.36	95.06 ± 0.30
KLM	77.89 ± 0.75	80.49 ± 0.85	95.06 ± 0.30
VIM	87.75 ± 0.28	89.62 ± 0.33	95.06 ± 0.30
KNN	89.73 ± 0.14	91.56 ± 0.26	95.06 ± 0.30
DICE	77.01 ± 0.88	79.67 ± 0.87	95.06 ± 0.30
RankFeat	77.98 ± 2.24	80.94 ± 2.80	95.06 ± 0.30
ASH	74.11 ± 1.55	76.44 ± 0.61	95.06 ± 0.30
SHE	80.31 ± 0.69	82.76 ± 0.43	95.06 ± 0.30
GEN	87.21 ± 0.36	89.20 ± 0.25	95.06 ± 0.30
ConfBranch	88.91 ± 0.25	90.77 ± 0.25	94.88 ± 0.05
RotPred	91.19 ± 0.32	94.17 ± 0.24	95.35 ± 0.52
RotPred + PixMix	**93.48 ± 0.13**	96.24 ± 0.14	95.91 ± 0.25
G-ODIN	88.14 ± 0.60	90.09 ± 0.54	94.70 ± 0.25
CSI	88.16 ± 0.16	90.87 ± 0.23	91.16 ± 0.14
ARPL	86.76 ± 0.16	88.12 ± 0.14	93.66 ± 0.11
MOS	70.57 ± 3.04	72.34 ± 3.16	94.83 ± 0.37
VOS	86.57 ± 0.57	88.84 ± 0.48	94.31 ± 0.64
LogitNorm	90.95 ± 0.22	93.70 ± 0.06	94.30 ± 0.25
LogitNorm + PixMix	92.91 ± 0.25	95.60 ± 0.15	94.60 ± 0.25
CIDER	89.47 ± 0.19	91.94 ± 0.19	/
NPOS	88.57 ± 0.36	90.99 ± 0.35	/
ABN Module V2S	69.07	84.02	97.5
OE	90.54 ± 0.53	99.11 ± 0.34	94.63 ± 0.26
MCD	89.88 ± 0.07	92.18 ± 0.18	94.95 ± 0.04
UDG	88.62 ± 0.32	91.20 ± 0.20	92.36 ± 0.84
MixOE	87.47 ± 0.97	90.00 ± 0.73	94.55 ± 0.32
ABN Module V2M	93.36	**99.66**	97.5

Table 5.11. AUROC metric near-OOD detection on CIFAR10 Benchmark.

Technique	MNIST	SVHN	Textures	ID ACC
OpenMax	96.63 ± 0.52	92.36 ± 0.32	77.09 ± 0.89	95.06 ± 0.30
MSP	98.63 ± 0.29	95.79 ± 0.22	82.47 ± 1.03	95.06 ± 0.30
TempScale	98.77 ± 0.32	96.05 ± 0.25	83.53 ± 1.00	95.06 ± 0.30
ODIN	99.29 ± 0.29	93.73 ± 0.37	84.08 ± 1.68	95.06 ± 0.30
MDS	97.61 ± 0.77	95.07 ± 0.23	88.29 ± 1.34	95.06 ± 0.30
MDSEns	99.56 ± 0.29	81.08 ± 0.45	71.73 ± 0.39	95.06 ± 0.30
RMDS	98.70 ± 0.18	95.46 ± 0.11	85.50 ± 0.43	95.06 ± 0.30
Gram	93.36 ± 1.21	96.62 ± 1.81	55.93 ± 10.76	95.06 ± 0.30
EBO	99.06 ± 0.45	96.50 ± 0.35	84.92 ± 0.75	95.06 ± 0.30
OpenGAN	88.99 ± 7.04	74.87 ± 14.62	47.57 ± 17.14	95.06 ± 0.30
GradNorm	93.42 ± 1.59	78.89 ± 4.47	48.05 ± 6.07	95.06 ± 0.30
ReAct	98.84 ± 0.53	95.47 ± 1.22	84.66 ± 1.67	95.06 ± 0.30
MLS	99.03 ± 0.43	96.43 ± 0.34	84.70 ± 0.79	95.06 ± 0.30
KLM	97.61 ± 0.39	93.50 ± 0.49	77.39 ± 0.82	95.06 ± 0.30
VIM	98.94 ± 0.15	97.35 ± 0.43	92.43 ± 0.65	95.06 ± 0.30
KNN	98.95 ± 0.07	96.25 ± 0.17	87.26 ± 0.50	95.06 ± 0.30
DICE	97.91 ± 1.59	95.14 ± 1.05	72.11 ± 5.11	95.06 ± 0.30
RankFeat	94.70 ± 1.26	80.33 ± 4.78	55.39 ± 7.84	95.06 ± 0.30
ASH	97.39 ± 0.74	89.06 ± 3.15	72.85 ± 3.75	95.06 ± 0.30
SHE	98.43 ± 0.80	94.46 ± 0.44	77.28 ± 1.40	95.06 ± 0.30
GEN	98.95 ± 0.39	96.37 ± 0.41	84.71 ± 0.89	95.06 ± 0.30
ConfBranch	98.71 ± 0.23	97.26 ± 0.54	81.78 ± 1.05	94.88 ± 0.05
RotPred	99.43 ± 0.09	99.33 ± 0.03	94.10 ± 0.14	95.35 ± 0.52
RotPred + PixMix	99.68 ± 0.05	99.57 ± 0.03	97.30 ± 0.12	95.91 ± 0.25
G-ODIN	99.83 ± 0.09	99.12 ± 0.06	93.41 ± 1.24	94.70 ± 0.25
CSI	98.49 ± 0.34	97.75 ± 0.32	82.99 ± 0.94	91.16 ± 0.14
ARPL	98.53 ± 0.20	93.27 ± 0.44	78.27 ± 0.83	93.66 ± 0.11
MOS	95.24 ± 2.14	87.20 ± 5.22	62.67 ± 3.94	94.83 ± 0.37
VOS	98.53 ± 0.42	96.37 ± 0.79	84.05 ± 2.13	94.31 ± 0.64
LogitNorm	**99.87 ± 0.07**	99.31 ± 0.18	91.68 ± 0.86	94.30 ± 0.25
LogitNorm + PixMix	99.77 ± 0.02	99.35 ± 0.12	98.02 ± 0.06	94.60 ± 0.25
CIDER	98.75 ± 0.20	99.19 ± 0.07	88.93 ± 0.34	/
NPOS	98.54 ± 0.41	99.60 ± 0.04	90.18 ± 2.04	/
ABN Module V2S	92.3	98.48	95.37	97.5
OE	97.20 ± 0.37	**99.80 ± 0.07**	96.11 ± 0.17	94.63 ± 0.26
MCD	97.01 ± 0.38	96.39 ± 1.48	87.12 ± 1.78	94.95 ± 0.04
UDG	99.26 ± 0.34	96.86 ± 1.48	87.68 ± 1.83	92.36 ± 0.84
MixOE	98.63 ± 0.38	97.06 ± 0.75	87.48 ± 0.76	94.55 ± 0.32
ABN Module V2M	97.52	99.02	**99.5**	97.5

Table 5.12. AUPR metric far-OOD detection on CIFAR10 Benchmark.

Technique	CIFAR-100	TIN	ID ACC
OpenMax	81.25 ± 0.24	79.96 ± 0.37	95.06 ± 0.30
MSP	85.39 ± 0.26	85.48 ± 0.47	95.06 ± 0.30
TempScale	85.95 ± 0.26	86.27 ± 0.44	95.06 ± 0.30
ODIN	83.39 ± 1.16	82.67 ± 1.16	95.06 ± 0.30
MDS	80.71 ± 2.82	79.05 ± 3.54	95.06 ± 0.30
MDSEns	62.34 ± 0.36	57.54 ± 0.28	95.06 ± 0.30
RMDS	87.25 ± 0.27	87.80 ± 0.39	95.06 ± 0.30
Gram	59.24 ± 4.62	55.89 ± 5.56	95.06 ± 0.30
EBO	86.54 ± 0.24	87.54 ± 0.38	95.06 ± 0.30
OpenGAN	54.24 ± 5.38	52.46 ± 5.06	95.06 ± 0.30
GradNorm	58.81 ± 2.18	57.10 ± 1.84	95.06 ± 0.30
ReAct	86.21 ± 0.36	87.10 ± 0.24	95.06 ± 0.30
MLS	86.41 ± 0.25	87.34 ± 0.41	95.06 ± 0.30
KLM	80.18 ± 0.39	80.57 ± 0.54	95.06 ± 0.30
VIM	86.37 ± 0.23	86.27 ± 0.55	95.06 ± 0.30
KNN	88.22 ± 0.24	88.77 ± 0.45	95.06 ± 0.30
DICE	74.98 ± 1.88	74.61 ± 2.79	95.06 ± 0.30
RankFeat	74.44 ± 2.75	74.48 ± 3.41	95.06 ± 0.30
ASH	77.19 ± 1.60	77.30 ± 1.01	95.06 ± 0.30
SHE	81.82 ± 0.37	82.27 ± 0.73	95.06 ± 0.30
GEN	86.62 ± 0.21	87.22 ± 0.12	95.06 ± 0.30
ConfBranch	85.22 ± 0.29	85.78 ± 0.39	94.88 ± 0.05
RotPred	89.20 ± 0.36	91.75 ± 0.36	95.35 ± 0.52
RotPred + PixMix	92.14 ± 0.19	94.54 ± 0.20	95.91 ± 0.25
G-ODIN	88.09 ± 0.49	88.41 ± 0.49	94.70 ± 0.25
CSI	85.45 ± 0.24	87.30 ± 0.30	91.16 ± 0.14
ARPL	83.25 ± 0.34	82.66 ± 0.33	93.66 ± 0.11
MOS	72.70 ± 2.93	72.13 ± 3.17	94.83 ± 0.37
VOS	86.17 ± 0.68	86.98 ± 0.80	94.31 ± 0.64
LogitNorm	89.62 ± 0.13	91.61 ± 0.17	94.30 ± 0.25
LogitNorm + PixMix	92.05 ± 0.24	94.21 ± 0.17	94.60 ± 0.25
CIDER	87.27 ± 0.21	88.68 ± 0.32	/
NPOS	85.63 ± 0.63	87.08 ± 0.78	/
ABN Module V2M	73.34	86.27	97.5
OE	89.83 ± 0.40	98.68 ± 0.48	94.63 ± 0.26
MCD	87.07 ± 0.38	88.38 ± 0.37	94.95 ± 0.04
UDG	86.31 ± 0.97	87.48 ± 0.71	92.36 ± 0.84
MixOE	86.88 ± 0.54	87.90 ± 0.73	94.55 ± 0.32
ABN Module V2M	**94.84**	**99.71**	97.5

Table 5.13. AUPR metric near-OOD detection on CIFAR10 Benchmark.

Chapter 6

Conclusions

In this thesis, we have addressed the problem of defining a novel approach to robust Out-Of-Distribution detection, studying and defining methods in the context of deepfake detection.
Two different approaches that share a common pipeline are proposed. Such procedure is composed of two cascade modules A and B. A is mainly constituted by the In-Distribtion classifier, which in this case addresses the deepfake detection problem (generically speaking could be any classification task and model), producing the detection outcomes and other auxiliary data that are shared with B.
Module B is a custom Abnormality module that makes inferences on which sample belongs to the ID and which to the OOD, employing the classification outputs and the auxiliary data from module A. The two different approaches differ on the typology of Module A involved and which kind of metadata is shared with B. Specifically are proposed solutions with Convolutional Neural Networks and with Vision Transformers.
Excluding the basic custom abnormality module, we have proposed three different network architectures to design module B, with different data reduction strategies, discussing the pros and cons of each one.
Three different modalities of Abnormality module training are covered in the analysis: applying proposed techniques to synthesize OOD data from ID samples, employing real OOD data, and mixing real and synthetic OOD data. We have defined train and test model procedures on a custom version of the CDDB dataset, a continual learning deepfake detection dataset. This dataset is converted in first instance to a large-size binary classification dataset and then through the different partitioning scenarios: Content, Group, and Mix, adapted to Out-Of-Distribution detection task. In addition, to get an OOD detection comparison on our custom adapted dataset, we have decided to reimplement several baselines and record their performances. Moreover, it's integrated into the study the CIFAR10 OOD benchmark, to extend the analysis concerning a wider range of available techniques in the field.
We have obtained promising performances with both approaches, although the ViT-based, thanks to the Tiny DeiT model, is computationally convenient and provides a faster convergence in the ID classification task. This solution records better OOD detection performances with respect to the usage of Convolutional models like the novel U-Net Scorer. Above all, we point out Abnormality modules V2 and V3, which

show remarkable efficacy also for future research.
Results comparable to those of the state-of-the-art are obtained involving real OOD data as outliers, which is appreciable in the benchmark results. Despite that, it's hard to find suitable outlier samples to be included in the training of these OOD detection modules. Indeed, the model performance is strictly related to the quality of these data, and how well they describe distributions different from one learned by the classifier, which could vary based on the task typology. Thus, we believe that the inclusion of only synthetic OOD data to train the models is the most attractive direction for this study since reflects a more general, and easy-use solution. Looking at the content scenario metrics and the Far-OOD benchmark evaluation, we have demonstrated the feasibility of this approach. In addition, deepfake detection over the three scenarios reaches satisfying results and is not affected by the OOD analysis performed.

6.1 Future work

In this section, we highlight the possible future directions in the domain of OOD detection. Using development workstations with greater computational power to train and test our models with various parameters, such as larger batch size or the number of epochs, would be great to explore several variants of the proposed networks and overall architectures. It's always important to remember computational efficiency as an additional objective, thus, should be avoided to focus the resources on heavier models, but rather to speed up the model tuning and research.
We conclude this work with possible further research steps which would be interesting to carry on.

- This study is predisposed to continue in a multi-class classification ID task related to the deepfake techniques. It would be really interesting to show how the OOD detection performances change when the nature of the ID task also varies. This discussion could be also extended by designing a different task and moving to a multi-label characterization of the problem.

- Synthetization techniques of OOD data presented in this work, are limited to five. Thus, a possible effective study is about new alteration techniques. This would be useful since the covariate shift learned from abnormality modules is still limited. The goal should be getting closer to the covariate shift produced by a semantic shift in the two distributions (ID and OOD) of data. Possible ideas are background and foreground object introduction, ambient occlusion, filtering the image at different frequencies (like the Gaussian blur), the inclusion of salt and pepper noise or speckle noise, employment of generative models to produce new altered samples, and so on.

- Extend the following procedure to video or a multimodal approach, trying to reduce the huge impact on the resources necessary for this study.

- Extract noise fingerprint thanks to denoising models, and use this additional information as new auxiliary data for the OOD discrimination. An attractive research would be regarding the impact of auxiliary data with different natures.

Understanding which kind of information is more effective for the Abnormality module could be crucial for this type of research.

- Reproduce the same study, exploiting higher resolution images, and analyze more in detail the impact of this additional information in both deepfake and OOD detection.

Bibliography

[1] Cifar 10 and cifar 100 datasets. https://www.cs.toronto.edu/~kriz/cifar.html. Dataset created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

[2] Reface App. https://reface.app/.

[3] Svhn dataset. http://ufldl.stanford.edu/housenumbers/. Dataset created by Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng.

[4] Which face is real? http://www.whichfaceisreal.com.

[5] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark, 2016.

[6] DeepBrain AI. Deepbrain ai—best ai video generator. https://www.deepbrain.io/.

[7] Shivangi Aneja and Matthias Nießner. Generalized zero and few-shot transfer for facial forgery detection. *arXiv preprint arXiv:2006.11863*, 2020.

[8] A. Y. Aravkin, J. V. Burke, L. Ljung, A. Lozano, and G. Pillonetto. Generalized kalman smoothing: Modeling and algorithms, 2016.

[9] Salma Benazzouza, Mohammed Ridouani, Fatima Salahdine, and Aawatif Hayar. A novel prediction model for malicious users detection and spectrum sensing based on stacking and deep learning. *Sensors*, 22(17):6477, 2022.

[10] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 2006.

[11] Google Research Blog. Contributing data to deepfake detection research.

[12] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2019.

[13] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1511–1520, 2017.

[14] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation, 2018.

[15] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017.

[16] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

[17] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild, 2013.

[18] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

[19] Luca Cultrera, Lorenzo Seidenari, and Alberto Del Bimbo. Leveraging visual attention for out-of-distribution detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4447–4456, 2023.

[20] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11065–11074, 2019.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[22] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[23] Terrance DeVries and Graham W. Taylor. Learning confidence for out-of-distribution detection in neural networks, 2018.

[24] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The deepfake detection challenge (dfdc) dataset, 2020.

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[26] FakeApp. FakeApp 2.2.0. `https://www.malavida.com/en/soft/fakeapp/`.

[27] Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. 'in-between'uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019.

[28] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[29] Dani Gamerman and Hedibert F Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC press, 2006.

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

[31] Luca Guarnera, Oliver Giudice, and Sebastiano Battiato. Deepfake detection by analyzing convolutional traces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 666–667, 2020.

[32] Luca Guarnera, Oliver Giudice, and Sebastiano Battiato. Level up the deepfake detection: a method to effectively discriminate images generated by gan architectures and diffusion models, 2023.

[33] Ammarah Hashmi, Sahibzada Adil Shahzad, Wasim Ahmad, Chia Wen Lin, Yu Tsao, and Hsin-Min Wang. Multimodal forgery detection using ensemble learning. In *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1524–1532, 2022.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[36] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks, 2018.

[37] Javier Hernandez-Ortega, Ruben Tolosana, Julian Fierrez, and Aythami Morales. Deepfakeson-phys: Deepfakes detection based on heart rate estimation, 2020.

[38] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations*, 2018.

[39] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960, 2020.

[40] Yihao Huang, Felix Juefei-Xu, Qing Guo, Yang Liu, and Geguang Pu. Fakelocator: Robust localization of gan-based face manipulations. *IEEE Transactions on Information Forensics and Security*, 17:2657–2672, 2022.

[41] Liming Jiang, Ren Li, Wayne Wu, Chen Qian, and Chen Change Loy. Deeperforensics-1.0: A large-scale dataset for real-world face forgery detection, 2020.

[42] Wenyu Jiang, Yuxin Ge, Hao Cheng, Mingcai Chen, Shuai Feng, and Chongjun Wang. Read: Aggregating reconstruction error into out-of-distribution detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14910–14918, 2023.

[43] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[44] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[45] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020.

[46] Hasam Khalid, Shahroz Tariq, Minha Kim, and Simon S. Woo. Fakeavceleb: A novel audio-video multimodal deepfake dataset, 2022.

[47] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[48] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[49] Pavel Korshunov and Sebastien Marcel. Deepfakes: a new threat to face recognition? assessment and detection, 2018.

[50] Pavel Korshunov and Sebastien Marcel. Deepfakes: a new threat to face recognition? assessment and detection, 2018.

[51] Pavel Korshunov and Sébastien Marcel. Improving generalization of deepfake detection with data farming and few-shot learning. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 4(3):386–397, 2022.

[52] Iryna Korshunova, Wenzhe Shi, Joni Dambre, and Lucas Theis. Fast face-swap using convolutional neural networks, 2017.

[53] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[54] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017.

[55] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

[56] Christophe Leys, Olivier Klein, Yves Dominicy, and Christophe Ley. Detecting multivariate outliers: Use a robust variant of the mahalanobis distance. *Journal of experimental social psychology*, 74:150–156, 2018.

[57] Chuqiao Li, Zhiwu Huang, Danda Pani Paudel, Yabin Wang, Mohamad Shahbazi, Xiaopeng Hong, and Luc Van Gool. A continual deepfake detection benchmark: Dataset, methods, and essentials, 2022.

[58] Jingyao Li, Pengguang Chen, Zexin He, Shaozuo Yu, Shu Liu, and Jiaya Jia. Rethinking out-of-distribution (ood) detection: Masked image modeling is all you need. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11578–11589, 2023.

[59] Ke Li, Tianhao Zhang, and Jitendra Malik. Diverse image synthesis from semantic layouts via conditional imle. 2019 ieee. In *CVF International Conference on Computer Vision (ICCV)*, pages 4219–4228, 2019.

[60] Lingzhi Li, Jianmin Bao, Hao Yang, Dong Chen, and Fang Wen. Faceshifter: Towards high fidelity and occlusion aware face swapping, 2020.

[61] Shengyin Li, Vibekananda Dutta, Xin He, and Takafumi Matsumaru. Deep learning based one-class detection system for fake faces generated by gan network. *Sensors*, 22(20):7767, 2022.

[62] Yuezun Li, Ming-Ching Chang, and Siwei Lyu. In ictu oculi: Exposing ai generated fake face videos by detecting eye blinking, 2018.

[63] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. Celeb-df: A large-scale challenging dataset for deepfake forensics, 2020.

[64] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks, 2020.

[65] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[66] Ming Liu, Yukang Ding, Min Xia, Xiao Liu, Errui Ding, Wangmeng Zuo, and Shilei Wen. Stgan: A unified selective transfer network for arbitrary image attribute editing, 2019.

[67] Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. Energy-based out-of-distribution detection, 2021.

[68] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

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

[70] Francesco Marra, Cristiano Saltori, Giulia Boato, and Luisa Verdoliva. Incremental learning for the detection and classification of gan-generated images, 2019.

[71] Scott McCloskey and Michael Albright. Detecting gan-generated imagery using color cues. *arXiv preprint arXiv:1812.08247*, 2018.

[72] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[73] Yuval Nirkin, Yosi Keller, and Tal Hassner. Fsgan: Subject agnostic face swapping and reenactment, 2019.

[74] Yuval Nirkin, Lior Wolf, Yosi Keller, and Tal Hassner. Deepfake detection based on the discrepancy between the face and its context, 2020.

[75] Jonas Oppenlaender. The creativity of text-to-image generation. In *Proceedings of the 25th International Academic Mindtrek Conference*, pages 192–202, 2022.

[76] Deng Pan, Lixian Sun, Rui Wang, Xingjian Zhang, and Richard O Sinnott. Deepfake detection through deep learning. In *2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, pages 134–143. IEEE, 2020.

[77] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization, 2019.

[78] Ivan Perov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Mr. Dpfks, Carl Shift Facenheim, Luis RP, Jian Jiang, Sheng Zhang, Pingyu Wu, Bo Zhou, and Weiming Zhang. Deepfacelab: Integrated, flexible and extensible face-swapping framework, 2021.

[79] Carsten Peterson and Eric Hartman. Explorations of the mean field theory learning algorithm. *Neural Networks*, 2(6):475–494, 1989.

[80] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[81] Md. Shohel Rana and Andrew H. Sung. Deepfakestack: A deep ensemble-based learning technique for deepfake detection. In *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pages 70–75, 2020.

[82] Richard A Redner and Homer F Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM review*, 26(2):195–239, 1984.

[83] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001.

[84] Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan. A simple fix to mahalanobis distance for improving near-ood detection. *arXiv preprint arXiv:2106.09022*, 2021.

[85] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 102–118. Springer, 2016.

[86] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[87] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[88] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Niessner. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[89] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[90] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces, 2018.

[91] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *Advances in neural information processing systems*, 30, 2017.

[92] Chandramouli Shama Sastry and Sageev Oore. Detecting out-of-distribution examples with in-distribution examples and gram matrices, 2020.

[93] Yupeng Shi, Xiao Liu, Yuxiang Wei, Zhongqin Wu, and Wangmeng Zuo. Retrieval-based spatially adaptive normalization for semantic image synthesis, 2022.

[94] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[95] Kumar Sricharan and Ashok Srivastava. Building robust classifiers through generation of confident out of distribution examples. *arXiv preprint arXiv:1812.00239*, 2018.

[96] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[97] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[98] Engkarat Techapanurak, Masanori Suganuma, and Takayuki Okatani. Hyperparameter-free out-of-distribution detection using cosine similarity. In *Proceedings of the Asian conference on computer vision*, 2020.

[99] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures, 2019.

[100] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos, 2020.

[101] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

[102] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021.

[103] Melissa Turcotte, Juston Moore, Nick Heard, and Aaron McPhall. Poisson factorization for peer-based anomaly detection. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 208–210. IEEE, 2016.

[104] An Introductory Tutorial and ET Jaynesy. Bayesian methods: General background? *Maximum Entropy and Bayesian Methods in Applied Statistics*, page 1.

[105] Anton Vasiliuk, Daria Frolova, Mikhail Belyaev, and Boris Shirokikh. Limitations of out-of-distribution detection in 3d medical image segmentation. *arXiv preprint arXiv:2306.13528*, 2023.

[106] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[107] Luisa Verdoliva. Media forensics and deepfakes: an overview. *IEEE Journal of Selected Topics in Signal Processing*, 14(5):910–932, 2020.

[108] Tomasz Walczyna and Zbigniew Piotrowski. Quick overview of face swap deep fakes. *Applied Sciences*, 13(11), 2023.

[109] Haoran Wang, Weitang Liu, Alex Bocchieri, and Yixuan Li. Can multi-label classification networks know what they don't know? *Advances in Neural Information Processing Systems*, 34:29074–29087, 2021.

[110] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024.

[111] Run Wang, Felix Juefei-Xu, Lei Ma, Xiaofei Xie, Yihao Huang, Jian Wang, and Yang Liu. Fakespotter: A simple yet robust baseline for spotting ai-synthesized fake faces. *arXiv preprint arXiv:1909.06122*, 2019.

[112] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot...for now. In *CVPR*, 2020.

[113] Saima Waseem, Syed R Abu-Bakar, Zaid Omar, Bilal Ashfaq Ahmed, and Saba Baloch. A multi-color spatio-temporal approach for detecting deepfake. In *2022 12th International Conference on Pattern Recognition Systems (ICPRS)*, pages 1–5. IEEE, 2022.

[114] DeepFakes Web. Make your own deepfakes [online app]. `https://deepfakesweb.com/`.

[115] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[116] Deressa Wodajo and Solomon Atnafu. Deepfake video detection using convolutional vision transformer, 2021.

[117] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[118] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, et al. Openood: Benchmarking generalized out-of-distribution detection. *Advances in Neural Information Processing Systems*, 35:32598–32611, 2022.

[119] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[120] Xin Yang, Yuezun Li, and Siwei Lyu. Exposing deep fakes using inconsistent head poses, 2018.

[121] Yijun Yang, Ruiyuan Gao, and Qiang Xu. Out-of-distribution detection with semantic mismatch under masking. In *European Conference on Computer Vision*, pages 373–390. Springer, 2022.

[122] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop, 2016.

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

[124] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019.

[125] Jiangning Zhang, Xianfang Zeng, Mengmeng Wang, Yusu Pan, Liang Liu, Yong Liu, Yu Ding, and Changjie Fan. Freenet: Multi-identity face reenactment, 2020.

[126] Jingyang Zhang, Nathan Inkawhich, Randolph Linderman, Yiran Chen, and Hai Li. Mixture outlier exposure: Towards out-of-distribution detection in fine-grained environments. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5531–5540, January 2023.

[127] Jingyang Zhang, Jingkang Yang, Pengyun Wang, Haoqi Wang, Yueqian Lin, Haoran Zhang, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, et al. Openood v1. 5: Enhanced benchmark for out-of-distribution detection. *arXiv preprint arXiv:2306.09301*, 2023.

[128] Jinsong Zhang, Qiang Fu, Xu Chen, Lun Du, Zelin Li, Gang Wang, Shi Han, Dongmei Zhang, et al. Out-of-distribution detection based on in-distribution data patterns memorization with modern hopfield energy. In *The Eleventh International Conference on Learning Representations*, 2022.

[129] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, October 2016.

[130] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.

[131] Bojia Zi, Minghao Chang, Jingjing Chen, Xingjun Ma, and Yu-Gang Jiang. Wilddeepfake: A challenging real-world dataset for deepfake detection. In *Proceedings of the 28th ACM international conference on multimedia*, pages 2382–2390, 2020.

[132] Bojia Zi, Minghao Chang, Jingjing Chen, Xingjun Ma, and Yu-Gang Jiang. Wilddeepfake: A challenging real-world dataset for deepfake detection, 2021.