LUISS

Degree Program in Data Science and Management

Course of Big Data And Smart Data Analytics

Detecting Threats in Generative AI: Security, Interpretability, and Ethical Issues in Deepfake Content

Prof. Irene Finocchi

SUPERVISOR

Prof. Emilio Coppa

CO-SUPERVISOR

ID 776831

CANDIDATE

Academic Year 2024/2025

Contents

Chapter 1

Introduction

1.1 Research Background and Motivation

In recent years, artificial intelligence has experienced an extraordinary acceleration in the development of models capable not only of analyzing and classifying information, but also of generating new content. Among the sectors that have benefited most from this progress is the automatic generation of multimedia content, including images, text, voices, and, in particular, video. The advent of increasingly sophisticated synthesis techniques has enabled the creation of digital representations that, in terms of realism, are often indistinguishable from authentic ones. So-called deepfakes represent one of the most emblematic manifestations of this transformation. By leveraging machine learning algorithms capable of capturing fine-grained details from training data, it is now possible to generate videos in which an individual's face is seamlessly replaced by another's, or in which a person's voice is cloned with impressive fidelity. These technologies, initially confined to academic and creative experimentation, have become accessible even to users with no technical expertise, due to the proliferation of open-source software, mobile applications, and intuitive interfaces that enable the production of deepfakes within minutes.

Among synthetic content, deepfake videos represent the most insidious threat, as they combine both visual and auditory channels, resulting in an immersive experience that facilitates deception. The high degree of realism, combined with the increasing speed of generation and dissemination on social media platforms, makes them particularly difficult to detect. This work is situated within a historical moment in which combating disinformation and media manipulation requires technological tools that are at least as advanced as those used to produce falsified content. The present work arises from the need to critically examine the vulnerabilities introduced by generative models, to understand how they can be exploited for malicious purposes, and to contribute to the development of robust and scalable strategies for the detection of manipulated media. Furthermore, the study seeks to highlight the ethical and regulatory implications that stem from the use of such technologies, with particular emphasis on the principles of transparency, accountability, and interpretability in artificial intelligence models.

1.2 Study Objectives

The main objective of this work is to critically examine the threats posed by generative models, with a specific focus on deepfake videos, and to evaluate viable solutions for their identification and interpretation. In particular, the thesis aims to:

- Describe the core technologies underlying generative models, with attention to the mechanisms employed in the creation of deepfakes;
- Exploring the risks and ethical challenges in the proliferation of synthetic content;
- Conduct an experimental study to assess the performance of classification models trained on real-world datasets [1], such as the DeepFake Detection Challenge (DFDC), by systematically varying architecture, data quantity, and experimental configurations, with the ultimate goal of identifying and proposing an effective, robust, and deployable detection model;
- Promote a reflection on the importance of interpretability and transparency in AI models.

This study seeks to integrate both theoretical and empirical perspectives, proposing practically grounded solutions in a fast-evolving context that demands adaptive and interdisciplinary responses.

1.3 Structure of the Work

This thesis is organized into nine chapters, each addressing a specific aspect of the detection of AI-generated content, with a particular focus on deepfake videos. Chapter 2 introduces generative models by outlining their historical development, theoretical foundations, and current applications across various domains. Chapter 3 explores the security, ethical, and regulatory challenges associated with deepfake video generation, highlighting the risks posed by misuse and the importance of governance frameworks. Chapter 4 defines the experimental context, detailing the dataset adopted, the issue of class imbalance, and the evaluation metrics used to assess model performance. Chapter 5 presents the architectural choices, with a focus on the rationale behind selecting EfficientNet-B7

[1]Real-world datasets refers to a dataset collected under realistic and uncontrolled conditions, characterized by noise, imperfections, variability in devices, and complex scenarios. It includes both manipulated and authentic audiovisual content, recorded by real actors in natural environments and subject to common distortions such as compression, lighting variations, and different camera angles *(Paullada et al., 2021. Data and its (dis)contents: A survey of dataset development and use in machine learning research).*

and a comparison with alternative architectures. Chapter 6 describes the full implementation pipeline, including preprocessing, dataset construction, training strategy, testing procedures, and experimental variations. Chapter 7 reports the empirical results, evaluating model performance in terms of accuracy, robustness, and computational efficiency. Chapter 8 provides a critical discussion of the findings, reflecting on the ethical and practical implications of deepfake detection, as well as the methodological limitations of the study. Finally, Chapter 9 summarizes the main conclusions and outlines future research directions.

Chapter 2

Overview of Generative Models

Generative models are an increasingly central component in the field of artificial intelligence, due to their ability to autonomously produce novel and realistic content. This chapter provides a theoretical and historical overview of such models, examining their main architectures, operational mechanisms, and applications across various domains, with particular attention to the practical and ethical implications associated with their use.

2.1 Definition of Generative Models

Generative models represent a fundamental class of machine learning algorithms whose primary objective is to learn the underlying distribution of observed data in order to generate new samples that share similar statistical and semantic characteristics. Unlike discriminative models, which are designed to estimate the conditional probability or to directly learn decision boundaries between classes or predicting output variables given an observation (e.g., classification or regression tasks), generative models focus on learning the joint probability distribution, encompassing both observations and labels. This approach enables not only data classification, but also the generation of new samples that are consistent with the learned statistical structure. Generative models provide a formal framework for dealing with challenges such as missing data, variable-length sequences, and latent variables. They simulate the underlying process that generated the data, making them particularly suitable for structured or noisy inputs (e.g., natural language, DNA, or images). However, when the primary goal is classification accuracy, discriminative models tend to achieve superior performance, as they concentrate exclusively on class separation (*Jaakkola et al., 1998. Exploiting generative models in discriminative classifiers*). Discriminative models do not attempt to model the input distribution, but rather directly learn a decision function or the conditional probability, thus improving predictive effectiveness in many practical applications. At their most basic level, generative models attempt to estimate the probability $p(x)$ of observing a given data point x within the data space. Once this distribution is learned, it becomes possible to sample new data points, generate variants of existing inputs, or reconstruct missing parts of an input. This makes generative models particularly useful in a wide range of domains, from the generation of realistic images, audio, and video, to the simulation of complex scenarios and the creation of synthetic data for training other models. These algorithms constitute the backbone of modern generative artificial intelligence (GAI), as they not only reproduce structures observed in real-world data but also imagine new, coherent configurations, thereby enhancing AI's capacity to emulate human creativity (*Ran He et al., 2025. Generative artificial intelligence: A historical perspective*).

The probabilistic nature of generative models allows them to capture the variability and uncertainty inherent in observed phenomena. This translates into the ability to simulate multiple scenarios starting from the same initial conditions, an essential feature in disciplines such as computational physics, biology, economics, and medicine. Generative models provide a conceptual framework for analyzing complex systems, as they link empirical data to explicit theoretical constructs through structured mechanisms, such as latent variables, graphical models, or agent-based simulations [2] (*Kris Sankaran et al., 2023. Generative models: An interdisciplinary perspective*). These tools allow researchers to test hypotheses, explore possible outcomes, and conduct in silico experiments that would otherwise be infeasible in the real world. Another key strength of generative models lies in their interdisciplinary applicability. Their use extends far beyond computer science, encompassing fields such as statistics, neuroscience, social sciences, engineering, and more. For instance, in molecular biology, they are employed to simulate the behavior of cell populations; in computational linguistics, to generate coherent natural language; and in computer graphics, to synthesize realistic faces or three-dimensional environments. From a theoretical standpoint, generative models are based on various techniques for representing and learning data distributions: some, such as normalizing flows [3], allow for exact probability density estimation, while others, such as Generative Adversarial Networks (GANs) or Variational Autoencoders (VAEs), discussed in the following sections, rely on indirect or approximate estimation methods. Despite their methodological differences, their ultimate goal remains the same: to construct mechanisms capable of learning the deep structure of data and producing new instances that are indistinguishable from real ones. Their growing adoption reflects the increasing centrality of these tools in the development of creative, robust, and interpretable AI systems.

[2]Agent-based simulation are computational models that simulate the interactions of autonomous agents to analyze emergent behaviors in complex systems.

[3]Normalizing flows are generative models that transform a simple distribution into a complex one using invertible functions, allowing exact probability density computation.

2.2 Historical Evolution and Development of Generative Models

The evolution of generative models reflects the broader transformation of artificial intelligence over more than seventy years, encompassing distinct phases characterized by different theoretical foundations, computational capabilities, and application contexts.

The first stage, spanning from the 1950s to the 1990s, was dominated by rule-based generative systems. These systems were manually constructed by human experts through the specification of symbolic rules designed to produce responses or behaviors within narrow, well-defined contexts. A notable example is ELIZA (1966), one of the earliest chatbots, which simulated a psychotherapist using a set of predefined lexical rules. Although pioneering, such systems were highly rigid and poorly generalizable, as they could not adapt to data or situations not explicitly foreseen in their rule sets. This rigidity represented a structural limitation that prevented their extension to more complex or dynamic domains (*Ran He et al., 2025. Generative artificial intelligence: A historical perspective*). As computational capabilities advanced and interest in statistical learning grew, a second phase emerged, centered around the adoption of probabilistic and graphical models. During this period, techniques such as Hidden Markov Models (HMMs)[4], Bayesian networks[5], conditional Markov networks[6], and other structured models capable of explicitly representing probabilistic dependencies between variables became widespread. These tools, supported by algorithms like Expectation-Maximization (EM)[7], allowed for more flexible and robust modeling of sequential, spatial, and noisy phenomena. This phase marked a shift in AI from a symbolic and deterministic logic toward a statistical and inferential paradigm, better suited to handling uncertainty and variability in real-world data (*Hatice Kübra Kılınç et al., 2024. Generative Artificial Intelligence: A Historical and Future Perspective*).

The third and most disruptive phase began in the 2010s with the advent of deep neural networks and the development of deep learning-based generative models, capable of learning complex representations from large datasets. Among the most representative architectures are Generative Adversarial Networks (GANs), introduced by Ian Goodfellow in 2014. GANs have revolutionized the generation of realistic images through a com-

[4]HMMs are statistical models in which the system is assumed to be a Markov process with unobserved (hidden) states. Commonly used for modeling time series and sequential data.

[5]Bayesian networks are probabilistic graphical models that represent variables and their conditional dependencies using a directed acyclic graph. Useful for reasoning under uncertainty.

[6]Markov networks, also known as Markov Random Fields, these are undirected graphical models used to represent the joint distribution of variables with a focus on spatial and contextual dependencies.

[7]Expectation-Maximization is an iterative algorithm used to estimate parameters in models with latent variables by alternating between inferring missing data (E-step) and optimizing parameters (M-step).

petitive process between two neural networks: a generator, which attempts to produce realistic data, and a discriminator, which seeks to distinguish real data from synthetic samples. This scheme is grounded in game theory, where the two models are engaged in a zero-sum game: the generator tries to deceive the discriminator, while the discriminator strives to detect the forgeries (*Goodfellow et al., 2020. Generative Adversarial Networks*). GANs belong to the category of implicit generative models, as they do not require the explicit formulation of a density function. The generator, driven by a random latent variable z, learns a function $G(z)$ capable of transforming unstructured noise into realistic samples that are statistically indistinguishable from real data. The discriminator, in turn, estimates the probability that a given sample originates from the real dataset rather than from the generative model. Although this adversarial dynamic is effective, it is often subject to training instability and may lead to phenomena such as mode collapse, where the generator produces a limited number of output variants, thereby reducing the diversity of generated results. Nonetheless, GANs have demonstrated outstanding effectiveness in visual domains such as facial image synthesis and image restoration, and they continue to represent one of the most promising architectures for high-resolution content generation. Despite their impressive results, GANs are affected by challenges such as mode collapse, in which the generator tends to produce only a limited variety of outputs (*Ruthotto et al., 2021. An introduction to deep generative modeling*).

In the same year, Variational Autoencoders (VAEs) were also proposed. These models combine the efficiency of autoencoding with a probabilistic framework based on latent spaces. VAEs enable more direct manipulation of internal data representations, providing a certain level of interpretability and control in the generation process. However, the visual quality of the outputs is often lower compared to GANs, resulting in blurrier or less detailed images. (*Kyle Stratis, 2024. What is Generative AI? A generative AI primer for business and technical leaders*).

In subsequent years, increasing attention was directed toward diffusion models, which simulate the generative process as a gradual and stochastic denoising[8] procedure that starts from pure noise. Although computationally intensive, these models have demonstrated superior performance in generating high-quality visual and audio content, achieving state-of-the-art results in various benchmarks (*Ran He et al., 2025. Generative artificial intelligence: A historical perspective*).

Starting in 2017, Transformer-based architectures gained widespread adoption. Initially designed for natural language processing, Transformers revolutionized the field through attention mechanisms that allow for efficient modeling of long-range dependen-

[8]Denoising is a process that removes noise from input data, often used in generative models to reconstruct clean signals from random or corrupted inputs.

cies in sequences. These architectures architectures were originally developed for natural language processing and later extended to various other domains. Their self-attention mechanism enables the efficient modeling of long-range dependencies in complex sequences, making them ideal for both textual and multimodal generation. Although highly versatile and powerful, Transformers require substantial computational resources, particularly during training, and their performance strongly depends on access to large-scale datasets. A representative example is ChatGPT, a generative model developed by OpenAI based on the Transformer family, specifically leveraging Large Language Models (LLMs). ChatGPT is classified as a foundation model, as it is pre-trained on a massive corpus of text and subsequently adapted to specific tasks through fine-tuning and reinforcement learning techniques. Its ability to generate coherent text, answer questions, write code, and simulate complex dialogues makes it one of the most prominent examples of the capabilities of Transformer-based models in content generation. However, like other LLMs, ChatGPT requires significant computational resources for training and inference, and poses challenges related to interpretability, fine-grained control, and potential biases in the training data. Their ability to parallelize training processes made them the de facto standard in many generative applications, including text, images, code, and video (*Kumar et al., 2024. Exploring the efficacy of adaptive learning platforms enhanced by artificial intelligence: A comprehensive review*).

Model	**Strengths**	**Limitations**
GAN	Highly realistic generation, suitable for images and videos	Training instability, risk of mode collapse
VAE	Controllability, interpretable probabilistic structure	Less sharp outputs, lower visual quality
Transformer	Versatility, excellent for sequences and multimodal content	High computational cost
Diffusion Models	High generative quality, robustness, and zero-shot generalization	Slow inference and high computational cost

Table 1: Comparison of current generative model architectures

Table 1 lists the most commonly used models in contemporary applications. As highlighted in Table 1, each generative architecture presents specific strengths and limitations: GANs excel in visual fidelity, VAEs offer greater control and interpretability, Transformers are distinguished by their versatility, while diffusion models provide extremely high-quality outputs at the expense of substantial computational costs. Nevertheless, alongside their technical capabilities, it is essential to consider the risks associated with the misuse of these models, thus requiring ethical reflection and the implementation of appropriate regulatory measures.

2.3 Applications of Generative Models

Thanks to their ability to create new, coherent, and adaptable content, generative models are now applied across a wide range of sectors, both creative and technical, reshaping how we work, learn, communicate, and produce knowledge.

- **Business and productivity:** Models such as ChatGPT are used to automate customer service, assist in drafting emails, documents, and reports, or generate marketing content in a fast and tailored manner. Other tools, like GitHub Copilot (based on Codex), support developers in writing code, reducing time and operational complexity (*Feuerriegel et al., 2024. Generative AI*).

- **Education:** Generative models function as virtual tutors, writing assistants, exercise generators, or tools for personalized learning. However, improper use may undermine academic integrity, facilitating practices such as plagiarism or cheating that challenge traditional assessment processes (*Nah et al., 2023. Generative AI and ChatGPT: Applications, challenges, and AI-human collaboration*).

- **Healthcare:** Generative AI can be used for patient support, automated report drafting, preliminary consultations, or diagnostic assistance. LLM-based tools have demonstrated the ability to pass medical exams and provide high-quality answers. Nonetheless, concerns remain regarding sensitive data privacy, lack of clinical validation, and the risk of misinformation (*Kenthapadi et al., 2023. Generative AI meets responsible AI: Practical challenges and opportunities*).

- **Multimedia content creation:** Systems like DALL·E and Stable Diffusion generate images from textual descriptions, while MusicLM or AudioLM create coherent and creative audio tracks.

- **Journalism and communication:** Generative models are used to write articles, generate news from structured data, and summarize transcripts.

- **Digital art and human-AI collaboration:** AI is increasingly integrated into artistic processes, enabling co-creation between humans and generative systems (*Weisz et al., 2024. Design principles for generative AI applications*).

Despite their numerous advantages, the widespread adoption of generative models also entails risks and responsibilities, which will be further addressed in the following chapter focusing on video generation.

Chapter 3

Security and Ethics in Deepfake Video Generation

This chapter explores the complex and multifaceted landscape of risks related to deepfake generation, with a focus on its disruptive consequences across political and social domains. Through an examination of disinformation, technical vulnerabilities, interpretability issues, and ethical concerns, the analysis highlights the urgent need for robust safeguards. Particular attention is given to model attacks, governance mechanisms, and the evolving regulatory environment, including the European Union's AI Act. The chapter aims to provide a critical framework for understanding how generative technologies, if not adequately regulated and governed, can undermine democratic stability, digital integrity, and public trust.

3.1 Deepfakes and Disinformation

The phenomenon of deepfakes represents one of the most emblematic manifestations of both the potential and the risks associated with generative models. Initially developed within academic and creative contexts for experimental and artistic purposes, the use of these technologies has rapidly transcended the boundaries of research, entering the public domain often without adequate control mechanisms or widespread awareness. The improper use of deepfakes has led to the creation of highly realistic audiovisual content, artificially generated and capable of perfectly mimicking the appearance, voice, and gestures of real individuals. This ability to produce "believable fakes" has opened particularly concerning scenarios in political, journalistic, and social contexts. The impact of deepfakes on disinformation is now documented in numerous real-world cases. Manipulated videos portraying political leaders issuing compromising or inflammatory statements have been deliberately disseminated to influence public opinion, generate panic, or delegitimize institutional figures. A significant example is the case of *Zao*, a Chinese app launched in 2019 that enabled users to superimpose their faces onto those of famous actors in a matter of seconds. The application reached millions of downloads within a few days, highlighting how quick and easy it has become to produce realistic and viral deepfake content (*Y. Patel et al., 2023. Deepfake generation and detection: Case study and challenges*). This episode raised concerns not only about the security of biometric data, but also about the broader implications of such technologies in contemporary digital communication. Digital platforms, especially social networks, face a dual ethical and operational dilemma: on one hand, they provide the tools and distribution channels that amplify the reach of manipulated content; on the other, they are expected to develop moderation policies and detection tools capable of identifying deepfakes in a timely manner and limiting their dissemination. However, the speed at which this content spreads makes it difficult to respond promptly and effectively, thereby undermining citizens' trust in the authenticity of online information and contributing to the so-called "crisis of truth" (*S. Gupta et al., 2025. Deepfake Overview: Generation, Detection, Risks and Opportunities*). In this scenario, the proliferation of deepfakes constitutes a tangible threat to democratic stability, social cohesion, and institutional security, necessitating coordinated responses at technological, legal, and cultural levels.

3.2 Manipulation of Textual and Visual Content

Deepfakes are no longer limited to traditional face swapping, but now include advanced techniques such as morphing, face reenactment, and landmark alignment. Morphing enables the gradual transformation of one face into another by progressively blending distinct facial features, and was initially used in the film industry for special effects. Face reenactment, on the other hand, allows the transfer of one person's facial expressions onto another subject in real time, using algorithms capable of reconstructing micro-expressions with high fidelity. Finally, landmark alignment relies on the automatic detection of key facial points (e.g., eyes, mouth, eyebrows), which are used to digitally deform and synchronize the target image in a manner consistent with the movements of the source subject. These techniques, although originally developed for legitimate and useful purposes, such as rehabilitative medicine, cinema, or the creation of avatars in augmented reality, are now increasingly employed in activities that raise serious ethical concerns. Manipulated content can be used in fraud, scams, personal revenge, the spread of fake news, and even the fabrication of false judicial evidence. The uncontrolled dissemination of these tools highlights the urgent need not only for effective detection methods, but also for normative and cultural frameworks capable of governing their use.

3.3 Attacks on Generative Models

Although generative models are extraordinarily powerful, they are subject to structural vulnerabilities that can significantly compromise their reliability and security. Among the most relevant threats are adversarial attacks and data poisoning, both of which exploit intrinsic weaknesses of the models to alter, deceive, or manipulate them for malicious purposes.

Adversarial attacks are based on the introduction of minimal perturbations, often imperceptible to the human eye, into the input data fed to the model, such as images, texts, or audio signals. Technically, an adversarial attack consists of computing a specific perturbation to apply to a given input, with the goal of misleading the model. This perturbation is represented as a vector, often matching the input in size and shape, and is calculated using the gradient of the model's loss function with respect to the input. In practice, the attacker performs a targeted optimization: identifying a direction in the data space where even a slight shift can cause the model to produce an incorrect response. The effectiveness of these attacks lies in their ability to explore marginal regions of the data distribution learned during training, where the model's behavior is less stable and predictable. These perturbations operate within the model's "blind spots," exploiting the lack of coverage in the training data to trigger erroneous outputs. For example, an image of a dog can be subtly modified (by changing a few pixels imperceptibly) so that the model misclassifies it as a cat, even though it remains visually identical to a human observer. In the case of a generative model, the same principle can be applied to produce a completely different face than expected, simply by slightly altering the input image or the initial prompt. Equally insidious is the threat known as data poisoning, which occurs during the training phase of the model. Here, the attacker deliberately introduces corrupted or manipulated data into the training set, thereby distorting the model's learning process. The consequences may include anomalous behavior, the insertion of backdoors that can be triggered by specific inputs, or a drastic reduction in generative capabilities under certain conditions. The severity of these attacks is amplified by the increasing adoption of open-source models, which are often reused or adapted without appropriate security measures. In environments where transparency and code sharing are encouraged, the absence of input and training data control can constitute a true systemic vulnerability.

3.4 Reliability in Generative Models

The growing diffusion and increasing sophistication of generative models necessitate a thorough reflection on their security, understood not only as protection against external attacks but also as the capacity to operate in a reliable, controllable, and transparent manner. Unlike traditional predictive models, generative models are capable of autonomously producing new content, which makes them potentially more dangerous if compromised. A primary line of defense consists in improving training techniques. *Adversarial training*, for instance, involves inserting perturbed examples during training to enhance the model's robustness (*W. Zhao et al., 2022. Adversarial training methods for deep learning: A systematic review*). While effective, this approach significantly increases computational costs and may reduce the creative capabilities of the model. For example, during the adversarial training of a generative image model, facial images with subtle, imperceptible distortions in key features are deliberately included in the training dataset. This process helps the model learn to recognize such inputs as valid, rather than being misled by them. Consequently, when the model encounters similarly perturbed inputs during inference, it is more likely to respond appropriately without generating distorted or erroneous outputs.

However, security cannot rely solely on technological solutions. Technical measures must be complemented by a suitable regulatory framework. The recent *Artificial Intelligence Act* (2024) of the European Union, for example, represents a major step in this direction by imposing requirements related to transparency, traceability, technical documentation, and risk assessment for systems considered to be high-impact, such as generative models. These obligations also include controls over the datasets used for training and the implementation of mitigation mechanisms against foreseeable risks (*AI Act, 2024*). The security of generative models must be conceived as a balance between innovation, protection, and responsibility. Securing these systems means safeguarding not only the integrity of the technologies themselves, but also the public's trust in artificial intelligence and its growing role in contemporary society.

3.5 Interpretability Methods

The expanding adoption of generative models in increasingly critical and sensitive domains has made the issue of interpretability indispensable. Unlike traditional models, generative models operate in highly opaque ways, often described as *black boxes*[9], making it difficult for humans to understand the decision-making process behind the generated results. This raises crucial questions not only on a technical level, but also on ethical and legal grounds, particularly in contexts where automated decisions can have significant consequences for individuals. The *Explainable AI* (XAI) approach aims to develop methodologies and tools capable of making artificial intelligence models interpretable by providing human-understandable explanations of the model's internal functioning, variable relationships, and the reasoning behind predictions or generations (*Xu, Feiyu, et al., 2019. Explainable AI: A brief survey on history, research areas, approaches and chal-*

[9]A black box refers to a system whose internal workings are not visible or understandable to the user, making it difficult to trace how inputs are transformed into outputs.

lenges). Techniques such as *SHAP*[10], *LIME*[11], and *saliency maps*[12] are among the most widely used to analyze decisions made by complex models, including the behavior of visual and textual generators. These tools help identify, for instance, which visual or conceptual features had the most influence on the generation of a given output, thereby improving traceability and fostering trust in AI systems. In generative models based on GANs, interpretability becomes even more complex due to the interactive nature between the generator and the discriminator, making it difficult to causally attribute a decision to a specific part of the network. Nevertheless, dedicated tools for latent space analysis are becoming more common, allowing researchers to understand how certain variables (e.g., skin tone, facial expression, vocal tone) are encoded and manipulated during the generation process. These analyses not only help improve the quality of the generated content, but also serve to highlight and correct potential biases present in the data or the model architecture.

3.6 Security and Governance in Generative Models

In addition to the technical protection of models, it is now essential to address the issue of governance, namely the set of regulatory, ethical, and organizational mechanisms that regulate the development, distribution, and responsible use of generative models. The governance of generative AI cannot be limited to technical measures such as algorithmic security or robustness against attacks, but must also embrace ethical, social, and legal dimensions. One of the most delicate aspects concerns transparency and traceability: who created a given artificially generated content? Has it been modified? From which model does it originate?

These questions are being addressed through mechanisms such as digital watermarking and metadata tracking systems, which make it possible to identify the source of generated content. At the same time, there is increasing demand for shared responsibility among developers, platforms, and end-users. Debates on regulation often focus on the effects of deepfakes, overlooking the crucial role of developers, who may choose whether to release source code, how to distribute their models, and whether to impose access restrictions or ethical safeguards. In this sense, governance is not only a legal matter but also a cultural and professional one: it involves operational choices, responsible design practices, and the commercial logic of those who develop or deploy such tools. Ensuring the

[10]SHAP (SHapley Additive exPlanations) is a method that assigns an importance value to each feature, based on game theory, to explain the output of any machine learning model.

[11]LIME (Local Interpretable Model-agnostic Explanations) is a technique that approximates a complex model locally with an interpretable one to explain a specific prediction.

[12]Saliency maps are visual representations that highlight the most influential regions of an input (e.g., image) that affect a model's decision, commonly used in computer vision.

security of generative models requires a multi-level governance approach that integrates legal standards, ethical principles, and professional accountability. Only a convergence of technology, regulation, and social awareness can curb abuse, promote responsible innovation, and ensure that generative models operate in accordance with democratic values and fundamental rights.

3.7 Ethical Issues and Bias

Generative models, if not carefully balanced, can perpetuate discrimination based on gender, race, sexual orientation, religion, or social class, not as a result of deliberate design, but as a statistical consequence of machine learning. These biases may manifest in various ways: through the generation of stereotypical images, preferences for certain facial traits in synthetic outputs, or the production of texts containing exclusionary or biased language. From the standpoint of responsibility, another critical concern is the issue of non-consensual use, particularly in cases of pornographic deepfakes, where the faces of real individuals are superimposed onto bodies in sexually explicit contexts without their consent. This phenomenon has been widely documented as one of the most prevalent and damaging forms of generative model abuse (*M. Pawelec 2024. Decent deepfakes? Professional deepfake developers' ethical considerations and their governance potential*). Beyond the violation of personal dignity, such cases raise problems related to digital identity, the right to be forgotten, and image rights. An additional ethical issue is tied to the dilution of truth and the risk of an epistemological crisis: if synthetic content becomes indistinguishable from authentic material, how can we be certain of the origin of what we see, hear, or read? In this context, the spread of deepfakes represents a threat to social trust, undermines media credibility, and facilitates the dissemination of propaganda, disinformation, and fraud.

3.8 Regulation and Policy

The growing power and diffusion of generative models has highlighted the need for a clear and binding regulatory framework capable of governing their development and use. In this context, the European Union's *AI Act* (Regulation EU 2024/1689) represents the world's first horizontal legislative initiative aimed at regulating artificial intelligence through a risk-based approach. The AI Act introduces a classification of AI systems according to risk level, minimal, limited, high, and unacceptable, with increasing obligations for more critical systems. Generative models, and in particular general-purpose systems with high societal impact, known as *GPAI*[13], are subject to specific requirements: transparency in functioning, technical documentation, risk assessment, and the implementation of safeguards against misuse. The regulation also mandates that AI-generated content must be clearly identified as such, to prevent deception and uphold transparency (*AI Act, 2024*). Beyond the AI Act, other European regulations contribute to building an integrated legal ecosystem. The *Digital Services Act* (DSA), for example, governs digital platforms by imposing content moderation requirements, algorithmic transparency, and protections against illegal content. Furthermore, the 2024 EU directive on combating gender-based violence explicitly prohibits the distribution of non-consensual pornographic deepfake material, acknowledging the gravity of the phenomenon and protecting victims through criminal sanctions (*M. Pawelec 2024. Decent deepfakes? Professional deepfake developers' ethical considerations and their governance potential*). At the international level, regulatory and cooperative initiatives are also multiplying. UNESCO, the OECD, the G7, and the Council of Europe have adopted guidelines and recommendations on AI ethics, with particular attention to the protection of human rights, accountability, and sustainability. The United States recently proposed an *AI Bill of Rights*, while China has introduced specific legislation for *deep synthesis* providers[14] (*AI ACT Brief, 2024*). Despite these advances, significant challenges remain: the rapid pace of technological evolution often outstrips the capacity of institutions to regulate effectively; furthermore, the global nature of digital platforms makes it difficult to harmonize standards across legal jurisdictions. To address these challenges, it is essential to promote a coordinated, multi-level, and proactive approach involving legislators, tech companies, the scientific community, and civil society. Regulating generative models requires a balance between innovation, the protection of fundamental rights, and shared responsibility. The AI Act marks a fundamental step in this direction, but it will need to be continuously updated and expanded to respond to the new threats and opportunities emerging from an ever-evolving ecosystem.

[13]GPAI (General-Purpose AI) refers to AI systems designed for a wide range of tasks, which can be adapted to multiple domains and use cases beyond their original context.

[14]Deep synthesis refers to technologies that generate or alter content (e.g., audio, images, or video) using AI, often for realistic simulation or imitation purposes.

Chapter 4

Experimental Context, Dataset, and Evaluation Criteria

This chapter outlines the experimental objectives and the application context related to deepfake detection, with particular focus on the data collection and preprocessing phases. Furthermore, it discusses the class distribution and the issue of imbalance, which represents one of the main challenges in binary classification tasks under real-world conditions. Finally, the evaluation metrics adopted for model assessment are presented and justified in light of the project's specific goals and the unique characteristics of the application domain.

4.1 Experimental Objectives and Context

The primary objective of this study is to design a model for the detection of artificially generated content, with a specific focus on deepfake videos, that integrates the best strategies emerging from recent literature. The aim is to extract the most effective insights from each examined architecture. The adopted approach seeks to strike a balance between accuracy and computational efficiency, in order to achieve optimal results both in terms of performance metrics and processing time. This model is therefore intended to combine the predictive robustness of state-of-the-art architectures with the scalability and adaptability required for deployment in real-world scenarios. The increasing dissemination of deepfakes poses a concrete and multidimensional threat: misinformation in the media, financial fraud, reputational damage, privacy violations, and political manipulation are just a few of the implications associated with realistically altered audiovisual content. The easy accessibility of automatic generation tools and the speed at which such content can spread online make the development of scalable, reliable, and adaptable detection systems a pressing need. An important source of inspiration for the development of the proposed model was the Deepfake Detection Challenge (DFDC), which provided a significant reference in terms of methodology and application scenarios. The purpose, dataset structure, and experimental implications of this initiative will be explored in the following sections in order to contextualize the design choices adopted. The proposed approach does not aim to replicate existing models, but is instead based on a critical and selective analysis of the most effective strategies, with the goal of leveraging their strengths while addressing

their limitations in terms of generalization, efficiency, and adaptability. In this sense, the present work positions itself as a reasoned synthesis of best practices, oriented toward the construction of a robust and scalable system capable of operating effectively on real-world data.
The implementation of deepfake detection systems presents several computational challenges. Memory management is often complex, as processing high-resolution images requires significant resources, especially during the training phase. Moreover, the long times required to complete training on large datasets highlight the need for careful planning of experimental configurations. An inherent trade-off also emerges between predictive performance and computational cost: increasing the amount of data or the number of training epochs can improve accuracy but may lead to overfitting and prohibitively long training times. Scalability represents an additional critical aspect: systems must be able to operate in high-throughput real-world environments, ensuring fast response times without compromising detection reliability. Within this context, the present work fits into the broader field of generative model security, with the aim of critically assessing the detection capabilities offered by deep learning models. At the same time, it seeks to propose a solution that meets key requirements in terms of accuracy, computational efficiency, and robustness one that proves effective not only in controlled environments, but also in realistic and highly variable scenarios.

4.2 Dataset Structure and Composition

For the purpose of this experimental study, we adopted the *DeepFake Detection Challenge Dataset*[15] (DFDC), made available by Meta AI as part of the international competition launched in 2020 on *Kaggle*[16]. The DFDC is one of the most authoritative and large-scale initiatives in the field of synthetic content detection, and is still considered among the most impactful global benchmarks for video manipulation detection using artificial intelligence.

This dataset represents one of the most comprehensive public collections of synthetic video content generated through face-swapping techniques based on generative models. It was specifically designed to support the training and validation of automated deepfake detection algorithms. The DFDC dataset provides approximately 470 GB of video data, divided into 50 ZIP files of around 10 GB each. Each archive contains a mix of real and manipulated (fake) videos, maintaining a similar distribution between the two

[15]The DeepFake Detection Challenge (DFDC) dataset is a large-scale benchmark dataset developed by Meta AI to support research on detecting synthetic media, particularly deepfake videos.
[16]Kaggle is an online platform for data science and machine learning competitions, offering datasets, code sharing, and a collaborative community of researchers and practitioners.

classes within each individual file. However, due to computational limitations, only a fraction of the full dataset was used in this study. The selected subset was structured to preserve the diversity and complexity of the original collection, while keeping the dataset size manageable for training and evaluation phases. The version of the dataset used in this study comprises approximately 120 GB of data, consisting of over 25,300 ten-second video clips recorded by 3,426 professional actors (all of whom provided informed consent) in naturalistic settings, both indoor and outdoor, under varying lighting conditions, resolutions, and camera angles.

Figure 1: Example of videos contained in the dataset, showing subjects in different settings, environments, and situations. Source: DFDC

The generation techniques include a variety of approaches: autoencoders, GANs (e.g., StyleGAN, FSGAN), morphing, face reenactment, and landmark alignment-based methods, thus representing a broad spectrum of possible manipulations. Specifically, The dataset adopted for the experimental phase of this study was organized as follows:

- **Training set:** 10,420 videos (approximately 60 GB)
- **Test set 1 (standard):** 3,464 videos (approximately 10 GB)
- **Test set 2 (extended):** 11,335 videos (approximately 50 GB)

During training, the dataset was also progressively split into incremental subsets, with the aim of evaluating the impact of data quantity on the model's learning capacity and identifying an optimal trade-off between predictive accuracy, robustness to variability, and computational efficiency. This aspect will be examined in detail in the following chapters.

4.3 Class Distribution and Imbalance

One of the most critical aspects from both a statistical and methodological perspective in building the experimental model lies in the pronounced asymmetry between the classes within the dataset. Specifically, in the training data used for model development, a strong predominance of the FAKE class is observed, accounting for approximately 90% of the samples, while the REAL class constitutes only the remaining 10%. The dataset was intentionally structured to emphasize the challenges posed by the overrepresentation of manipulated content, mirroring realistic scenarios encountered in operational contexts. Even in the two test sets employed during the validation phase, although the imbalance is slightly less severe, a significant skew remains: the REAL class represents about 15% of the data, while the remaining 85% corresponds to FAKE content.

This configuration should not be considered a statistical anomaly, but rather the result of a deliberate design choice. In practical applications such as social media monitoring, digital surveillance, or forensic analysis, it is common to encounter datasets where suspicious or altered content outnumbers authentic material. This is often due to selection criteria as well as the high incidence of digital manipulations in high-risk environments. Therefore, the presence of an imbalanced class distribution, while undoubtedly posing challenges during training, is also necessary to realistically simulate the operational conditions of automatic detection systems. However, from a supervised learning standpoint, such an imbalanced distribution introduces substantial risks of model bias. Without appropriate countermeasures, the algorithm tends to optimize the objective function by assigning relatively low penalties to errors on the minority class, favoring predictions toward the dominant class. The result is an apparently high accuracy that conceals poor performance on metrics sensitive to the underrepresented class, such as recall for REAL videos, leading to an increased rate of false negatives.

To prevent the algorithm from learning incorrect heuristics based solely on frequency distribution, a class balancing strategy was adopted by weighting each label's contribution in the loss function. In particular, PyTorch's `BCEWithLogitsLoss`[17] function was modified through the introduction of a parameter known as `pos_weight`, further detailed in later chapters, which allows the model to assign a higher penalty to errors made on the minority class. The weights were calculated to be inversely proportional to the relative frequency of the two classes in the training set, thereby increasing the penalty for misclassifying REAL content as FAKE, and vice versa. This strategy significantly mitigated the effect of structural imbalance, improving performance on minority-sensitive metrics without compromising the model's overall ability to distinguish between real and manipulated content. Additionally, the adoption of this mechanism helped achieve a more calibrated predictive behavior, reducing the risk that the model would rely on misleading statistical shortcuts linked to class dominance.

[17]`BCEWithLogitsLoss` is a binary cross-entropy loss function in PyTorch that combines a sigmoid activation with the standard binary cross-entropy loss, improving numerical stability.

4.4 Evaluation Metrics for Deepfake Detection Models

Evaluating the performance of a deepfake detection model requires careful consideration of the chosen metrics, especially in highly imbalanced scenarios where the majority of content is either artificially generated or real. In this context, it is essential to distinguish between metrics suitable for model optimization (e.g., during training) and those appropriate for operational evaluation, that is, for simulating the system's behavior in real-world conditions. The metrics analyzed in this study include probabilistic, classification-based, and weighted measures, each offering different insights into the model's behavior.

In binary classification tasks applied to deepfake detection, it is customary to represent the two classes using numerical labels: 0 for authentic (REAL) content and 1 for artificially manipulated (FAKE) content. This binary encoding, widely adopted in both academic and industrial settings, not only simplifies the computational implementation of machine learning models but also aligns naturally with the probabilistic interpretation of model outputs. Specifically, the use of the sigmoid activation function in neural networks enables the assignment of a continuous probability between 0 and 1 to each sample, interpreted as the likelihood that the content belongs to the FAKE class. The final class assignment depends on a predefined decision threshold (typically 0.5), above which the content is labeled as manipulated, and below which it is considered real.

4.4.1 Logarithmic Loss (Log Loss)

Log Loss, also known as binary cross-entropy, is a probabilistic metric that measures the distance between the predicted probabilities and the true class labels. It is formally defined as:

$$\text{Log Loss} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right] \qquad (1)$$

where:

- $y_i \in \{0, 1\}$ is the true label of observation i (0 = real, 1 = fake),
- $p_i \in (0, 1)$ is the predicted probability that the observation is fake,
- n is the total number of samples.

This metric is continuous, differentiable, and suitable for optimization via gradient descent. It heavily penalizes incorrect predictions made with high confidence: for example, misclassifying a sample with a predicted probability of 0.99 incurs a much greater penalty than an incorrect prediction with a probability of 0.6. Although it does not operate directly in terms of false positives (FP) or false negatives (FN), Log Loss effectively reflects the overall probabilistic quality of the predictions.

In scenarios characterized by significant class imbalance, it is recommended to complement Log Loss with additional metrics that offer better insight into error structure, aiming for a more comprehensive and balanced evaluation of model performance. Nonetheless, Log Loss was adopted as the official evaluation metric in the DeepFake Detection Challenge (DFDC) to assess the performance of participating models, precisely due to its sensitivity to calibration and the confidence of predictions.

4.4.2 Precision and Recall

Precision measures the proportion of true positives among all instances that the model has classified as positive (i.e., fake):

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \textit{(2)}$$

where:

- TP: true positives (fake videos correctly identified as fake),
- FP: false positives (real videos incorrectly classified as fake).

A high precision indicates that the model is reliable when it chooses to classify a video as manipulated. However, it does not account for false negatives; hence, a model that correctly identifies only a few deepfakes with high confidence may achieve high precision while still being inadequate in practice.

Recall, on the other hand, measures the model's ability to correctly identify all fake videos present in the dataset, where FN denotes false negatives, fake videos incorrectly classified as real:

$$\text{Recall} = \frac{TP}{TP + FN} \qquad \textit{(3)}$$

A high recall is crucial in contexts where missing a deepfake could lead to severe consequences (e.g., misinformation or forensic misjudgment). However, similar to precision, this metric alone does not ensure a balanced evaluation of the classifier.

To overcome the limitations of using precision and recall in isolation, it is common practice to consider both metrics jointly. While precision reflects the reliability of positive predictions, recall reflects the model's ability to capture the actual number of manipulated videos. The integration of these two aspects allows for a more complete and balanced assessment of model performance. In particular, their trade-off is effectively summarized through the *F1-score*, which represents the harmonic mean of precision and recall, offering a unified indicator of the model's robustness in scenarios characterized by class imbalance and high variability in detection risk.

4.4.3 F1 Score

The F1 score is the harmonic mean between precision and recall, and is defined as:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \qquad (4)$$

Unlike the arithmetic mean, the harmonic mean tends to emphasize the lower of the two values. In other words, the F1 score will be high only when both precision and recall are high. If either metric is low, the resulting F1 score will also be low. This property is particularly useful in binary classification contexts where it is crucial to maintain a balance between the reliability of positive predictions (precision) and the model's ability to identify all actual positive cases (recall). The use of the harmonic mean reflects a practical need: to avoid having high precision compensate for low recall (or vice versa) in the computation of an aggregate metric. In summary, the F1 score heavily penalizes models that perform well in one dimension at the expense of the other, providing a more conservative and balanced view of overall performance.

However, it is important to note that the F1 score does not take true negatives (TN) into account. In the presence of imbalanced datasets, this limitation can result in a misleading representation if not interpreted alongside other metrics. For this reason, the analysis incorporated additional complementary metrics that account for the full error structure (AUC-ROC), including true negatives, to obtain a more complete understanding of model behavior. Furthermore, during the training phase, a weighted version of the F1 score was implemented to strengthen its sensitivity to class imbalance and ensure balanced optimization across both classes.

4.4.4 AUC-ROC

AUC-ROC represents the Area Under the ROC Curve (Receiver Operating Characteristic), which illustrates the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) across different decision thresholds:

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN} \qquad (5)$$

As the classifier's decision threshold varies, the proportion of true positives and false positives changes, thus shaping the ROC curve. The area under this curve (AUC) provides a synthetic measure of the model's ability to distinguish between classes. An AUC value of 1.0 indicates a perfect classifier capable of distinguishing real and fake content without errors, while an AUC of 0.5 corresponds to a random classifier with no discriminative power. Since AUC is independent of the chosen threshold, it is particularly useful for comparing the global quality of different models. However, this metric does not directly reflect the operational impact of classification errors in highly imbalanced scenarios, where distinguishing between false positives and false negatives may carry very different consequences depending on the specific application. For this reason, in this study, AUC was used in combination with the previously described metrics to provide a more complete, balanced, and contextually relevant assessment of model performance.

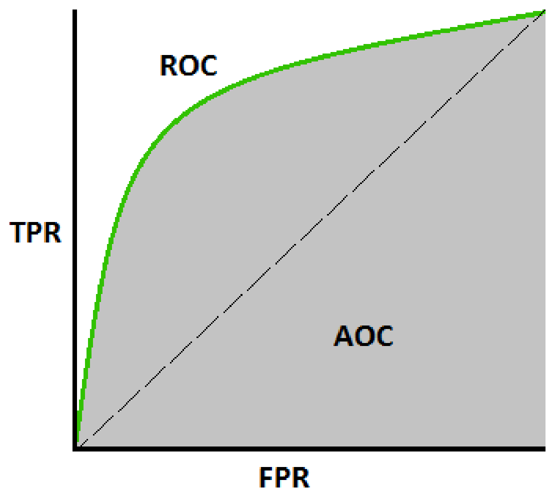


Figure 2: Illustration of the ROC Curve and AUC Concept.

4.4.5 Weighted Precision

In real-world applications, such as social media monitoring, automated moderation, or forensic investigations, the proportion of manipulated content compared to authentic material is often extremely low. This creates a significant discrepancy relative to the datasets used for training and evaluating models, in which the "fake" class is often overrepresented for methodological reasons.

To address this asymmetry, the *Weighted Precision* (wP) metric has been introduced. It adjusts for the real-world prevalence of fake content in the target operational environment:

$$wP = \frac{TP}{TP + \alpha \cdot FP} \qquad \text{with} \qquad \alpha = \frac{\text{real-world prevalence}}{\text{dataset prevalence}} \tag{6}$$

where:

- TP refers to true positives, i.e., fake content correctly classified as fake;
- FP denotes false positives, i.e., real content incorrectly classified as fake;
- α is a correction factor that scales the weight of false positives according to the difference between the prevalence of manipulated content in the dataset and its prevalence in real-world operational contexts.

Several studies in the literature simulate prevalences greater than 5% to reflect high-risk or pre-filtered operational scenarios, where the likelihood of encountering manipulated content is significantly higher than the global average (*Zhou et al., 2023; Chen et al., 2023*). In alignment with these academic references, this analysis adopts a simulated real-world prevalence of deepfake content equal to 5%. This coefficient was used to proportionally modulate the impact of false positives, which, although statistically less likely in real-world contexts, can lead to significant operational and decision-making consequences, particularly in automated detection systems.

From a practical standpoint, this metric proves especially valuable when transitioning from academic validation to real-world deployment, where the goal is not merely to maximize performance on a balanced test set, but rather to minimize the operational risks associated with misclassifications. The use of weighted precision thus allows the evaluation of model performance to be aligned with real-world criteria such as risk, impact, and accountability, which are critical in applied contexts where these models are actually deployed.

4.5 The Integrated Use of Metrics in Model Validation

Evaluating the performance of a deepfake detection model requires a critical and comprehensive analysis of the metrics employed. In the context of binary classification applied to synthetic content, often characterized by class imbalance, traditional metrics, while essential during the training phase, are often insufficient to capture the model's real-world effectiveness. For this reason, based on the literature, in this chapter we have overviewed an evaluation strategy based on a heterogeneous set of indicators, including probabilistic metrics (such as Log Loss), classical classification metrics (Precision, Recall, F1-score), global performance indicators (AUC-ROC), and correction-based measures tailored for real-world application (weighted precision using the coefficient α).The use of binary encoding (0/1) enabled an efficient computational formulation of the problem, while the

sigmoid activation function allowed the model to output interpretable probabilistic predictions, an essential feature for applying dynamic thresholds and calculating calibration-sensitive metrics. For instance, Log Loss proved particularly valuable in penalizing high-confidence misclassifications more severely, offering a continuous and differentiable signal ideal for the optimization phase. Nevertheless, metrics such as precision and recall, while crucial for understanding binary prediction performance, suffer from structural limitations when used in isolation. The adoption of the F1-score as a harmonic mean allowed for a balanced synthesis of the model's ability to identify manipulated content without overlooking the reliability of its predictions. However, this balance does not account for true negatives, thus necessitating the inclusion of global metrics such as AUC-ROC, which evaluates overall model behavior across varying decision thresholds.
Finally, weighted precision with coefficient α has emerged as a key metric in bridging the gap between theoretical evaluation and real-world deployment. In scenarios where deepfake content is less prevalent but highly critical from an operational standpoint, this metric enables the targeted penalization of false positives, reducing the risk of misclassifying authentic content. The introduction of α, calculated as the ratio between real-world prevalence and dataset prevalence, allowed for the simulation of high-risk contexts such as forensic or investigative domains, where the significance of individual errors is amplified compared to generic settings. In summary, this chapter has shown that a robust evaluation of deepfake detection models cannot rely on a single metric. Instead, it must adopt a coherent and context-aware combination of indicators capable of addressing both methodological rigor and practical requirements. Only through this integrated approach is it possible to ensure an effective measurement of a model's discriminative capability while anticipating the risks and limitations of its real-world application.

Chapter 5

Model Engineering and Architectural Choices

The effectiveness of a deep learning–based classification system largely depends on the architecture of the model employed. This chapter presents the architecture selected as the foundation of the experimental framework, with a specific focus on the EfficientNet family, considered among the most advanced and efficient convolutional neural network architectures available. After a general overview of the common characteristics of this model family, the analysis will concentrate on the EfficientNet-B7 variant, examining its strengths, internal structure, and the rationale behind its selection. For the implementation of the deepfake detection model, EfficientNet-B7 was selected by adopting a transfer learning approach using the pretrained version available in TorchVision, originally trained on ImageNet. Open-source code was used to load the model, and full fine-tuning was performed on the custom dataset created from the DFDC corpus, as previously described in Chapter 4. Subsequently, this chapter will explore alternative architectures used for comparative purposes, ResNet18 and Xception, will be introduced. The chapter concludes with a methodological discussion of the computational infrastructure adopted, highlighting the operational choices and the benefits associated with deploying a high-performance virtual machine (VM).

5.1 Introduction to the EfficientNet Family

For the model considered in this study, an architecture from the EfficientNet family was selected, specifically, the EfficientNet-B7 variant, due to its high discriminative capacity, architectural depth, and excellent balance between accuracy and computational scalability. This choice was not arbitrary, but driven by the need to balance accuracy, generalization, and structural robustness, all of which are essential in the automatic detection of manipulated audiovisual content.

EfficientNet is one of the most advanced and optimized families of convolutional neural networks developed in recent years, thanks to a method known as *compound scaling*, which enables the simultaneous and balanced scaling of depth, width, and input resolution. Unlike traditional models that grow unevenly, EfficientNet employs a controlled growth strategy that maintains a high performance-to-cost ratio. The family includes several variants, from B0 (lightweight and fast) to B7 (deepest and most accurate), all based on the same modular design. Below is a summary of the main configurations:

- **EfficientNet-B0:** The baseline model, designed for mobile devices and low-resource contexts. It contains approximately 5.3 million parameters and uses 224×224 pixel input images. Although highly efficient, it is not well-suited to complex tasks such as detecting subtle visual manipulations (*Keras, EfficientNet B0 to B7*).

- **EfficientNet-B1/B2/B3:** These represent a progression from the base model, with input sizes of 240×240, 260×260, and 300×300 pixels respectively, and up to roughly 12 million parameters for B3. They are suitable for general classification tasks but do not yet offer sufficient capacity to distinguish fine-grained alterations in facial features, as required by more sophisticated deepfake detection scenarios (*Keras, EfficientNet B0 to B7*).

- **EfficientNet-B4/B5:** These occupy the upper-mid range of the family. B4 accepts 380×380 pixel input with around 19 million parameters, while B5 increases to 456×456 with over 30 million parameters. These models offer strong performance in more demanding computer vision tasks, but the trade-off between precision and computational cost becomes increasingly significant (*Keras, EfficientNet B0 to B7*).

- **EfficientNet-B6:** One of the higher-end variants in the series, B6 uses 528×528 pixel inputs and contains approximately 43 million parameters. It is well-suited to scenarios where a solid balance is required between accuracy and available computational resources. Its architectural depth enables the learning of complex visual representations, making it appropriate for advanced image classification, object detection, and video analysis tasks. However, despite accuracy gains over B4 and B5, its training cost grows considerably, especially with larger batch sizes (*Keras, EfficientNet B0 to B7*).

- **EfficientNet-B7:** The most powerful variant in the series and the one selected for this project. In Keras, it is configured for 600×600 pixel input images, while in PyTorch, due to GPU compatibility and stability considerations, it is often used with 380×380 input resolution without significant performance degradation. The model features a deep and refined structure with approximately 66 million parameters and employs the Noisy Student Training strategy, a pre-training approach, detailed in the following paragraph, that combines weak supervision with extensive unlabeled data to improve generalization. This enables B7 to achieve superior robustness and better generalization when facing perturbed or manipulated content. Its characteristics make it particularly suitable for deepfake detection, where visual anomalies are often subtle and localized. EfficientNet-B7 is capable of capturing imperceptible details such as irregularities in facial boundaries, inconsistencies in eye structure, or texture variations in the skin (*Keras, EfficientNet B0 to B7*).

5.2 Why EfficientNet-B7 Was Selected

The selection of EfficientNet-B7 as the reference model for this study was not arbitrary, but rather the result of a careful evaluation that considered both theoretical and experimental aspects. In a complex task such as automatic deepfake video detection, where visual manipulations can be extremely subtle and localized, it is essential to rely on an architecture capable of capturing minimal differences between authentic and manipulated content, while ensuring robustness and generalization. EfficientNet-B7 represents the most advanced model in the EfficientNet family and stands out for the following reasons:

- **High accuracy on standard benchmarks:** EfficientNet-B7 has achieved state-of-the-art performance on datasets such as ImageNet [18], ranking among the top-performing standard CNN models.
- **High-resolution input (380×380 in PyTorch):** This allows the model to operate on very fine visual details, including facial imperfections, edge misalignments, and ocular artifacts.
- **Deep architecture with 66 million parameters:** Its structural depth makes it particularly suitable for high-complexity visual classification tasks.
- **Proven adoption in production systems and advanced pipelines:** Its widespread use in real-world applications reflects the architectural maturity and practical reliability of the model.

In addition, EfficientNet-B7 was pre-trained using the *Noisy Student Training* technique, a semi-supervised approach that leverages a large set of unlabeled data (approximately 300 million images from Google's internal JFT-300M [19] dataset) combined with pseudo-labels generated by a teacher model previously trained on ImageNet (1.2 million labeled images). The resulting student model benefits from a significantly enhanced generalization capability, enabling strong performance even on previously unseen content. This training strategy also enabled EfficientNet-B7 to reach a top-1 accuracy of 88.4%

[18]ImageNet is a widely used benchmark dataset in computer vision, containing over 1.2 million labeled images across 1,000 object categories, designed to support the development and evaluation of large-scale image classification models.

[19]JFT-300M is a large-scale internal dataset developed by Google, consisting of approximately 300 million weakly labeled images drawn from the web, used primarily for training and pre-training deep learning models at scale.

on ImageNet, outperforming larger models trained on billions of weakly labeled images. These characteristics make EfficientNet-B7 one of the most suitable architectures for rigorously and effectively addressing the challenge of deepfake detection in video analysis.

5.3 EfficientNet-B7 in Detail

As previously discussed, the guiding principle behind the entire EfficientNet family is *compound scaling*, which enables simultaneous and balanced scaling of three core dimensions: depth (number of layers), width (number of channels), and input resolution. The result is a more compact yet highly expressive network, capable of capturing increasingly abstract and detailed hierarchical representations. Figure 2 schematically illustrates the architecture of blocks 1 through 7 in EfficientNet-B7, highlighting the sequence and types of convolutional modules applied to the input data. The data flow through the network is described below:

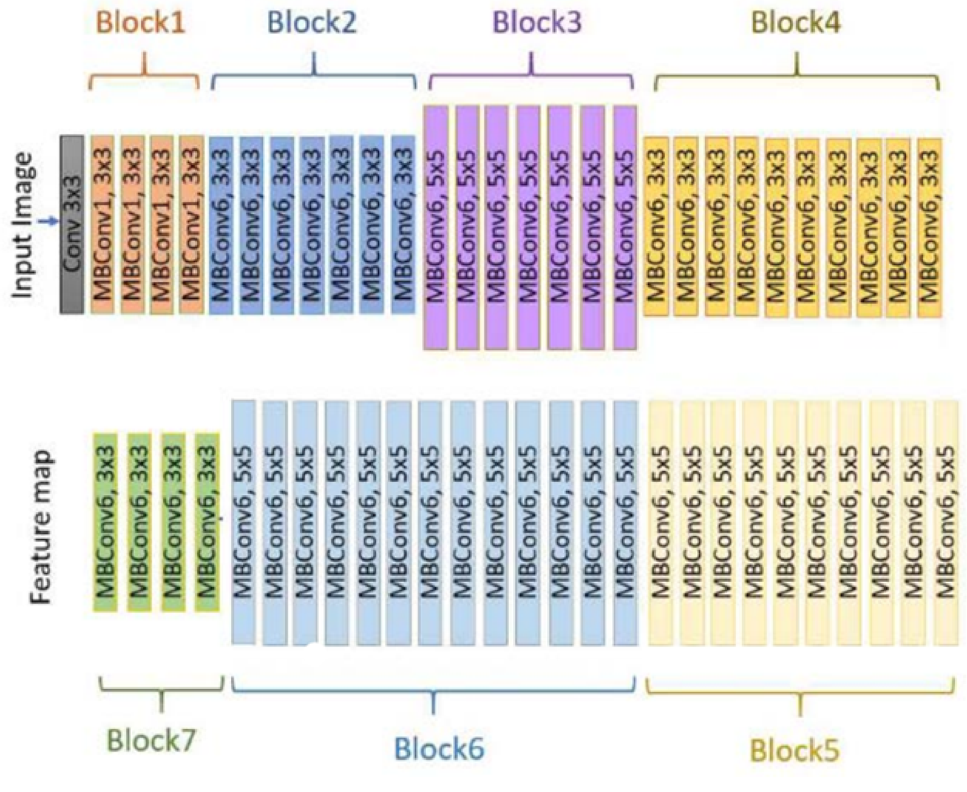


Figure 3: Architecture of the EfficientNet-B7 model: schematic overview of convolutional blocks. Source: Yahia Ibrahim, N., & Talaat, A. S. (2022)

- **Initial Block (Stem):** The input image (e.g., 380×380 pixels) is first processed by a standard 3×3 convolution that extracts low-level visual features such as edges and textures. This step prepares the data for the subsequent MBConv blocks [20].

- **Block 1:** Consists of MBConv1 modules with 3×3 kernels and reduced expansion (limited channel expansion). It performs the first nonlinear transformation of the data while maintaining low computational cost.
- **Block 2:** Introduces MBConv6 modules (with ×6 expansion) and 3×3 kernels, significantly increasing the number of filters and enhancing the model's depth and ability to capture more complex patterns.
- **Block 3:** Incorporates larger 5×5 kernels to detect broader spatial patterns. The sequence of MBConv6 modules increases the receptive field and abstraction capacity of the network.
- **Block 4:** Continues the previous strategy with additional MBConv6 3×3 modules, now operating on more compact feature maps with deeper channels.
- **Block 5:** Entirely composed of MBConv6 with 5×5 kernels. At this stage, the network processes highly abstract representations, approaching the final classification layer.
- **Block 6:** One of the deepest blocks, featuring numerous MBConv6 5×5 modules. Although the input has been significantly downsampled, the number of channels is high, enabling the model to discriminate between very similar classes.
- **Block 7:** The final convolutional block, composed of three MBConv6 modules with 3×3 kernels. This block refines the last visual features before the classification step.

Following the last convolutional block, the model applies a global average pooling operation, followed by a fully connected (dense) layer for final classification. EfficientNet-B7 processes the input image through a highly structured pipeline, where each block progressively refines the spatial features. Thanks to the extensive use of MBConv modules, depthwise convolutions, and squeeze-and-excitation (SE) attention mechanisms, the model achieves high performance with a number of parameters that, although considerable, is optimized relative to its architectural depth. With 66 million parameters, EfficientNet-B7 delivers extremely high predictive capacity, making it well-suited for demanding applications.

[20]MBConv (Mobile Inverted Bottleneck Convolution) is a lightweight block combining depthwise separable convolution, expansion-projection, and optional SE attention. It enables rich feature extraction with low computational cost and is a core component of EfficientNet's scalable architecture.

5.4 Architectural Comparison: Alternative Models to EfficientNet-B7

Although EfficientNet-B7 represents one of the most advanced solutions in terms of accuracy and discriminative power for image classification tasks, it is essential from an experimental standpoint to critically evaluate whether such a complex and computationally demanding model is truly necessary. In realistic deployment scenarios, where hardware resources may be limited, or systems are required to operate in real time or on edge devices, the use of an architecture comprising over 66 million parameters may prove prohibitive. For this reason, the present work includes a systematic comparison with alternative models, selected for their architectural simplicity and proven reliability in binary classification tasks. The aim of this comparison is not merely competitive but analytical: to explore whether simpler and less resource-intensive architectures, when well-designed, can deliver comparable or acceptable performance in deepfake detection. This analysis allows for an assessment of the trade-off between accuracy and computational cost, helping identify which architecture offers the best compromise based on the intended application context. Moreover, employing alternative models provides a broader perspective on the robustness of the task itself, investigating whether the visual information needed for classification can be effectively captured only by deep models, or also by lighter architectures. To this end, two models were selected: **ResNet18** and **Xception**, both recognized in the scientific community for their favorable balance between architectural complexity and generalization capacity.

- **ResNet18** is one of the most compact versions of the Residual Network family, introduced by *He et al. (2015)*, whose main innovation is the use of residual skip connections. These connections allow gradients to propagate more easily through the network, mitigating the degradation problem that affected earlier deep networks. ResNet18 contains approximately 11 million parameters, trains quickly, and exhibits stable behavior, making it an ideal baseline for evaluating more complex architectures. Despite its limited depth (18 layers), ResNet18 is capable of capturing robust visual patterns and is widely used in low-cost binary and multiclass classification tasks.

- **Xception**, short for "Extreme Inception," was proposed by *François Chollet (2017)* as an evolution of the Inception-v3 architecture [21]. Its primary contribution lies in

[21] Inception-v3 is a convolutional neural network architecture developed by Szegedy et al. (2016), known for its modular structure and factorized convolutions. It was designed to improve classification accuracy while reducing computational cost, and became one of the top-performing models on the ImageNet benchmark.

the extensive use of depthwise separable convolutions, which decompose the standard convolution into two stages: a depthwise (channel-wise) convolution followed by a 1×1 pointwise convolution. This approach dramatically reduces the number of parameters and operations compared to standard convolutions, while maintaining or even improving performance. Xception has a significantly deeper structure than ResNet18 and lies somewhere between the latter and EfficientNet-B7 in terms of complexity. It is known for strong performance in image classification and has been successfully applied in early studies on detecting synthetically generated content, including deepfakes.

The comparison among these three models was not based on identical training configurations but rather adapted to suit the specific characteristics and computational needs of each architecture. In particular, for ResNet18 and Xception, more lightweight and faster training setups were adopted compared to those used for EfficientNet-B7, in order to explore the possibility of achieving good performance with reduced training time and lower resource consumption. The following chapters provide a detailed explanation of the setup used for each model, including input preprocessing, training parameters, dataset management, and optimization techniques, thereby ensuring a transparent and comprehensive understanding of the experimental conditions.

The goal is not only to compare the resulting accuracy or F1-score, but also to evaluate model resilience to computational constraints, scalability across data availability, and generalization to previously unseen manipulated content. This comparison with ResNet18 and Xception helps determine whether lighter models can serve as valid alternatives to EfficientNet-B7, at least in certain application scenarios, contributing to a broader discussion on the optimization of efficiency in AI systems focused on digital content security.

5.5 Computing Environment: Use of a Virtual Machine (VM)

In the context of this work, a virtual machine (VM) was adopted as the primary computing environment to leverage the various advantages it offers in terms of flexibility, scalability, and resource management. A VM enables the creation of an isolated and controlled environment that can be replicated across different infrastructures. Moreover, using a VM allows precise configuration of the development environment, installing only the necessary components and optimizing the use of available resources. A mid-range virtual machine was deliberately adopted to promote the replicability of the experiments and the results obtained. The selected VM, provided through the Vast.ai [22] platform, is equipped with an NVIDIA RTX 4080 GPU based on the Ada Lovelace architecture, offering 48.3 TFLOPS of FP32 computational power, supported by 16 GB of video memory and 12.8 GB of effective usable CUDA memory. This configuration proved to be a key enabler for training EfficientNet-B7, which is known for its high demand in terms of VRAM and computational throughput.

The GPU is paired with an AMD EPYC 7502 processor, a 32-core / 64-thread CPU with PCIe 4.0 x16 support, capable of efficiently handling the parallel workload generated by the training and preprocessing pipelines. Additionally, the machine is equipped with a 600 GB NVMe SSD (WD_BLACK SN850) with read speeds exceeding 6,000 MB/s, an essential component for mitigating I/O bottlenecks, especially during the continuous reading of thousands of video frames in the image extraction process. The VM also provides a stable and adequate network connection for transferring large datasets, with upload speeds of 183 Mbps and download speeds of 328 Mbps, allowing efficient data and model exchanges with the local development environment. This computational infrastructure thus served as an enabling factor for the successful implementation of a complete deep learning workflow for training, testing, and comparing models designed for detecting manipulated content.

[22]Vast.ai is a cloud computing platform that provides on-demand access to virtual machines with GPU acceleration. It is designed to offer cost-effective and flexible infrastructure for machine learning, scientific computing, and rendering tasks, allowing users to rent computing power from a decentralized network of providers.

Chapter 6

Experimental Pipeline and Model Implementation

This chapter presents a comprehensive overview of the experimental pipeline developed for deepfake detection, encompassing all stages from data preprocessing to model training and architectural implementation. The primary objective is to detail the engineering and methodological choices made to construct a robust and efficient deep learning workflow, without delving into the performance results or comparative analyses, which are addressed in subsequent chapters.All implementation activities were carried out using the PyTorch framework, selected for its flexibility, modularity, and wide adoption in the research community. The chapter begins by describing the dataset preparation process, including frame extraction, face detection, and normalization techniques. It then outlines the construction of training, validation, and test splits, followed by the configuration of data loaders and preprocessing pipelines. Subsequent sections focus on the implementation of the selected models, detailing the hyperparameters, loss functions, and optimization strategies employed. Particular attention is given to EfficientNet-B7, the main architecture under study, as well as alternative models such as ResNet18 and Xception.

6.1 Preprocessing and Dataset Preparation

The effectiveness of a deep learning model for deepfake detection is closely tied to the quality and consistency of the dataset used for training. In this project, the preprocessing phase was designed as a series of deterministic and repeatable operations aimed at generating standardized facial images that accurately represent the semantic structure of the videos. The entire process was optimized to ensure consistency across the training, validation, and test sets, while minimizing noise and variability introduced by irrelevant artifacts.

6.1.1 Metadata Parsing and Binary Label Assignment

The initial step involved reading and analyzing the `metadata.json` file provided with the DFDC (DeepFake Detection Challenge) dataset. This file maps each video to a set of information, including the type of manipulation and the corresponding binary label

(0 for real content, 1 for fake content). Metadata parsing was implemented in Python using the `json` module, producing a dictionary structure that enabled the mapping of each video to its respective class. The label was then persistently associated with the images extracted from the video frames, ensuring full traceability between raw data and preprocessed model inputs.

This step was essential for constructing a coherent and verifiable foundation for the supervised dataset, upon which a CSV file was generated for the subsequent cross-validation phase. The CSV file, created after parsing and label assignment, represents the core structure of the supervised dataset. For each image derived from a video frame, a row was recorded containing: the image filename, the associated binary label (0 = real, 1 = fake), and the assigned fold number (from 1 to 5). This structure supports the 5-fold cross-validation strategy, which divides the dataset into five partitions: in each iteration, four folds are used for training and one for validation. The model is thus validated on each partition at least once, ensuring a robust and reliable performance estimate. The CSV file is later read by the dataloader to dynamically load images, distinguishing between training and validation sets according to the selected fold.

6.1.2 Uniform Extraction of 32 Frames per Video (Frame-Based Approach) and Face Detection with MTCNN

Subsequently, 32 frames were uniformly extracted from each video, distributed across the entire duration of the footage. This method allows for the preservation of temporal diversity in facial expressions and movements, providing comprehensive coverage of the visual content. Frame extraction was performed using video-processing tools (OpenCV), configured to skip frames at equal intervals regardless of the total video length, ensuring a constant and representative sampling rate. This strategy proved particularly effective in capturing facial manipulations that occur at specific moments, avoiding biases introduced by randomly selected or initial frames.

Each of the 32 extracted frames was subjected to face detection using the MTCNN (Multi-task Cascaded Convolutional Networks) architecture, implemented through the `facenet-pytorch` library. MTCNN is one of the most reliable neural networks for face detection in uncontrolled environments, due to its ability to perform bounding box prediction and facial landmark localization simultaneously. For each frame, the most prominent face was automatically identified, discarding any frames where no face could be detected. The selection was limited to the face with the highest detection probability, in order to avoid ambiguity in videos featuring multiple subjects. Problematic cases, such as missing faces, decoding errors, or corrupted frames, were handled through exception management and logged in a dedicated file for subsequent analysis.

6.1.3 Cropping with Contextual Margin and Resizing to 380×380 px

Once the face was identified, the image was cropped with a contextual margin of 30% relative to the original bounding box. This extension allowed the inclusion of adjacent areas such as hair, parts of the neck, and the immediate background, contributing to the preservation of visual cues useful for the model, in line with recommendations from the literature. After cropping, the images were resized to 380×380 pixels, an optimal resolution for input into the EfficientNet-B7 model. This size represents a trade-off between representational capacity and computational efficiency: it is lower than the input size used by more redundant architectures, yet sufficient to capture subtle visual artifacts typical of deepfakes (e.g., blurred contours, skin inconsistencies, etc.).

6.1.4 Saving Cropped Faces into /real and /fake Directories

The cropped facial images were saved in `.png` format into two separate directories (`/real` and `/fake`), according to the binary label obtained from the metadata. Each image was uniquely named to include references to the source video and the corresponding frame number, ensuring full traceability throughout the entire preprocessing pipeline. In total, more than 330,000 preprocessed images were generated. This hierarchical structure enabled efficient data management in the subsequent stages of CSV generation, training, and testing. It also simplified dataset organization for alternative model implementations such as Xception and ResNet18.

6.1.5 Preprocessing Parallelization with multiprocessing

Preprocessing the entire video dataset required significant computational resources. To reduce processing time, a parallelization strategy was implemented using Python's `multiprocessing` module, leveraging all available CPU cores and, where possible, asynchronous GPU capabilities. Videos were divided into batches and assigned to separate processes for frame extraction, face detection, and image saving. This approach reduced total processing time from several days to just a few hours, making preprocessing scalable even for large datasets (over 100 GB of video data).

6.1.6 Logging, Error Handling, and Reproducibility

Throughout all preprocessing stages, specific measures were adopted to ensure reproducibility and traceability. Any errors or exceptions encountered (e.g., unreadable videos, frames without detectable faces, I/O issues) were recorded in a `log.txt` file, including a timestamp, filename, and error type. This logging strategy not only allowed for continuous monitoring of the pipeline's reliability but also enabled the identification of recurring issues associated with specific video subcategories. Finally, to ensure full reproducibility of the experiments, a global random seed (e.g., 42) was fixed, and all configuration parameters and relative paths were saved in a dedicated `.yaml` configuration script[23]. This made the entire workflow replicable across different machines and hardware settings.

6.2 Construction and Organization of the Supervised Dataset

The effective organization of the dataset represents a crucial step in ensuring consistency and reproducibility throughout the training process. After completing the preprocessing operations and obtaining a set of labeled images (i.e., faces extracted from video frames, normalized and stored), it was necessary to construct a logical and functional structure to enable cross-validation, manage class imbalance, and clearly separate the training, validation, and test phases. This structure was formalized through the generation of a CSV file, which allowed fine-grained control over the behavior of the DataLoader during all stages of the experimental pipeline.

6.2.1 Generation of the CSV File with Labels and Paths

As discussed in the previous sections, the CSV file represents the core of the supervised dataset: for each preprocessed image (corresponding to a face extracted from a video frame), a row was generated containing at least three essential pieces of information:

1. **File path:** the absolute or relative path of the image, stored in either the `/real` or `/fake` directory;
2. **Binary label:** obtained from the `metadata.json` file, where 0 denotes real content and 1 denotes manipulated (deepfake) content;
3. **Fold number:** an integer value between 1 and 5, randomly assigned to implement fold-based cross-validation.

[23]YAML (Yet Another Markup Language) is a human-readable data serialization format commonly used for configuration files. It allows for structured, hierarchical parameter definitions using indentation-based syntax. Its simplicity and readability make it ideal for specifying experimental setups in machine learning pipelines.

This tabular structure was entirely constructed in Python using libraries such as `pandas` for DataFrame management and `os` for iterating over image paths. The resulting file, saved in `.csv` format, was then employed by the PyTorch `Dataset` module to dynamically load data batches, thereby facilitating the separation between training and validation sets at each stage of the learning process. Additionally, for experimental variants (e.g., datasets reduced to 5%, 10%, 15%, and 20% of the original data), dedicated CSV files were generated (e.g., a CSV file for the 20% training subset), maintaining the same structure but containing progressively smaller subsets of the original dataset.

6.2.2 Assignment to 5 Folds for Cross-Validation

The K-Fold Cross-Validation technique, adopted here with $k = 5$, allows for a robust evaluation of model performance that is independent of the initial data split. The assignment to the five folds was performed through random sampling, using a fixed seed (`random seed = 42`) to ensure reproducibility of the experiments. Each image was thus assigned to one of the five folds (from 1 to 5) in a stratified manner, maintaining approximately the same class distribution (real and fake) within each fold. This approach prevented any single fold from being unbalanced, which could have compromised the validity of the evaluation. The choice of using five folds represents a compromise between statistical accuracy and computational efficiency. A smaller k value (such as 2 or 3) would lead to unstable estimates, highly sensitive to the specific data split. Conversely, higher values (such as 10 or leave-one-out) would significantly increase training time, as the model would need to be trained many more times. In the case of complex architectures like EfficientNet-B7, for which a single training cycle can require several hours even on advanced GPUs, an intermediate value like $k = 5$ is widely accepted in the scientific literature and represents an optimal choice in terms of balancing accuracy and computational sustainability.

The assignment of images to the folds was stratified, ensuring approximately the same proportion of real and fake content within each subset. This strategy is particularly important in the presence of class imbalance, as in the DFDC dataset, where manipulated videos account for about 90% of the total. A purely random distribution without stratification could have undermined the quality of the validation, resulting in folds with insufficient representation of the minority class. During training, for each iteration, four folds were used as the training set and one as the validation set, in rotation. This mechanism ensured that each image was validated exactly once and used for training in the remaining iterations, thereby maximizing the informational efficiency of the dataset.

6.2.3 Use of Folds for Training/Test Separation

Beyond their direct use in cross-validation, the fold structure was repurposed to separate data for training and testing in a coherent and controlled manner. Although the two main test sets used in the project (one of 10 GB and one of 50 GB) were entirely separate from the training dataset, the folds were also used internally to distinguish data during "out-of-fold" inference simulations [24]. During the training of the EfficientNet-B7 model, for each fold, the model that achieved the best validation loss across 10 epochs was saved. Subsequently, each model was tested not only on unseen videos (contained in the external test sets), but also on data from folds different from the one it was trained on, in order to obtain more stable performance aggregation and reduce dependency on specific partitions. In practice, the folds enabled the construction of a modular system in which:

- Each image is clearly labeled and traceable;
- Each data batch can be filtered depending on the phase (training or validation);
- Inference results can later be aggregated for video-level evaluation through frame-wise heuristics.

6.2.4 Handling Class Imbalance via pos_weight

One of the most critical aspects in constructing the dataset was the imbalance between classes: in both real and synthetic datasets used in this study, the `fake` class accounts for approximately 90% of the total videos, while the `real` class comprises only around 10%. This imbalance, deliberately imposed to maximize the model's discriminative capacity, presents significant challenges for supervised training, particularly in the accurate detection of the minority class (real videos). To mitigate this imbalance, the `pos_weight` parameter of PyTorch's `BCEWithLogitsLoss` function was employed. This parameter allows adjustment of the internal weighting of the loss function by assigning greater importance to errors made on the minority class. The `pos_weight` value was calculated as the ratio between the total number of class 1 (fake) and class 0 (real) examples, and it was dynamically updated for each fold. For instance, if a fold contained 1,000 real examples and 9,000 fake ones, the `pos_weight` was set to 9.0. This means that an error on a real (rarer) sample was penalized more heavily than an error on a fake one. Through this mechanism, the model was explicitly encouraged not to disregard the less frequent

[24]Out-of-fold (OOF) inference refers to generating predictions on a subset of the data using a model that has not been trained on that specific subset. It is commonly used in K-Fold Cross-Validation to ensure unbiased performance estimation and to prevent information leakage.

class, thereby improving critical evaluation metrics such as recall, F1 score, and weighted precision.

6.3 Implementation of a Custom DataLoader

To ensure efficient and modular image loading during training, validation, and testing phases, a custom class named `DeepFakeDataset` was implemented, following the standard structure defined by PyTorch. The preprocessed image dataset (stored in `/real` and `/fake`) was made accessible via a CSV file containing image paths, binary labels, and fold numbers for cross-validation. This approach enabled the creation of a highly flexible loading pipeline, capable of dynamically filtering by fold, conditionally applying transformations, optimizing GPU throughput, and integrating with distributed batch training strategies.

6.3.1 Definition of the DeepFakeDataset Class

The `DeepFakeDataset` class was implemented in Python by extending the abstract interface `torch.utils.data.Dataset`[25]. Its main purpose is to provide modular and controlled access to the supervised dataset, by reading directly from the CSV file containing preprocessed image paths, associated binary labels, and fold assignments for cross-validation. The dataset is initialized by loading the CSV file using the `pandas` library, from which the relevant columns (image paths, labels, and fold numbers) are extracted. For each requested sample, the corresponding image is loaded into memory in RGB format[26] using either `PIL.Image`[27] or `cv2.imread` (properly converted), and then passed through a transformation pipeline defined externally via `torchvision.transforms`.

This transformation pipeline may include data augmentation[28], resizing, and normalization. Once the transformations are applied, the image is converted into a PyTorch tensor and paired with the corresponding label (also converted into a tensor), returning a tuple[29] `(input_tensor, label)` compatible with training, validation, or testing phases. The class structure was designed to be highly flexible. Upon instantiation, it is possible to specify different input parameters, including the desired split type (train, val, or test),

[25]`torch.utils.data.Dataset` is a base class provided by PyTorch for defining custom datasets. It allows developers to implement methods to support flexible and efficient data loading.

[26]RGB is a standard color model where images are represented using red, green, and blue color channels.

[27]`PIL.Image` is a module from the Python Imaging Library (Pillow) used for opening, manipulating, and converting image files in various formats.

[28]Data augmentation refers to a set of techniques used to artificially increase the diversity of training data by applying transformations such as rotation, flipping, scaling, or color adjustment.

[29]A tuple is a fixed-size ordered collection of elements. In this context, it contains the input tensor and its corresponding label.

the fold to exclude for out-of-fold logic, and whether to enable data augmentation. This modular approach enabled the construction of dynamic training and evaluation pipelines while maintaining full consistency across different experimental configurations.

6.3.2 Integration with PyTorch DataLoader

The `DeepFakeDataset` class was integrated into PyTorch's `DataLoader` module to enable efficient data loading during training, validation, and testing phases. The `DataLoader` plays a central role in managing data flow in deep learning environments, as it allows for automatic batching of the dataset, optimized loading latency, and proper memory handling between CPU and GPU. In this project, the `batch_size`[30] was set to 8 when training with EfficientNet-B7 to accommodate GPU memory constraints and the large image resolution (380×380 pixels). For lighter models such as ResNet18, larger batch sizes, up to 16, were feasible, benefiting from the lower memory footprint of the architecture. During the training phase, the presentation order of the data was randomized using automatic shuffling to prevent the model from learning fixed sequences or undesirable patterns related to data ordering.

The `DataLoader` was also configured to perform data loading in parallel using the `num_workers` parameter, which enables the use of multiple subprocesses for *prefetching*[31] and batch preparation. This significantly reduced I/O latency, especially when images had to be read from disk or converted into tensors. Finally, asynchronous data transfer to the GPU was enabled by setting `pin_memory=True`[32], which improves the data transfer speed between RAM and GPU memory through pre-allocated pinned memory. This integration between the custom dataset and the `DataLoader` ensured high throughput during training, even with complex models and high-resolution images, thus supporting both the scalability of the system and the reproducibility of experimental results.

6.3.3 Data Augmentation During Training

During the training phase, a data augmentation module was implemented to increase dataset variability and reduce the risk of overfitting, without altering the semantic properties of the images. The transformations were applied only during training (not during validation or testing) and included the following:

[30] `batch_size` refers to the number of samples processed simultaneously during a single forward/backward pass of the model. Smaller values reduce memory usage, while larger values may improve training speed.

[31] Prefetching is the technique of loading or preparing data in advance while the model processes the current batch, thereby minimizing idle time and improving efficiency.

[32] `pin_memory` is a PyTorch option that pre-allocates page-locked memory in RAM to speed up data transfer between CPU and GPU.

- **Random Horizontal Flip:** horizontal flipping with a probability of 0.5, useful for simulating facial symmetry variations;
- **Color Jitter:** random adjustments of brightness, contrast, and saturation to mimic changes in lighting conditions;
- **Random Crop / Resize:** random cropping followed by resizing, useful for introducing small geometric variations.

These transformations were implemented using the `torchvision.transforms.Compose`[33] module and controlled via a boolean flag passed to the dataset. Their application improved the model's generalization capability while preserving visual consistency with the original image. It is worth noting that, to ensure fairness in model comparison (EfficientNet-B7, Xception, ResNet18), data augmentation was deliberately disabled in some experimental configurations, as described in the section on experimental variations (Section 6.6).

6.3.4 Transformation and Normalization Pipeline

To ensure compatibility with models pre-trained on ImageNet and to maintain stability during training and inference phases, all images were processed through a standardized transformation pipeline, applied uniformly to both training and validation/test data.

The first operation in the pipeline involved resizing the images according to the architecture in use: for EfficientNet-B7, images were scaled to 380×380 pixels; for ResNet18, to 224×224; and for Xception, to 299×299 pixels, the official input resolution used during pretraining on ImageNet. This design choice is based on the specific input size requirements dictated by each model's pretraining configuration. Next, the images were converted into PyTorch tensors using the `transforms.ToTensor()`[34], which performs initial normalization of pixel values. This conversion is essential for the proper functioning of convolutional operations, which require continuous numerical inputs within a normalized range.

Finally, channel-wise RGB normalization was applied using the standardized mean and standard deviation values from ImageNet. These parameters reflect the statistical distribution of pixel intensities in ImageNet data and are critical for producing balanced activations in the initial convolutional layers. Any deviation from these statistics can hinder

[33]`torchvision.transforms.Compose` is a PyTorch utility that allows multiple image transformations to be chained together and applied sequentially.

[34]`transforms.ToTensor()` is a PyTorch function that converts a PIL Image or NumPy array into a normalized tensor with pixel values scaled between 0 and 1. This format is required for tensor operations in neural networks.

learning efficiency, slow convergence, or introduce undesired behavior in the network, especially during the early stages of training. Strict adherence to this preprocessing pipeline ensured uniformity across all input data, enabling effective reuse of pre-trained weights and improving the model's ability to generalize to previously unseen videos, while reducing the risk of overfitting to specific visual characteristics.

6.4 Training Strategy

Model training represents the central phase of the entire experimental pipeline. Architectural choices, optimization strategies, epoch management, and evaluation metrics all contribute to shaping a process aimed at building a robust, generalizable classifier that is resilient to class imbalance. The training strategy described in this section is structured into six main components, each outlining a specific step in the K-Fold-based training process.

6.4.1 Adapting the Classifier for Binary Classification

As outlined in previous chapters, the model selected for the main experiment is EfficientNet -B7, one of the most advanced architectures in terms of balancing accuracy and computational complexity. EfficientNet introduces a paradigm known as *compound scaling*, which enables simultaneous and harmonious scaling of depth, width, and input resolution, delivering high performance with fewer parameters than traditional convolutional networks. EfficientNet-B7, in its original version, is designed for multi-class classification tasks on ImageNet, with a fully connected output layer of size 1000. To adapt it for the task of deepfake detection, formulated as a binary classification problem (real vs. fake), the final classifier was replaced with a single neuron producing a continuous output.

From a theoretical standpoint, this classifier modification is not only valid but also exemplifies a key principle of transfer learning: the adaptation of pre-trained models to new tasks, often involving a different number of output classes. The deep convolutional layers, which represent the most informative component of the model, are preserved, while the task-specific output layer is replaced to reflect the new binary structure. In this case, adding a single output neuron without an explicit sigmoid activation enables seamless integration with the `BCEWithLogitsLoss` function, which internally applies the sigmoid transformation to yield probabilistic outputs. This approach improves numerical stability during training and enables a clear and interpretable decision threshold (0.5), facilitating the aggregation of predictions during the video-level inference phase.

6.4.2 Weighted Binary Cross-Entropy Loss (BCEWithLogitsLoss)

The loss function adopted for training was Binary Cross-Entropy with Logits (`BCEWith-LogitsLoss`), specifically designed for binary classification tasks. This loss measures the distance between the model's continuous output (prior to the sigmoid activation) and the binary label, and is used to update the network's weights during optimization. A critical aspect of the training phase was the dataset imbalance, with the `fake` class accounting for approximately 90% of the samples. To address this asymmetry, the `pos_weight` parameter was used, which increases the penalty associated with errors on the minority class. The value of `pos_weight` was dynamically calculated for each fold as the ratio between the number of `fake` and `real` samples, ensuring that the network learned not to ignore false negatives while still distinguishing fake content correctly. This choice aligns with best practices for rare event classification and allows the model to maintain high sensitivity to the `real` class, preventing it from being overlooked due to class imbalance.

6.4.3 Optimization with Adam and Learning Rate Configuration

Parameter optimization was performed using the Adam algorithm (Adaptive Moment Estimation), known for its efficiency in adapting the learning rate of each weight based on estimates of the first and second moments of the gradients. Adam is particularly effective in scenarios involving moderately sized datasets and complex architectures, as it provides good stability and rapid convergence even in the presence of noise. The initial learning rate was set to 1×10^{-4}, a value considered conservative yet stable for pre-trained networks. This choice aimed to avoid significantly disrupting the pre-initialized weights during the early epochs, preserving the feature representations learned from ImageNet. In later configurations, a dynamic learning rate adjustment mechanism (scheduler) was implemented to automatically reduce the learning rate if the validation loss did not improve after a predefined number of consecutive epochs.

6.4.4 Training on 5 Folds and Best Model Saving

Training was conducted following the 5-Fold Cross-Validation scheme described in Section 6.2.2. In each iteration, the model was trained on four folds of the dataset and validated on the remaining one. This mechanism enabled fair and generalizable evaluation of the model's predictive capability, ensuring that each image was used exactly once for validation and four times for training. Throughout each training cycle, the validation loss was monitored epoch by epoch. The model achieving the lowest validation loss was saved as the *best model* for that fold and was subsequently used during the testing phase for inference on unseen data. This approach allowed the construction of an ensemble of independently validated models, improving prediction reliability and reducing the risk of overfitting to specific data subsets.

6.4.5 Overfitting Control and Epoch Management

To monitor and mitigate overfitting, the training and validation loss curves were continuously recorded during each epoch. Training was performed for a maximum of 10 epochs per fold, based on a methodological compromise between learning depth and computational sustainability. This value was selected to provide the model with sufficient capacity to learn complex representations without excessively prolonging the training cycle or overloading computational resources. It is important to note that, as discussed in Section 6.6.2, a systematic comparison was conducted between models trained for 10 epochs and those trained for 5 epochs. This comparison aimed to assess the impact of training duration on final performance, offering broader insights into the role of epoch count in balancing predictive accuracy and computational efficiency.

In cases where a significant divergence between the training and validation curves was observed, a soft early stopping mechanism was employed, saving the model corresponding to the lowest validation loss and avoiding further training of overfitted models. Additionally, the use of data augmentation techniques (described in Section 6.3.3) increased the variability of input samples, reducing the likelihood that the model would learn superficial patterns or dataset-specific artifacts. These precautions, together with the implicit regularization provided by the Adam optimizer and standardized data normalization, contributed to a robust and reproducible training process, enabling the models to perform effectively even on previously unseen videos, as demonstrated during the testing phase.

6.5 Testing and Inference

The testing and inference phase represents the final and most critical stage of the experimental process, during which the model's true ability to distinguish between authentic and manipulated content is evaluated on data never seen during training. The objective is to start from raw video input and arrive at the assignment of a binary label along with an associated probability score. The testing infrastructure was designed to ensure reproducibility, modularity, and consistency with the entire preprocessing and training pipeline.

6.5.1 Frame-Level Inference with Model in Evaluation Mode

Inference was performed at the frame level, in line with the frame-based approach adopted during the training phase. Each video was previously split into 32 equally spaced frames, from which normalized faces were extracted and saved into the testing directory. During inference, the pre-trained model is loaded in `eval` mode, a configuration that disables stochastic operations used during training (such as dropout and batch normalization in update mode), thereby ensuring deterministic predictions. For each frame, a *forward pass*[35] is executed through the model, producing a continuous output, which is then transformed into a probability using the sigmoid function. This value represents the estimated probability that the given frame was artificially generated.

6.5.2 Heuristic Aggregation of Frame-Level Predictions

Since the ultimate goal is to classify the entire video rather than individual frames, it was necessary to introduce a mechanism for aggregating frame-level predictions. For this purpose, a simple mean heuristic was implemented, consisting of computing the arithmetic average of the predicted probabilities across all frames of a video. The decision to use the mean as an aggregation strategy was based on two main considerations: first, it offers a robust compromise between sensitivity and predictive stability, avoiding situations in which a single anomalous frame determines the overall outcome; second, it is an interpretable and computationally efficient method that can be easily integrated into a *real-time pipeline*[36].

6.5.3 Final Label Assignment per Video

Once the average frame probability is computed, the final binary label for the video is determined using a fixed decision threshold of 0.5. If the average probability exceeds this value, the video is classified as `fake`; otherwise, it is considered `real`. Although this threshold can be adjusted to optimize for precision or recall in specific application scenarios, using the neutral value of 0.5 represents a fair and balanced choice for experimental evaluation, consistent with standard benchmark metrics. This decision strategy, combined with the aforementioned heuristic aggregation, enables consistent, reproducible, and easily interpretable evaluation of model performance at the video level.

[35] A *forward pass* refers to the process of feeding input data through the model to obtain predictions, without performing any weight updates.

[36] A *real-time pipeline* refers to a processing system designed to handle data and produce results with minimal latency, suitable for time-sensitive applications such as video streaming or live detection.

6.5.4 Collection of y_true, y_pred, and y_prob for Evaluation

For each video processed during the inference phase, three key values were recorded to support the subsequent evaluation of model performance. First, the ground truth label of the video, denoted as `y_true`, was obtained directly from the `metadata.json` file, which specifies whether the content is authentic or manipulated. This was paired with `y_pred`, the binary label predicted by the model (0 for real, 1 for fake), derived by applying a fixed threshold to the aggregated probability. Lastly, the continuous value `y_prob`, corresponding to the average frame-level probability before thresholding, was also stored.

The simultaneous availability of the `y_pred` (binary labels) and `y_prob` (continuous probabilities) vectors allows for the computation of different evaluation metrics, each with a distinct sensitivity to the model's confidence level. Specifically, while discrete metrics such as precision, recall, F1-score, or accuracy consider only the correctness of the predicted class, continuous metrics like log loss or AUC-ROC take into account the predicted probability itself, penalizing high-confidence errors more severely. For example, predicting a probability of 0.9 when the actual label is real constitutes a more serious mistake (and is penalized accordingly) than an incorrect prediction with a confidence of 0.6. This enables a more nuanced assessment of model performance, which is especially useful in scenarios where the classification threshold may be adjusted based on operational risk.

6.5.5 Prediction on Test Set 1 (10 GB) and Test Set 2 (50 GB)

The entire inference pipeline was applied to two distinct test sets, designed to evaluate the model's generalization capability on content not seen during the training phase.

The first, referred to as *Test Set 1*, has a size of approximately 10 GB and consists of videos drawn from the same distribution used for training, although entirely excluded from the training process. This set was used as an intermediate evaluation tool due to its manageable size and structural consistency with the data processed during model learning.

The second set, referred to as *Test Set 2*, has a size of approximately 50 GB and was designed to test the model's robustness under more heterogeneous and realistic conditions. This set includes a greater variety of faces, visual conditions, video compression levels, and recording quality. Its class distribution is unbalanced, with 85% of the content labeled as `fake` and 15% as `real`. Although this ratio is closer to real-world scenarios than a perfectly balanced set, it still does not reflect the actual prevalence of manipulated content in operational contexts, where deepfakes typically represent only a small minority. To better simulate realistic conditions in which deepfakes occur with an estimated prevalence of around 5%, as discussed in the literature and analyzed in Chapter 4, corrective evaluation metrics were introduced based on an assumed real-world distribution.

Among these, the *Weighted Precision* metric was adopted. This metric appropriately adjusts the impact of false positives and false negatives, offering a performance assessment that accounts not only for the internal distribution of the test set but also for operational risk in real-world deployment scenarios. In both cases, inference was performed in a fully automated manner, maintaining the same preprocessing parameters, transformations, and dataloader structure used during training.

6.6 Experimental Variations

In addition to the main experiment conducted on the full available dataset, a series of secondary experiments were designed and executed with the aim of analyzing the model's behavior along three fundamental dimensions: the amount of training data, the number of training epochs, and robustness to randomness. These variations made it possible to evaluate not only the absolute effectiveness of the adopted architecture, but also its sensitivity to different and more constrained operational conditions.

6.6.1 Progressive Subsets

To evaluate the impact of data availability on the learning process, five progressive subsets of the full training dataset were created, corresponding to 5%, 10%, 15%, 20%, and 100% of the total training images. Each subset was selected in a stratified manner, preserving the original class distribution (fake and real) and ensuring consistency across fold assignments. This procedure allowed the model's behavior to be measured under conditions of data scarcity, verifying whether the network could still achieve satisfactory performance in low-information scenarios.

6.6.2 Metadata Generation and Image Management

For each generated subset, a dedicated copy of the `.csv` metadata file was created, containing only the paths and labels of the images selected for that specific subset size. The fold assignments remained unchanged from the full dataset, allowing for direct comparison between models trained on different subsets. Folder structure and image path references were also kept consistent, ensuring compatibility with the existing dataloader and transformation pipeline. This modular management enabled automated and reproducible training for each data volume level.

6.6.3 Objective: Identifying the Data-Performance Trade-Off

The main goal of this analysis was to identify an optimal balance point between the amount of training data and the quality of the model's predictions. The evaluation of models trained on progressive subsets made it possible to observe how performance improved as data availability increased, while also identifying potential thresholds beyond which additional data yielded only marginal benefits relative to the added computational cost. The results of this analysis are presented in Chapter 7 and serve as practical guidance for defining more efficient training strategies in resource-constrained environments.

6.6.4 Epoch Variation: Comparison Between 5 and 10 Epochs

In addition to training data volume, the effect of training duration on model performance was also analyzed. Specifically, two configurations were compared: one with 10 epochs (the standard setting), and one with only 5 training epochs. Both configurations were replicated across each of the previously defined subsets, enabling a cross-comparison between learning depth and data quantity. The goal was to determine whether a reduced number of epochs could still yield acceptable performance while significantly decreasing training time. This type of comparison is particularly useful in real-world scenarios where computational resources are limited or where rapid model deployment is a critical requirement.

Reducing the number of epochs can help mitigate the risk of overfitting, especially when working with small datasets. However, too few iterations may hinder model convergence, leading to an underestimation of its full potential. As discussed in the following sections, the results showed that in some cases, the 5-epoch configuration produced performance comparable to that of the 10-epoch setting, particularly on the smaller subsets, while halving computational cost. These findings raise important considerations about dynamically adapting the number of training epochs based on data volume and task complexity.

6.6.5 Robustness and Randomness: Replicability Analysis Using Multiple Seeds

To assess the statistical robustness of the results, a replicability experiment was conducted using two different random seeds. Since many training operations, such as data shuffling, weight initialization, and batch ordering, depend on pseudorandom number generators, it is essential to verify that the observed performance is not the result of a lucky configuration, but is instead stable and repeatable. In each replication, training was carried out under identical configurations, except for the random seed. Performance metrics were then compared to measure the variance and possible sensitivity of the model to stochastic components.

The results, presented in the following sections, demonstrate that while minor variations in performance occurred between runs, the model's overall behavior remained consistent. This suggests that the architecture and pipeline setup are sufficiently robust to absorb the fluctuations introduced by initialization randomness. These findings underscore the importance of fixing the random seed in scientific experiments and of repeating evaluations to ensure the reliability of results, particularly in contexts where small differences may significantly impact operational decisions or model comparisons.

6.7 Comparative Experiments with Alternative Architectures

As previously outlined, in order to evaluate the effectiveness and flexibility of the experimental pipeline, two alternative convolutional neural network architectures, ResNet18 and Xception, were considered in addition to the primary model, EfficientNet-B7. The objective of this phase was not to replace the reference model, but rather to explore lighter and less computationally demanding architectural solutions. The aim was to assess their structural compatibility with the implemented data pipeline and to evaluate their potential impact on training and inference efficiency. The experiments were conducted using the same methodology as for EfficientNet-B7, in order to enable a controlled comparison based solely on architectural differences.

6.7.1 Training of ResNet18

ResNet18 is one of the lightest variants of the ResNet (Residual Networks) family, known for introducing residual blocks that help mitigate the degradation problem often encountered in deep networks. The decision to include ResNet18 among the tested models was motivated by its compact architecture, consisting of only 18 layers, which allows for faster training even in environments with limited computational resources.

To ensure compatibility with the pretrained weights from ImageNet, input images were resized to 224×224 pixels, in accordance with the original dimensional specifications of the model. The final classifier was also modified by replacing the fully connected layer with 1000 neurons with a single neuron producing a continuous output, in line with the adaptation performed for EfficientNet-B7. The loss function used was again

`BCEWithLogitsLoss`, and the training procedure was kept identical, including standardized data transformations and 5-fold cross-validation.

6.7.2 Training of Xception

The second alternative model evaluated was Xception (Extreme Inception), a deep convolutional network based on depthwise separable convolutions and designed as an evolution of the Inception-v3 architecture. Xception offers a good trade-off between representational power and computational efficiency, owing to the replacement of standard convolutions with depthwise separable ones, which significantly reduce the number of parameters while maintaining high performance. In this case as well, to ensure compatibility with the pretrained ImageNet weights, input images were resized to 299×299 pixels, as required by the model's original design. The final classifier was adapted for the binary classification task, and the training scheme was preserved, featuring fold-based stratification, data augmentation, and normalization consistent with ImageNet statistics.

6.7.3 Lightweight Architectures: Motivations and Trade-Offs

The decision to experiment with lightweight architectures was motivated by several operational considerations. In particular, adopting models with a reduced number of parameters can be advantageous in contexts where computational resources are limited, or where the model must run in real-time or on edge devices such as smartphones, smart cameras, or embedded systems. ResNet18 and Xception are emblematic examples of this category, although they embody different architectural approaches: the former is a network deeply optimized for stability, while the latter emphasizes computational efficiency. However, the use of shallower architectures inevitably involves trade-offs in terms of representational capacity. Smaller models may exhibit lower sensitivity to complex visual artifacts or reduced generalization capability in heterogeneous scenarios. For this reason, the comparison was carefully planned by keeping all other components of the pipeline as constant as possible, in order to isolate the purely architectural effects on performance.

6.7.4 Consistency of Transformations and Adapted Configurations

To ensure methodological consistency across models, each architecture was tested within the same experimental infrastructure. This required careful adaptation of transformations, particularly during image preprocessing, to comply with the dimensional and statistical requirements of each backbone.

Basic transformations, such as resizing, normalization, and tensor conversion, were kept as equivalent as possible, with adjustments made only to strictly necessary parameters like input resolution. The configurations for learning rate, loss function, batch size, and number of training epochs were aligned with those used for EfficientNet-B7, except where mandatory adjustments were needed due to memory constraints or architectural compatibility. The aim of this consistency was to minimize experimental variables, allowing for more accurate attribution of any observed differences to the nature of the architecture itself, rather than to discrepancies in the training pipeline or hyperparameter settings.

Chapter 7

Empirical Evaluation and Results

This chapter is dedicated to the empirical evaluation of the proposed model, with the aim of thoroughly analyzing its performance under various training configurations. Following a description of the experimental protocol and the key variables adopted, the results obtained on the two test sets of different sizes, introduced in the previous chapters, are presented, followed by a comparison with alternative architectures and an in-depth analysis of computational efficiency and generalization capability. The chapter concludes with a comparative visualization of the main evaluation metrics and a summary of the models that achieved the best performance in terms of stability and computational sustainability.

7.1 Overview of the Evaluation Protocol

The evaluation phase followed an experimental protocol consistent with the methodological framework outlined in the preceding chapters, with the objective of systematically comparing the performance of different binary classification models in detecting manipulated video content. The reference model was EfficientNet-B7, previously introduced in the section on architectural choices. The training process employed a 5-fold cross-validation strategy, with the best model from each fold saved based on validation loss, as thoroughly detailed in Chapter 6.

Model performance was primarily assessed using the logarithmic loss (log loss) metric, selected for its ability to capture not only classification accuracy but also the calibration of predicted probabilities. This metric is also the official evaluation criterion used in the DeepFake Detection Challenge (DFDC), making it particularly suitable for standardized comparisons. In addition, complementary metrics were computed, including F1 score, precision, recall, AUC-ROC, and weighted precision, to provide a more comprehensive performance evaluation. The testing phase was conducted on two separate datasets:

- 10 GB set comprising 3,464 videos,
- 50 GB set comprising 11,335 videos.

This distinction allowed for an evaluation of model behavior as the quantity and diversity of input data increased, effectively simulating scenarios of growing complexity and conditions close to real-world, uncontrolled environments. Finally, the introduction of key experimental variables, the percentage of training data, the number of training epochs, and the model architecture, enabled an in-depth analysis of the trade-off between accuracy, computational efficiency, and generalization capacity.

7.2 Key Experimental Variables

This section outlines the main experimental variables that guided the design and evaluation of the deepfake detection model. In particular:

- **Training Data Percentage:** one of the central questions of this experiment was: how much data is truly necessary to develop an effective and generalizable model? To explore this, five incremental training set configurations were tested (5%, 10%, 15%, 20%, and 100%), while keeping the test set constant. The objective was to find a balance between sample size and predictive capability, and to determine whether high performance could be achieved even with a reduced subset. Once the 20% configuration demonstrated already strong results (notably in terms of log loss and F1 score), further intermediate increments (e.g., 25%, 30%) were deemed unnecessary, as the additional training time was unlikely to yield substantial improvements. The analysis therefore focused on comparing this “optimal” threshold against the upper bound (100%) to assess whether full-data training would significantly outperform lighter and more cost-efficient alternatives. This strategic approach maximized experimental insight while minimizing computational waste, emphasizing scalability and efficiency in real-world applications.
- **Number of Training Epochs:** alongside data quantity, the number of training epochs, that is, the number of complete passes over the dataset, was also varied. Two training depths were tested for each configuration: 5 and 10 epochs. This made it possible to observe how increased exposure to data influences model convergence, prediction stability, and the risk of overfitting, especially in low-data scenarios. A minimum of 5 epochs was required to allow the model to sufficiently learn discriminative patterns, particularly for a complex task such as deepfake detection. Fewer than 5 epochs led to unstable and unreliable results. On the other hand, 10 epochs represented a balanced choice between thorough learning and computational sustainability, especially for full-data configurations.
- **Model Architecture:** to evaluate the impact of architectural choices, three models were compared: EfficientNet-B7 (the primary model), ResNet18, and Xception. EfficientNet-B7 was selected for its strong ability to capture fine visual artifacts in

manipulated faces, despite its higher computational cost. ResNet18, being more compact and shallow, offered a lightweight and fast alternative, while Xception was considered an intermediate choice in terms of performance and resource requirements. Unlike EfficientNet-B7, which was analyzed across different training percentages and epochs, ResNet18 and Xception were only trained on 100% of the data. This choice limited training time while still allowing for evaluation under ideal conditions. Both architectures were tested on the 10 GB and 50 GB test sets to assess their behavior under increasing complexity and their generalization to realistic scenarios. The goal was not to replicate the entire experimental design of EfficientNet-B7, but to provide a focused comparative analysis, highlighting the strengths and structural limitations of each model. This strategy reduced the number of experiments while preserving analytical rigor, and enabled a clear comparison of accuracy, training time, and computational cost, key considerations for selecting an appropriate model based on available resources and intended applications.

7.3 Performance on the 10 GB Test Set

The first phase of the experimental evaluation was conducted on a relatively small test set (3,464 videos, approximately 10 GB) with the goal of analyzing the impact of training data volume and number of epochs on the performance of the EfficientNet-B7 model. Five different fractions of the training dataset (5%, 10%, 15%, 20%, and 100%) were tested, each with two levels of training depth (5 and 10 epochs). The analysis highlighted a particularly significant result: the model trained with 20% of the data for 5 epochs achieved the lowest log loss overall (0.2809), with the exception of the 100%–10 epochs configuration, which reached a slightly better log loss (0.2803). However, the latter required approximately ten times more training time. This suggests that a lighter configuration can still deliver high performance while significantly reducing computational costs.

Train %	Epochs	Test Set	Accuracy	Precision	Recall	F1 Score	ROC AUC	Log Loss	Weighted Precision	Training Time
5	5	10 GB	0.8303	0.8895	0.9211	0.905	0.6979	0.3867	0.8089	∼3.30 h
10	5	10 GB	0.8879	0.9045	0.9559	0.934	0.8046	0.3146	0.8513	∼6.30 h
15	5	10 GB	0.9046	0.8979	0.9779	0.9565	0.8347	0.3069	0.8679	∼10 h
20	5	10 GB	0.8932	0.9017	0.9859	0.9419	0.8322	**0.2809**	0.8759	∼13 h
100	5	10 GB	0.8832	0.8915	0.9318	0.9306	0.8225	0.3104	0.8634	∼2.7 d

Table 2: Performance of the model on 5 epochs (Test set: 10 GB)

The comparison among the 5-epoch configurations confirms the efficiency of the 20%–5 epochs model, which also achieved a strong F1 score of 0.9419 and a recall of

98.59%, showing an excellent ability to detect the minority class (i.e., fake videos). On the other hand, the 100%–5 epochs model recorded a worse log loss (0.3104) despite using significantly more data, possibly indicating cognitive overload or the presence of noisy information in the full dataset. To further explore the effect of training depth, each configuration was subsequently evaluated with 10 epochs. Results show a general improvement in the metrics, with the 100%–10 epochs model achieving the lowest absolute log loss (0.2803) and an F1 score of 0.9358, which is very close to the 20%–5 epochs configuration. However, this came at a much higher computational cost, making it less desirable in practice.

Train %	Epochs	Test Set	Accuracy	Precision	Recall	F1 Score	ROC AUC	Log Loss	Weighted Precision	Training Time
5	10	10 GB	0.8025	0.8924	0.8813	0.8868	0.7082	0.4015	0.8143	∼7 h
10	10	10 GB	0.8733	0.9122	0.9467	0.9292	0.7916	0.3252	0.8587	∼13 h
15	10	10 GB	0.8938	0.9037	0.9839	0.9421	0.8295	0.3081	0.8763	∼20 h
20	10	10 GB	0.8909	0.9073	0.9753	0.9401	0.8295	0.2925	0.8717	∼1 d
100	10	10 GB	0.8834	0.9061	0.9674	0.9358	0.8581	**0.2803**	0.8618	∼4 d

Table 3: Performance of the model on 10 epochs (Test set: 10 GB)

A direct comparison between the two best-performing models 100%–10 epochs and 20%–5 epochs, reveals the following insights:

- The log loss of the 100%–10 epochs model is only slightly better (0.2803 vs 0.2809).
- The 20%–5 epochs model achieved a higher recall (0.9859 vs 0.9674), a higher F1 score (0.9419 vs 0.9358), and a superior weighted precision.
- The complete model with 10 epochs required significantly more time and resources, thus being less efficient relative to the marginal performance gain.

These results demonstrate that a model trained on a reduced portion of the dataset with a limited number of epochs can offer highly competitive performance. This configuration represents an optimal trade-off between accuracy and computational sustainability, an insight of particular relevance in real-world scenarios where the availability of data, time, and resources may be constrained.

7.4 Performance on the 50 GB Test Set

The second phase of evaluation was conducted on a significantly larger test set (11,335 videos, approximately 50 GB), characterized by a greater variety in terms of visual quality, camera angles, manipulation techniques, and acquisition conditions. This more realistic and complex scenario was designed to assess the generalization ability of the models under less controlled and more real-world-like conditions. The configurations tested mirrored those of the previous experiment: EfficientNet-B7 models trained on increasing proportions of the training dataset (5%, 10%, 15%, 20%, and 100%) and with two levels of training depth (5 and 10 epochs). The results confirmed the trends observed in the first test, though with some notable differences due to the higher data complexity.

Train %	Epochs	Test Set	Accuracy	Precision	Recall	F1 Score	ROC AUC	Log Loss	Weighted Precision	Training Time
5	5	50 GB	0.8738	0.878	0.9899	0.9306	0.7418	0.3164	0.8607	∼3.30 h
10	5	50 GB	0.8779	0.9276	0.9446	0.9159	0.7802	0.3375	0.8448	∼6.30 h
15	5	50 GB	0.8865	0.9179	0.9774	0.9198	0.8174	0.3137	0.8603	∼10 h
20	5	50 GB	0.8778	0.880	0.9925	0.9329	0.852	0.3284	0.8711	∼13 h
100	5	50 GB	0.8632	0.8719	0.9633	0.9412	0.8122	0.3601	0.8545	∼2.7 d

Table 4: Performance of the model on 5 epochs (Test set: 50 GB)

In this context, the model trained with 20% of the data for 10 epochs achieved the best overall performance, obtaining the lowest log loss value of 0.2862 across all configurations tested on the 50 GB set. This is a particularly significant outcome, as it demonstrates how a deeper training process (10 epochs) on a limited but well-curated subset of the dataset enables the model to better adapt to the diversity and unpredictability of the test set, enhancing probability calibration and overall prediction accuracy.

Train %	Epochs	Test Set	Accuracy	Precision	Recall	F1 Score	ROC AUC	Log Loss	Weighted Precision	Training Time
5	10	50 GB	0.8714	0.881	0.9823	0.9289	0.7306	0.3724	0.851	∼7 h
10	10	50 GB	0.8672	0.8976	0.9535	0.9247	0.8396	0.3142	0.8496	∼13 h
15	10	50 GB	0.8778	0.8796	0.9930	0.9329	0.8224	0.3383	0.8720	∼20 h
20	10	50 GB	0.8763	0.8843	0.9841	0.9315	0.8718	**0.2862**	0.8603	∼1 d
100	10	50 GB	0.8708	0.878	0.9859	0.9288	0.8697	0.3795	0.8516	∼4 d

Table 5: Performance of the model on 10 epochs (Test set: 50 GB)

Unlike the 10 GB test, where the 20%–5 epochs configuration achieved an optimal balance between efficiency and performance, in this case the depth of training proved crucial. Increasing the number of epochs allowed the model to refine the weight parameters and capture more complex patterns without overfitting. Conversely, configurations with fewer epochs or larger training datasets did not yield substantial improvements and, in some cases, resulted in higher log loss values. Furthermore, the 20%–10 epochs model demonstrated excellent generalization capabilities, showing stable results across different data samples (as confirmed by robustness testing using multiple random seeds), and delivering performance comparable to or even exceeding that of models trained on 100% of the data. The 50 GB test highlights the fact that training quality outweighs quantity: a well-balanced model in terms of training depth and dataset size can outperform more intensive approaches, provided that the data used is sufficiently representative. Therefore, the 20%–10 epochs configuration is confirmed as the most suitable for complex environments, offering the best compromise between performance, generalization, and computational sustainability.

7.5 Comparison with Alternative Architectures

In addition to the extensive experiments conducted using EfficientNet-B7, the project included a systematic comparison with two alternative architectures: ResNet18 and Xception. These models were evaluated on both the 10 GB test set (3,464 videos) and the larger 50 GB test set (11,335 videos), in order to assess their effectiveness in terms of accuracy, probability calibration, training time, and generalization ability. The results confirm the superior performance of EfficientNet-B7 in high-complexity scenarios. Its architecture, based on the principle of compound scaling, enables an optimal balance between depth, width, and resolution. On both test sets, it achieved the best log loss scores (as low as 0.2803 on the 10 GB set and 0.2862 on the 50 GB set), demonstrating its high capacity for accurate probability calibration.

Metric	**Xception (10 GB)**	**ResNet18 (10 GB)**	**ResNet18 (50 GB)**	**Xception (50 GB)**
Accuracy	0.816	0.138	0.1498	0.8658
Precision	0.882	0.630	0.5405	0.8719
Recall	0.997	0.044	0.0392	0.9883
F1 Score	0.936	0.082	0.0731	0.9265
ROC AUC	0.752	0.277	0.2472	0.7536
Log Loss	0.546	5.113	6.4419	0.6294
Weighted Precision	0.865	0.567	0.4802	0.8431

Table 6: Comparison of performance metrics between Xception and ResNet18

The Xception model, although lighter and less computationally demanding, delivered surprisingly strong results. On both test sets, it achieved a recall close to 99% and an F1 score comparable to EfficientNet-B7, particularly on the 10 GB set. However, it exhibited higher log loss values (0.546 on the 10 GB and 0.629 on the 50 GB set), indicating lower reliability in probability estimation. While the model effectively detects both false positives and false negatives, it is less reliable in the confidence associated with its predictions. ResNet18, on the other hand, showed significantly lower performance in both contexts. On the 10 GB test set, it recorded a log loss above 5.1 and a recall of approximately 4%, making it unsuitable for deepfake detection tasks. On the larger 50 GB test set, performance remained poor, with an accuracy of only 14.9% and a log loss of 6.44. These results highlight the limited representational capacity of the architecture, which, despite being fast to train and lightweight, fails to effectively distinguish between real and manipulated content. In terms of training time, Xception required approximately 5 hours to complete training, while ResNet18 proved even faster, with an average training time of about 3 hours, due to its lighter and shallower architecture. From a computational perspective, EfficientNet-B7 was the most demanding model in terms of resources and training time, but delivered consistently high and stable performance, reaffirming its role as the go-to solution for those seeking maximum accuracy and reliability. Xception may serve as a viable alternative in scenarios where recall is prioritized and trade-offs in probability calibration are acceptable. ResNet18, although suitable for low-power environments due to its rapid training and low computational requirements, proved inadequate without significant architectural or training optimizations.

7.6 Efficiency and Trade-Off Analysis

One of the central insights emerging from the entire experimental workflow concerns the balance between predictive performance and computational cost. The comparative analysis of different configurations revealed that it is not always necessary to rely on the full dataset or extremely deep models to achieve excellent results. There exist optimal thresholds beyond which the marginal benefit becomes significantly lower relative to the time and resources required. In particular, the EfficientNet-B7 model trained with only 20% of the dataset for 10 epochs proved to be one of the most effective configurations in terms of trade-off. It achieved a minimum log loss of 0.2862 on the 50 GB test set, with performance comparable to or even exceeding that of the model trained on 100% of the data for the same number of epochs. The training time for this configuration was reduced by approximately 75% compared to the full-data setup, enabling more sustainable resource management and faster development and validation cycles. This result highlights that a well-curated subset of data, combined with deeper training, can lead the model to generalize better, avoiding the risk of overfitting on noise or redundancy typically found in large-scale datasets. Moreover, using a smaller data volume enhances the scalability of the pipeline, as it allows for faster updates, easier deployment in lower-powered environments, and more frequent retraining with new data.

On the other hand, the full-data approach can be advantageous in high-stakes scenarios where maximum data coverage is essential to prevent false negatives, or when high-performance infrastructure is available. In such contexts, training on the entire dataset, although it requires several days of computation, enables full exploitation of the model's expressive capacity, potentially improving robustness in rare cases or edge scenarios not represented in smaller subsets. However, the results show that the benefit of the full-data approach is not proportional to the increase in resources used. When comparing the EfficientNet-B7 model trained on 100% of the data for 10 epochs with the 20%–10 epochs model, the difference in log loss is marginal, while the training time is approximately four times longer. This imbalance between cost and benefit makes the 20% configuration a strategically more efficient choice for most operational applications, especially in environments where response time, scalability, and reproducibility are critical.

Another key element in the efficiency analysis is the classification time per video. For all configurations based on EfficientNet-B7, regardless of the data percentage or number of epochs, the average time to classify a single video as "real" or "fake" is approximately 1.5 seconds. This value includes the entire testing pipeline: frame reading (32 frames per video), face detection and preprocessing, frame-level inference, and final prediction aggregation. The consistent latency across configurations makes the model particularly suitable for near real-time applications such as automated video stream monitoring, large-scale screening, or integration into content verification platforms. The low inference time, combined with high accuracy, represents an additional strength of EfficientNet-B7-based solutions in real-world scenarios. As for alternative architectures, the Xception and ResNet18 models, due to their lighter computational footprint, showed lower inference times, around 1 second per video or slightly less. However, this came at the expense of generally lower performance. The 20%–10 epochs configuration stands out as the optimal balance between accuracy, computational efficiency, and implementation speed. While the full-data approach may be theoretically more comprehensive, it should be reserved for specific cases where maximal data coverage and minimal tolerance for error are absolute requirements. The ability to achieve high performance using a limited fraction of the data confirms the effectiveness of targeted strategies in the design of deep learning-based detection systems.

7.7 Robustness and Repeatability

To assess the stability and reliability of the configurations identified as optimal, specifically, EfficientNet-B7 trained with 20% of the data for 5 epochs (tested on the 10 GB set), and EfficientNet-B7 trained with 20% of the data for 10 epochs (tested on the 50 GB set), a robustness and repeatability analysis was conducted. The experiments were carried out using the full training dataset of 10,420 videos, from which two distinct 20% subsets were randomly generated using random seeds 42 and 99. The goal was to determine whether the strong performance previously observed was the result of a fortuitous data selection, or whether the model could generalize robustly across different training data samples of equal size and structure.

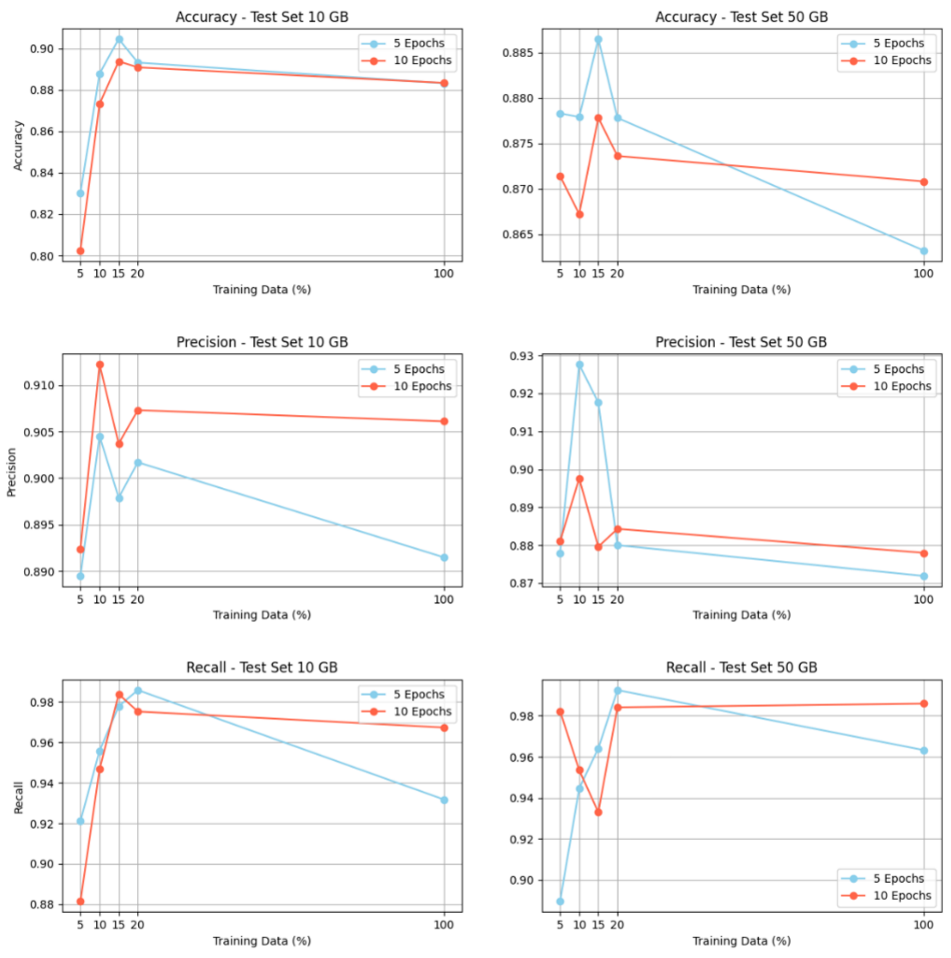
Random Seed	Epochs	Test Set	Accuracy	Precision	Recall	F1 Score	ROC AUC	Log Loss	Weighted Precision
42	5	10 GB	0.8932	0.9017	0.9859	0.9419	0.8322	0.2809	0.8759
99	5	10 GB	0.8943	0.8979	0.9824	0.9428	0.8317	0.2811	0.8831
42	10	50 GB	0.8763	0.8843	0.9841	0.9315	0.8718	0.2862	0.8603
99	10	50 GB	0.8823	0.8843	0.9841	0.9355	0.8853	0.2872	0.8831

Table 7: Robustness of the 20% configuration with different training samples

The results confirmed a high level of consistency in performance metrics across both test sets. For instance, on the 10 GB test set with 5 epochs, the log loss remained virtually unchanged: 0.2809 with seed 42 versus 0.2811 with seed 99. Similarly, both F1 score and recall showed only minimal variations (F1: 0.9419 vs 0.9428), well within the bounds of acceptable statistical variability. Likewise, on the 50 GB test set, the model trained with the two seeds yielded stable results: log loss increased only slightly from 0.2862 to 0.2872, and the F1 score ranged from 0.9315 to 0.9355. These minor differences demonstrate that the 20%–10 epochs configuration tested on 50 GB is not the result of a particularly favorable subset of data, but rather a robust, repeatable, and methodologically sound setup. These findings indicate that the model is resilient to sample variability, meaning it maintains high performance even when the composition of the training data changes, assuming size and training parameters remain constant. This trait is essential in real-world scenarios, where data may evolve over time or originate from heterogeneous sources. Stability in predictions and the ability to behave consistently across varying input conditions provide a crucial foundation for deploying the model in production environments. The repeatability experiment further reinforced the validity of the EfficientNet-B7 –20%–10 epochs configuration, confirming it as an optimal solution not only in terms of accuracy and generalization, but also in terms of operational reliability.

7.8 Comparative Visualization of Performance

To support the experimental analysis conducted, the following image provides a graphical summary of the trends in the main evaluation metrics (Accuracy, Precision, Recall, F1 Score, ROC AUC, Log Loss, and Weighted Precision) as a function of the percentage of training data used (5%, 10%, 15%, 20%, 100%) and the number of training epochs (5 or 10). The plots are arranged in vertical pairs, with the 10 GB test set on the left and the 50 GB test set on the right, allowing for a direct comparison between the two experimental conditions. Blue lines represent models trained for 5 epochs, while red lines indicate models trained for 10 epochs.



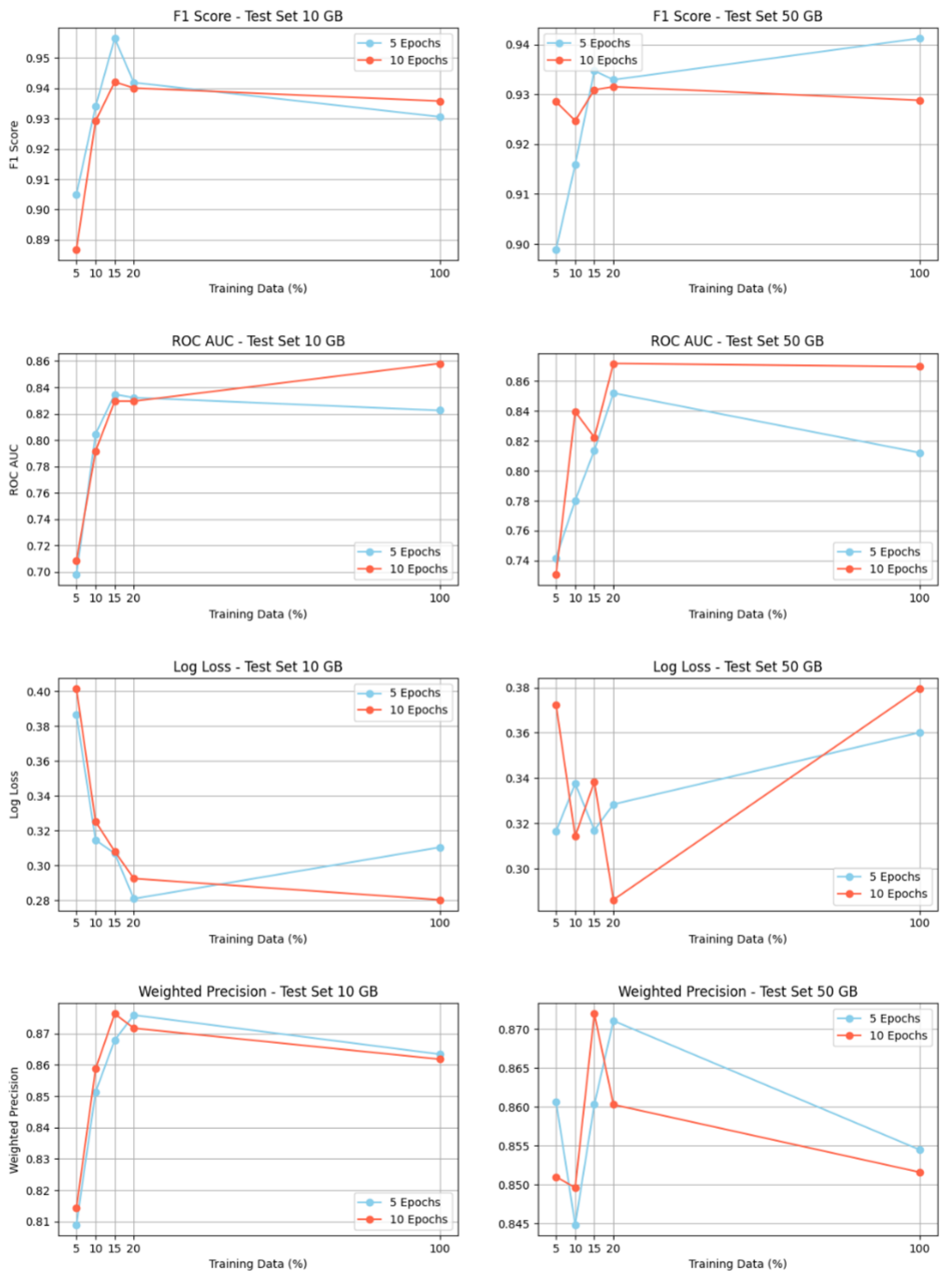


Figure 4: Visual Comparison of Evaluation Metrics by Training Sizes and Epochs.

From the visual analysis, it clearly emerges that the 20%–10 epochs configuration represents a particularly favorable balance point for nearly all evaluation metrics. Specifically, in the 50 GB test set, increasing the number of epochs results in a marked improvement in metric stability and a reduction in log loss, indicating a better calibration of predicted probabilities. In the 10 GB test set, the benefits of deeper training are more limited but still consistent.

- **Accuracy** increases rapidly at lower training percentages in the 10 GB set, peaking at the 20%–5 epochs configuration. Further increases in training depth yield only marginal improvements. In the 50 GB test set, models trained for 10 epochs show greater stability, while the 100%–10 epochs configuration exhibits a slight decline in accuracy, possibly due to noise or data redundancy.

- **Precision** is more sensitive to training depth. On the 10 GB test set, the 10-epoch configuration maintains consistently high values, whereas the 5-epoch model exhibits greater variability. On the 50 GB test set, precision tends to decline as training size increases, with the 20%–10 epochs setup providing the best compromise.
- **Recall** is high across all 5-epoch configurations, especially at 20%. However, deeper training helps reduce variability between training percentages and improves metric consistency, particularly in the more complex 50 GB test set.
- **F1 Score**, which balances precision and recall, confirms the strong performance of the 20%–10 epochs configuration in both test sets. On the 50 GB test set, it achieves the highest score, outperforming the 100%–10 epochs model.
- **ROC AUC** improves in both test sets, with stronger gains observed in 10-epoch models. In the 10 GB test set, the metric plateaus around 15–20% training data. In contrast, the 50 GB test set shows more fluctuations, reflecting its higher complexity.
- **Log Loss** results highlight that increasing training data does not always yield better calibration. On the 10 GB test set, the lowest log loss is achieved with the 20%–5 epochs model. On the 50 GB test set, the 20%–10 epochs configuration performs best, while the 100%–10 epochs version suffers from a noticeable increase in log loss.
- **Weighted Precision** follows a consistent trend with other metrics: the 20% configurations deliver high and stable results, while the full-data version experiences a slight decline. The 20%–10 epochs model maintains strong performance across both test sets and stands out as the most balanced overall.

7.9 Summary of Key Results

The analysis of results made it possible to identify distinct optimal configurations depending on the size and complexity of the test set. For the 10 GB test set, which is relatively smaller and more homogeneous, the EfficientNet-B7 model trained on 20% of the data for 5 epochs proved to be the most efficient and effective. It achieved the highest

F1 score (0.9419), a very low log loss (0.2809), and a reduced training time of approximately 13 hours. This makes it an excellent solution for scenarios characterized by limited computational resources or requiring rapid development cycles. The configuration using EfficientNet-B7 trained on 100% of the data for 10 epochs also delivered excellent results on the same test set, registering the lowest log loss overall (0.2803). However, the required training time was approximately six times longer, making it less advantageous when considering the trade-off between computational cost and marginal performance gain. Therefore, its use may be justified only in high-stakes contexts, where even minimal improvements in accuracy or probability calibration are considered strategically relevant.

Regarding the 50 GB test set, which is more heterogeneous and complex, the best-performing configuration was EfficientNet-B7 trained on 20% of the data for 10 epochs. With a log loss of 0.2862 and an F1 score of 0.9315, this model demonstrated strong capability to handle data variability while maintaining high computational efficiency, with a training time of approximately one day. As such, this configuration is confirmed to be the most balanced for large-scale production scenarios, where strong performance must be achieved without compromising time and resource constraints. In contrast, the EfficientNet-B7 configuration trained on 100% of the data for 10 epochs required over four days of training, while yielding a slightly lower F1 score (0.9288) and a significantly higher log loss (0.3795). These results make it less competitive compared to the lighter configuration and, for many practical applications, difficult to justify in terms of efficiency. For smaller or more controlled datasets such as the 10 GB set, the 20%–5 epochs model proves the most advantageous in terms of accuracy, speed, and stability. For larger and more complex datasets, such as the 50 GB set, the 20%–10 epochs configuration stands out as the optimal choice, capable of effectively balancing predictive performance and computational sustainability. The full-data (100%) approach, while delivering solid results, should be reserved for situations where computational resources are not a limiting factor and the objective is to maximize every percentage point of performance, even at the cost of significant time and resource investment.

Train %	**Epochs**	**Test Set**	**Accuracy**	**F1 Score**	**Log Loss**	**Weighted Precision**	**Training Time**
20%	5	10 GB	0.8932	0.9419	0.2809	**0.8759**	∼13 hours
100%	10	10 GB	0.8834	0.9358	0.2803	0.8618	∼4 days
20%	10	50 GB	0.8763	0.9315	0.2862	0.8603	∼1 day
100%	10	50 GB	0.8708	0.9288	0.3795	0.8516	∼4 days

Table 8: Summary Table – EfficientNet-B7 (Key Configurations)

Chapter 8

General Discussion and Implications

This chapter briefly discusses the ethical, methodological, and practical implications of the work. It highlights potential biases and limitations of the adopted approach, offering critical reflections on the applicability and reliability of the proposed deepfake detection.

8.1 Ethical and Practical Implications of Detection

The automatic detection of deepfakes raises significant ethical and practical considerations. From an application standpoint, models such as EfficientNet-B7, capable of classifying a video in approximately 1.5 seconds with high accuracy, offer promising solutions for integration into digital surveillance systems, fact-checking frameworks, and content moderation platforms. However, the availability of such tools must be accompanied by responsible usage. While the technology can help mitigate dangerous phenomena such as disinformation, non-consensual content, or digital fraud, its large-scale deployment introduces risks related to privacy, potential abuse of power, and algorithmic censorship. It is essential that detection technologies be implemented transparently, with explainable and verifiable outcomes, and within a well-defined regulatory framework to prevent misuse. In this context, artificial intelligence applied to deepfake detection emerges as a valuable, but not neutral, tool, demanding careful, participatory, and multidisciplinary governance.

8.2 Methodological Limitations and Potential Biases

The methodology adopted in this study ensured a high level of experimental rigor and reproducibility, thanks to the use of cross-validation, best-model checkpointing, and repeated experiments on subsets generated with fixed random seeds. Nevertheless, several factors could represent sources of bias or methodological limitations. A first concern relates to the nature of the dataset itself (DFDC), which, although extensive, does not encompass all possible video manipulation techniques currently in use. Some of the more recent or less widespread methods may not have been adequately represented, potentially limiting the model's generalizability in emerging scenarios. Secondly, the choice of architectures and hyperparameters, though justified by computational constraints and prior literature, may have influenced the outcomes. For instance, transformer-based approaches or spatiotemporal models were not explored, even though they might offer advantages in different contexts. A potential drawback, though more accurately reflecting the complexity of the task, is the relatively long training time (up to four days for full-data configurations). However, such durations are entirely reasonable given the large volume of data involved, the need to process tens of thousands of frames per video, and above all, the emphasis on ensuring replicability and robustness of the results. In real-world applications, this initial training effort is more than compensated by the resulting speed and reliability of the inference phase.

8.3 Limitations of the Work

Despite the methodological robustness of the analysis, this study presents certain limitations that deserve to be acknowledged. The first concerns the adopted approach, which is based on a frame-based pipeline and, therefore, does not take into account the temporal component of videos. While this choice aligns with many strategies commonly found in the literature, it may reduce the effectiveness of detection in cases where manipulation signals are distributed over time or manifest through motion inconsistencies, which are difficult to detect when frames are analyzed in a static and isolated manner. A second limitation relates to the handling of video length. In this study, the DFDC dataset was used, in which videos are relatively short (approximately 10 seconds), allowing each video to be treated as a single analytical unit. However, in real-world scenarios, video content may have significantly longer durations, making the direct application of the model challenging or inefficient. A more scalable approach would involve segmenting longer videos into shorter temporal fragments to be analyzed independently. The predictions associated with each segment could then be aggregated to obtain an overall assessment. Although this strategy increases the model's flexibility and applicability to extended content, it may also introduce critical issues: fragmented analysis risks losing the coherence of the overall content or producing inconsistent predictions between segments of the same video, particularly in the presence of partial, intermittent, or localized manipulations. A simple summation of isolated decisions may therefore fail to accurately reflect the true nature of the video as a whole, ultimately compromising the quality of the final judgment.

Chapter 9

Conclusions and Future Directions

This study set out to systematically address the problem of detecting content generated by artificial intelligence, with a specific focus on deepfake videos, a phenomenon that is becoming increasingly widespread and insidious. The initial objectives of this research included: describing the generative technologies underlying deepfakes; exploring the ethical challenges posed by the proliferation of synthetic content; experimentally evaluating the effectiveness of classification models trained on real-world datasets; and promoting a critical reflection on the importance of interpretability and transparency in artificial intelligence systems. All these objectives were addressed throughout the course of the work. After outlining in the first chapter the theoretical, technological, and ethical framework that contextualizes the phenomenon, the study combined a conceptual discussion with an advanced experimental analysis aimed at evaluating the behavior of deep learning models in binary classification tasks involving real versus manipulated videos. The original contribution of this research lies in the development and validation of a pipeline based on modern convolutional architectures applied to video frames. The experimental phase demonstrated that it is possible to achieve very high performance even when using a reduced portion of the training dataset, provided that an appropriate training strategy is adopted. This result underscores that efficiency and accuracy are not necessarily tied to the massive use of data, but rather to intelligent selection and optimization of the learning process. The proposed models also proved robust in terms of replicability. Experiments conducted on different random subsets showed good performance stability, indicating that the solutions identified are consistent and not dependent on a specific initial data split. Moreover, the system was able to classify videos with very short inference times, making it suitable for deployment in real-world scenarios where speed is a crucial requirement. Looking ahead, several promising directions for further development emerge. The first involves moving beyond the frame-based approach toward models capable of capturing the temporal dimension of videos, by leveraging complete sequences through architectures such as video transformers. The second involves the adoption of interpretability techniques to enhance the transparency of model decisions and increase trust in sensitive applications. Finally, it will be crucial to train models on increasingly updated, diverse, and representative datasets that reflect the plurality of subjects and contexts in which deepfakes may appear, in order to strengthen their generalizability and adaptability at a global scale. In conclusion, this study represents a concrete step toward the design of deepfake detection models that are not only accurate, but also efficient, replicable, and ready for deployment in real-world environments. It makes a meaningful contribution to the scientific and societal debate on the safe use of generative artificial intelligence and the development of reliable tools to safeguard the integrity of information.

References

AI Act. (2024). *Regulation (EU) 2024/1689 of the European Parliament and of the Council on Artificial Intelligence*. Official Journal of the European Union.

AI ACT Brief, 2024. Official Journal of the European Union.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., & Liang, P. (2021). *On the opportunities and risks of foundation models*. ArXiv preprint.

Dasgupta, D., Vanugopal, D., & Dattagupta, K. (2023). *A review of generative AI from historical perspectives*. Technical report.

Feuerriegel, S., Hartmann, J., Janiesch, C., & Zschech, P. (2024). *Generative AI*. *Business & Information Systems Engineering*, 66(1).

Fui-Hoon Nah, F., Zheng, R., Cai, J., Siau, K., & Chen, L. (2023). *Generative AI and ChatGPT: Applications, challenges, and AI-human collaboration*. *Journal of Information Technology Case and Application Research*, 25(3).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., & Bengio, Y. (2020). *Generative adversarial networks*. *Communications of the ACM*, 63(11)

Gupta, S., Kaushik, A., & Kaur, P. (2025). *Deepfake Overview: Generation, Detection, Risks and Opportunities*.

He, R., Cao, J., & Tan, T. (2025). *Generative artificial intelligence: A historical perspective*. *National Science Review*, 12(5).

Jaakkola, T., & Haussler, D. (1998). *Exploiting generative models in discriminative classifiers*. In *Advances in Neural Information Processing Systems*, 11.

Kaggle. *Deepfake Detection Challenge*. Hosted by Meta and Kaggle.
https://www.kaggle.com/competitions/deepfake-detection-challenge

Kenthapadi, K., Lakkaraju, H., & Rajani, N. (2023, August). *Generative AI meets responsible AI: Practical challenges and opportunities*. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Keras, *EfficientNet B0 to B7. https://keras.io/api/applications/efficientnet/*

Kılınç, H. K., & Keçecioğlu, Ö. F. (2024). *Generative Artificial Intelligence: A Historical and Future Perspective. Academic Platform Journal of Engineering and Smart Systems*, 12(2).

Kumar, P., Vashishtha, S., Sharma, P., & Agarwal, E. (2024). *Exploring the efficacy of adaptive learning platforms enhanced by artificial intelligence: A comprehensive review*. In *Integrating Generative AI in Education to Achieve Sustainable Development Goals*.

Patel, Y., Tanwar, S., Gupta, R., Bhattacharya, P., Davidson, I. E., Nyameko, R., & Vimal, V. (2023). *Deepfake generation and detection: Case study and challenges*. IEEE Access.

Paullada, A., Raji, I. D., Bender, E. M., Denton, E., & Hanna, A. (2021). *Data and its (dis)contents: A survey of dataset development and use in machine learning research. Patterns*, 2(11).

Pawelec, M. (2024). *Decent deepfakes? Professional deepfake developers' ethical considerations and their governance potential*. AI and Ethics.

Ruthotto, L., & Haber, E. (2021). *An introduction to deep generative modeling. GAMM-Mitteilungen*, 44(2).

Sankaran, K., & Holmes, S. P. (2023). *Generative models: An interdisciplinary perspective. Annual Review of Statistics and Its Application*, 10(1).

Stratis, K. (2024). *What is Generative AI? A generative AI primer for business and technical leaders*. Sebastopol, CA: O'Reilly Media.

Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., & Zhu, J. (2019). *Explainable AI: A brief survey on history, research areas, approaches and challenges*. Springer International Publishing.

Weisz, J. D., He, J., Muller, M., Hoefer, G., Miles, R., & Geyer, W. (2024, May). *Design principles for generative AI applications*. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*.

Zhao, W., Alwidian, S., & Mahmoud, Q. H. (2022). *Adversarial training methods for deep learning: A systematic review*. Algorithms, 15(8).