INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

UNIVERSITY OF WESTMINSTER

University of Westminster, Coat of Arms

GenDF: A Modular Framework for Evaluating Generalization and Adaptability in Deepfake Detection

A dissertation by

Miss. Kavini Fernando

W1833630 | 20200760

Supervised by

Miss. Kalhari Walawage

Submitted in partial fulfilment of the requirements for
the BSc in Computer Science degree at the University of Westminster.

April 2025

ABSTRACT

Deepfake media poses a growing threat to digital integrity, with existing detection systems struggling to generalize across unseen manipulation techniques, domains, or datasets. Most current models tend to overfit specific forgery types and require retraining to adapt to new threats, limiting their real-world applicability. This research addresses the need for a generalizable, explainable, and deployable deepfake detection solution capable of handling varied and evolving conditions.

To solve this, the study proposes GenDF, a modular deepfake detection framework built on EfficientNet-B0 and enhanced with three advanced techniques: Reinforcement Learning (RL) for dynamic augmentation selection, Contrastive Learning (CL) for feature consistency, and Test-Time Training (TTT) for real-time adaptation. The RL component uses a Proximal Policy Optimization (PPO) agent to select optimal augmentations during training. CL introduces a contrastive projection head that aligns representations of augmented pairs, and TTT employs a lightweight adapter trained with KL-divergence regularization to adjust feature embeddings during inference. These components are implemented across three system versions; Version A (RL), Version B (RL and CL), and Version C (RL, CL and TTT) and integrated into a usable Streamlit UI with Grad-CAM explainability (for A and B).

Evaluation using unseen validation data revealed that Version A achieved the most balanced performance, with an accuracy of 77.45%, F1 score of 81.89%, and AUC of 0.89, confirming the effectiveness of RL-based augmentation. Version B improved recall (89.51%) and interpretability through Grad-CAM but reduced precision. Version C added adaptability via TTT, with slight robustness gains but limited explainability. These findings contribute original insights into how modular components impact generalization, positioning GenDF as a foundation for future research in adaptable and explainable AI systems.

Subject Descriptors: Theory of computation -> Theory and algorithms for application domains -> Machine learning theory -> Sample complexity and generalization bounds

Keyword: Deepfake detection, generalization, computer vision, augmentation techniques, cybersecurity

DECLARATION

I hereby declare that this thesis is the result of my own independent work and has not been submitted, nor is it currently being considered for submission, for any other degree or academic award at any other university or educational institution. All concepts, data, and ideas taken from other sources have been appropriately cited throughout this document. This declaration affirms my commitment to academic integrity and the principles of ethical scholarly research.

Student Name: Kavini Fernando
Registration Number: w1833630 / 20200760

Signature: [illegible] _ _ _ _ _ _ _ _ _ _ _ _ _ **Date:** 15th April 2025

ACKNOWLEDGEMENT

The completion of this research has been an immensely fulfilling journey, marked by challenges, perseverance, and personal growth. Throughout this process, I have been fortunate to receive the guidance, encouragement, and unwavering support of many individuals, without whom this project would not have been possible.

First and foremost, I extend my sincere gratitude to my supervisor, Miss Kalhari Walawage, for her invaluable insights, constructive feedback, and continuous encouragement. Her guidance was instrumental in shaping the direction and quality of this research. I would also like to thank Mr Guhanathan Poravi, our module leader, for the dedicated support and advice he gave throughout this Final Year Project journey. Their collective mentorship has played a vital role in my academic and professional development.

I am deeply appreciative of the lecturers whose teaching and mentorship over the past four years, especially during this final year, have provided me with the knowledge and skills required to undertake this project. Their commitment to excellence in education has been a constant source of inspiration.

Furthermore, I extend my heartfelt appreciation to all the domain experts and participants who contributed their time and expertise to evaluate my research. Their insights greatly enriched the study and helped refine the approach.

I am grateful to my friends and fellow students for the encouragement and countless discussions that kept me motivated throughout this journey. Their support made this experience both rewarding and memorable.

Finally, I owe my deepest gratitude to my family, whose belief in me has been my greatest source of strength. Their continuous support, patience, and encouragement are what helped me navigate this challenging yet rewarding endeavor.

This thesis is a testament not only to my dedication but also to the collective support of everyone who stood by me along the way. I am sincerely grateful to each and every one.

PUBLICATIONS

1. **An Adaptive Learning Approach to Modular Deepfake Detection in Resource-Constrained Environments**

   Conference: Second International Research Conference in Education (IRCE 2025)

   Status: Abstract submitted on April 15, 2025

   Authors: D. L. K. S. Fernando*, K. S. A. Walawage

   Venue: Kailasapathy Auditorium and Faculty of Graduate Studies, University of Jaffna, Sri Lanka

   Dates: 14–15 June 2025

   Content: This abstract proposes a modular deepfake detection framework integrating Reinforcement Learning (RL), Contrastive Learning (CL), and Test-Time Training (TTT) to enhance model generalization and adaptability. Submitted under the sub-theme Innovative Practices and Digital Infrastructure, it highlights adaptive learning strategies for responsible AI in constrained environments, aligning with the conference's focus on sustainable digital infrastructure.

   Reference: see ***APPENDIX 01*** for the full abstract and submission proof.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Acronym	**Description**
AI	Artificial Intelligence
PPO	Proximal Policy Optimization
GAN	generative adversarial networks
RL	Reinforcement Learning
CL	Contrastive Learning
TTT	Test-Time Training
OST	one-shot test-time training
MAML	Model-Agnostic Meta-Learning
ADM	Artifact Detection Modules
MFS	Multi-Scale Facial Swap
ML	Machine Learning
DNN	Deep Neural Network
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
IIL	Implicit Identity Leakage
DRDA	Domain Refinement Data Augmentation
NN	Neural Networks
DT	Decision Tree
HQ	High Quality
LQ	Low Quality
FF++	FaceForensics++
DF1.0	DeeperForensics-1.0
DFDC	Deepfake Detection Challenge
UADFV	Unconstrained Audio DeepFake Videos Dataset
WDF	Wild Deepfake
DCL	Dual Contrastive Learning
AUC	Area Under the Curve

HTER	Half Total Error Rate
XAI	Explainable AI
OOADM	Object-Oriented Analysis and Design Methodology
OOP	Object-Oriented Programming
FPR	False Positive Rate
FNR	False Negative Rate
GPU	Graphics Processing Unit
UI	User Interface
FR	Functional Requirements
NFR	Non-functional Requirements

CHAPTER 01: INTRODUCTION

1.1 Chapter overview

This chapter introduces the author's research by presenting the problem background, key challenges, and motivation for this study. A review of existing work in deepfake detection is given to frame the research gap, highlighting the need for improvement in generalization to detect unseen manipulation techniques. The chapter then outlines the project's contributions, research questions, objectives, and aim to establish the study's direction.

1.2 Problem background

1.2.1 Deepfake technology and its expansion

The rise of Deepfake technology has created new challenges when it comes to validating digital information. Deepfake technology powered by Generative Adversarial Networks (GANs) and diffusion models have revolutionized synthetic media generation (Gupta *et al.*, 2023). While deepfakes were initially developed for harmless applications such as movie effects and digital avatars, it has now raised serious cybersecurity, misinformation, and privacy concerns (Masood *et al.*, 2023). Malicious deepfake applications include identity theft, political disinformation, financial fraud, and cyberbullying, making them a growing societal threat (Stanciu and Ionescu, 2024). Even non-experts can create highly realistic manipulated content with deepfake generation tools becoming widely accessible for everyone (Jain, Korshunov and Marcel, 2021). The increasing sophistication of GAN-based methods (e.g., StyleGAN, DeepFaceLab) and AI-driven video synthesis makes detection significantly harder (Gupta *et al.*, 2023) . As a result, effective and generalizable deepfake detection systems are a must to combat the misuse of AI-generated media.

1.2.2 Inconsistencies with current deepfake detection systems

Even though existing deepfake detection models can identify manipulated content, they frequently lack generalizability. One key issue is the lack of cross-dataset generalization, as most models are dataset-dependent and struggle to maintain accuracy when tested on unseen deepfake manipulations which limits their effectiveness beyond controlled datasets (Gupta *et al.*, 2023).

Additionally, deepfake generators are becoming more sophisticated because of using diffusion-based techniques and GAN inversions to bypass detection which in turn makes traditional classifiers ineffective (Talreja *et al.*, 2024). Another critical challenge is achieving adversarial robustness. Minor intentional modifications or adversarial noise added to deepfake images can easily mislead neural networks, causing detection failures. This highlights the need to explore adaptive augmentation strategies such as reinforcement learning that may enhance model resilience against such attacks (Stanciu and Ionescu, 2024). Moreover, computational trade-offs present further obstacles because advanced architectures like transformers and multimodal networks improve accuracy but demand high computational resources, making real-time deployment difficult (Gupta *et al.*, 2023). These limitations underscore the important requirement for novel deepfake detection approaches that can dynamically adapt to evolving deepfake techniques without being constrained by dataset biases or computational inefficiencies.

1.2.3 The ever-evolving field that demands further research

The above discussed information makes it clear that Deepfake detection is an ever-evolving field which demands ongoing research and development in order to stay up to date with novel deep-fake generating techniques. Stanciu and Ionescu remark that the authenticity of information encountered online has never been more questionable. More research on new algorithms and improved feature extraction methods are required to improve detection efficiency and reduce the risks caused by Deepfake media. (Talreja *et al.*, 2024)

1.3 Problem definition

Deepfake detection remains a significant challenge due to the lack of cross-dataset generalization, where detection models perform well on the datasets, they are trained on but struggle with unseen deepfakes. Additionally, most models rely on static training pipelines and are not equipped to dynamically adapt or generalize beyond dataset-specific patterns. These issues create vulnerabilities in real-world applications (Lin *et al.*, 2024) where evolving manipulation techniques and input domains are common (Gupta *et al.*, 2023).

Another key limitation is the static nature of existing data augmentation techniques. Current augmentation methods are manually predefined and do not dynamically adapt to improve model robustness. As a result, when new deepfake manipulation techniques emerge, detection models fail to generalize effectively, leading to outdated detection systems (Chen *et al.*, 2022).

This lack of generalization and adaptability presents critical risks across multiple domains:

- *Cybersecurity* – Deepfakes are being used in identity fraud and cyberattacks, posing serious security threats (Talreja *et al.*, 2024).
- *Misinformation Control* – Fake videos of public figures can spread false narratives, leading to misinformation and political manipulation (Masood *et al.*, 2023).
- *Digital Forensics* – Forensic investigators struggle to verify the authenticity of digital media due to undetectable deepfake manipulations (Stanciu and Ionescu, 2024).

1.4 Problem statement

Current deepfake detection models lack cross-dataset generalization and fail to dynamically adapt to evolving deepfake generation techniques, reducing their reliability in real-world applications (Lin *et al.*, 2024; Stanciu and Ionescu, 2024).

1.5 Research Motivation

With Deepfake material created using generative adversarial networks (GANs) and diffusion models (Lin *et al.*, 2024) being consumed on social media and in marketing on a regular basis, the author has seen how it can be confusing to differentiate between what is actually real and what is not. Public figures, public trust and personal safety are at high risk in the present day because of people being able to easily access Deepfake generation tools to quickly create false content at any given instance (Smelyakov, Kitsenko and Chupryna, 2024). And it is evident that when new types of deepfakes develop, the reliability of existing detection approaches becomes limited by their inability to generalize across the newly introduced different manipulation techniques and real world situations (Gupta *et al.*, 2023). Addressing this problem will help create more flexible and reliable detection systems that enhance our protection against dangerous Deepfake content while also possibly improving the security and trust of online environments in the future.

1.6 Research gap

Despite significant advancements in deepfake detection, several critical limitations persist in existing research. These limitations hinder the reliability, adaptability, and robustness of current detection models.

1.6.1 Lack of cross-dataset generalization

- Most deepfake detection models struggle to maintain consistent performance across diverse datasets. They often fail when exposed to real-world variations such as differences in lighting conditions, facial expressions, and image quality.
- This limitation arises because many models are trained on specific datasets that do not adequately represent the full spectrum of deepfake manipulations. As Lin *et al.*, (2024) highlights, this weakens the model's reliability when encountering previously unseen deepfake techniques, reducing its effectiveness in real-world applications.

1.6.2 Bias towards training data

- Many deepfake detection models exhibit a strong dependency on the specific types of forgery techniques present in their training datasets. This results in biased detection, limiting their ability to generalize when presented with new or evolving manipulation techniques.
- As noted by Gupta *et al.* (2023), such biases lower the fairness and accuracy of deepfake detection models. This is particularly problematic when the models are applied in real-world scenarios where the nature of deepfake manipulations is constantly changing, rendering detection models ineffective against new attack strategies.

1.6.3 Poor robustness against advanced deepfake techniques

- The rapid evolution of deepfake generation techniques, including Generative Adversarial Networks (GANs) and diffusion models, has outpaced the capabilities of most existing detection models. Many models fail to detect unique artifacts introduced by sophisticated forgery methods, making them less reliable against high-quality deepfake content.
- Chen *et al.* (2022) highlights that current models struggle with deepfake variations that involve subtle manipulations, such as blending improvements, advanced adversarial training, and noise

reduction techniques. Without improved robustness, detection systems remain vulnerable to novel deepfake methods that can bypass traditional detection mechanisms.

1.6.4 Limited application of novel data augmentation for generalization

While data augmentation holds promise for improving deepfake detection generalization, most existing models rely on static, predefined techniques. According to Smelyakov, Kitsenko and Chupryna, 2024, this limits model adaptability and reduces effectiveness against unseen manipulations.

There is a clear research gap in exploring dynamic, task-informed augmentation strategies. Reinforcement learning (RL) offers a novel way to select augmentations adaptively, potentially enhancing robustness and cross-dataset generalization.

Addressing this gap could help future models:

- Improve real-world generalization
- Reduce training data bias
- Enhance resilience to sophisticated forgeries
- Learn from a broader range of manipulation styles

1.7 Contribution to body of knowledge

The author will be addressing the following Alternative Hypothesis (H1)

H1- A modular augmentation-based framework combining reinforcement learning, contrastive learning, and test-time training enhances the robustness and adaptability of deepfake detection models across datasets and manipulation types, even under constrained accuracy trade-offs.

1.7.1 Contribution to the research domain

From an academic and technological standpoint, this research advances the field of generalizable deepfake detection by proposing a modular augmentation-based framework comprising reinforcement learning, contrastive learning, and test-time training.

While prior works explore augmentation and adversarial learning individually, this project contributes the first structured evaluation of,

1. **Reinforcement learning for augmentation selection** in deepfake detection.
2. **Contrastive learning integration** to stabilize feature representations across augmentations.
3. **Lightweight test-time adaptation** using KL-divergence regularization, for handling unseen inputs in real-time without retraining.

The research tests these components across three staged model versions (A, B, and C), allowing individual and combined impact analysis. Although RL-only (Version A) yielded the best balance of accuracy and performance, Versions B and C revealed improvements in feature sensitivity, adaptability, and Grad-CAM-based interpretability contributing valuable empirical insights into the trade-offs involved.

This project also contributes to broader domains such as **Cybersecurity and digital forensics** by offering a practical framework to enhance fraud and misinformation detection.

- **AI model adaptability**: Introducing dynamic augmentation and lightweight test-time adaptation mechanisms.
- **Ethical and explainable AI**: Supporting visual explanation and safe inference workflows under resource-constrained conditions.

1.7.2 Contribution to the problem domain

This research addresses a pressing challenge in deepfake detection, which is the limited ability of models to generalize across novel or unseen manipulation techniques. By introducing an adaptive, augmentation-driven framework that incorporates RL, CL, and TTT, this project offers a practical solution capable of enhancing robustness and reducing dependency on dataset-specific patterns.

The modular evaluation structure further contributes to industry-relevant understanding of where each technique excels or underperforms, which will provide a blueprint for future development of adaptable and interpretable deepfake detection systems suitable for real-world deployment.

1.8 Research Challenge

1.8.1 Achieving cross-dataset generalization without overfitting

One of the major challenges in deepfake detection is ensuring that the model generalizes well across different datasets while maintaining high accuracy. Many existing models perform well on the datasets they are trained on but fail when tested on unseen deepfakes due to overfitting(Lin *et al.*, 2024). This occurs because models learn dataset-specific features instead of recognizing deepfake-specific patterns that are independent of the training data.

To address this, the model must be trained with diverse augmentation techniques to improve robustness while avoiding overfitting. However, there is a delicate balance—over-reliance on augmentation can introduce artificial patterns that do not exist in real-world deepfake scenarios, reducing overall model effectiveness (Gupta *et al.*, 2023). Finding the right augmentation strategies that enhance adaptability without distorting real-world features is a crucial challenge.

1.8.2 Adapting to emerging deepfake manipulation techniques

Deepfake technology is rapidly evolving, with newer, more sophisticated techniques emerging frequently. Traditional detection models struggle to keep up because they rely on static feature extraction methods that do not adapt well to new forms of deepfake attacks (Chen *et al.*, 2022).

To effectively counter novel deepfake techniques, the detection model must incorporate reinforcement learning-based augmentation selection and test-time training (TTT). However, implementing real-time adaptability poses a challenge, as it requires the model to continuously learn from new deepfakes without significant retraining overhead. This makes it essential to develop a low-cost, efficient strategy for ongoing adaptation while maintaining high detection accuracy.

1.8.3 Managing computational complexity without compromising performance

Applying extensive data augmentation and adversarial training to improve deepfake detection requires high computational resources. Models incorporating contrastive learning, reinforcement learning-based augmentation, and test-time adaptation demand more processing power, which can lead to slower inference speeds (Stanciu and Ionescu, 2024).

This presents a challenge in real-time applications, where deepfake detection must occur quickly and efficiently. Designing a model that balances high accuracy with computational efficiency is crucial. Implementing hardware optimization techniques such as quantization, model pruning, and GPU acceleration can help, but these must be carefully tuned to ensure that they do not degrade the detection performance.

1.8.4 Developing a reliable evaluation framework for generalization

Deepfake detection models are often evaluated on benchmark datasets, but these benchmarks do not always reflect real-world conditions. Many existing detection models achieve high accuracy in controlled testing environments but fail when deployed in realistic, manipulated media scenarios (Masood *et al.*, 2023).

A major challenge is designing an evaluation framework that accurately measures the model's generalization ability across multiple deepfake generation techniques and real-world datasets. This requires incorporating cross-dataset validation, adversarial robustness testing, and real-time stress testing. Without a strong evaluation framework, it is difficult to quantify improvements and fine-tune the model for deployment.

1.8.5 Addressing ethical and security concerns in deepfake detection

As deepfake detection technology advances, there are ethical and security considerations that must be addressed. False positives (incorrectly flagging real content as deepfake) can have serious consequences, such as damaging reputations or suppressing legitimate content. On the other hand, false negatives (failing to detect actual deepfakes) can lead to misinformation, fraud, and security threats (Talreja *et al.*, 2024).

Ensuring that the model is fair and unbiased across different demographics is also a challenge. Many deepfake datasets lack diversity, which can introduce bias in detection models, making them less effective for certain groups. This requires careful dataset selection, fairness-aware training strategies, and ongoing bias evaluation to prevent ethical issues.

1.9 Research questions

RQ1: What are the recent advancements and challenges in achieving cross-dataset generalization in existing deepfake detection?

RQ2: How can reinforcement learning–based augmentation improve generalization across diverse datasets and manipulation techniques?

RQ3: What is the individual and combined impact of contrastive learning and test-time training on the robustness, generalization, and explainability of deepfake detection systems?

1.10 Research aim

The aim of this research project is to design, develop, and evaluate a modular deepfake detection framework that integrates reinforcement learning, contrastive learning, and test-time training to explore their individual and combined effects on detection generalization, robustness, and real-world usability.

1.11 Research objectives

Objectives	**Research Objective**	**Description**	**LOs Mapped**	**RQ Mapped**
problem identification	RO1: Conduct a preliminary study on existing approaches	Read, understand, and assess the literature to collect important information.	LO4, LO5, LO6	RQ1
	RO2: Explore Deepfake generation techniques and the current scope of Deepfake detection.	Study Deepfake creation techniques to learn about manipulation strategies and evaluate the advantages and disadvantages.	LO4, LO5	RQ1
Literature review	RO3: Investigate the role of reinforcement learning in adaptive	Identify requirements for augmentation methods to improve the performance	LO4, LO5, LO6	RQ1, RQ2

	augmentation for deepfake detection.	and flexibility of the model on various datasets.		
	RO4: Analyze existing adversarial augmentation and test-time adaptation techniques.	Get detailed insight into improving generalization in Deepfake detection	LO4, LO5	RQ1, RQ2
Requirement elicitation	RO5: Define requirements for a reinforcement learning-based augmentation strategy.	Evaluate project requirements, including needs for improving model generalization capabilities.	LO1, LO3	RQ2
	RO6: Gather various Deepfake datasets for model training and evaluation	Collect datasets from multiple domains (FaceForensics++, DFDC, Celeb-DF, etc.)	LO1, LO3, LO4	RQ2, RQ3
	RO7: Get insights from technology and domain experts.	Conduct interviews and learn from subject-matter and technology specialists.	LO3, LO4, LO6	RQ2, RQ3
System design	RO8: Design a multi-stage deepfake detection architecture integrating reinforcement learning, adversarial augmentation, and contrastive learning.	Develop an end-to-end model design that learns generalizable features across datasets.	LO1, LO7	RQ2, RQ3
	RO9: Evaluate and select reinforcement learning models	Compare policy gradient approaches and identify the best reinforcement learning	LO1, LO4, LO7	RQ2, RQ3

	(PPO/DQN) for augmentation optimization.	method for dynamically selecting augmentations.		
	RO10: Design a test-time adaptation module for real-time deepfake detection.	Implement a model adaptation mechanism that allows test-time fine-tuning.	LO1, LO2, LO7	RQ2, RQ3
	RO11: Develop a contrastive learning framework for feature consistency across datasets.	Ensure that feature representations remain consistent across different deepfake datasets to improve transferability.	LO1, LO2, LO7	RQ3
Implementation	RO12: Develop and implement the RL-augmented deepfake detection model.	Integrate reinforcement learning, adversarial augmentation, and test-time adaptation.	LO1, LO5, LO7	RQ2
	RO13: Build and integrate a prototype for demonstrating real-time deepfake detection capabilities	Deploy a prototype that demonstrates the impact of adaptive augmentation on unseen data.	LO1, LO7	RQ3
Testing	RO14: Define quantitative metrics for evaluating generalization effectiveness.	Establish evaluation metrics, such as cross-dataset accuracy and recall.	LO4, LO7	RQ3
	RO15: Conduct extensive testing across different datasets and	Conduct thorough testing to verify the robustness of the model and make sure that generalization is	LO4, LO7, LO9	RQ3

	manipulation techniques	accomplished successfully over a variety of deepfake variations.		
Documentation	RO16: Document the continuous progress of the research project, including any obstacles and insights gained.	Maintain detailed records of every phase of the project, including problems, fixes, and updates.	LO6, LO8	-
Publication	RO17: Prepare research findings for academic publication	Combine research contributions, performance results, and insights into a comprehensive research paper for submission to a recognized academic journal.	LO4, LO6, LO8, LO9	RQ1, RQ2, RQ3

Table 01: Research Objectives

1.12 Chapter summery

This chapter outlined the background and motivation for the study, identified the key research gap in cross-dataset generalization, and presented the project's aim, contributions, challenges, and objectives. The study introduces a modular detection framework evaluated in three versions (RL, RL+CL, RL+CL+TTT) to explore the individual and combined impact of these techniques on deepfake detection performance.

CHAPTER 02: LITERATURE REVIEW

2.1 Chapter overview

This chapter reviews existing research on Deepfake detection, focusing on algorithmic approaches, benchmark datasets, evaluation metrics, and key challenges in enhancing model performance and generalization.

2.2 Concept map

A concept map outlining the key themes, technologies, and challenges explored in this literature review is provided in ***APPENDIX 02***.

2.3 Challenges in generalized deepfake detection

Despite significant advancements in deepfake detection, existing systems struggle with generalization as they perform well on known datasets but failing against novel manipulations or previously unseen techniques(Chen *et al.*, 2022).

This challenge arises primarily because most detection models exhibit high accuracy when tested on datasets that closely resemble their training data. However, their performance drops significantly when exposed to deepfakes generated using unseen techniques or subjected to different post-processing modifications, such as compression artifacts, adversarial perturbations, and noise injection (Dong *et al.*, 2023).

This lack of generalization arises due to:

1. **Overfitting to dataset-specific artifacts** – Many deepfake detectors learn superficial patterns or dataset-specific noise instead of extracting robust deepfake-specific features (Chen et al., 2022). This results in models failing when applied to real-world deepfakes that do not exhibit the same artifacts (Stanciu and Ionescu, 2024).
2. **Rapid evolution of deepfake generation techniques** – The continuous improvements in GAN-based architectures and diffusion models enable more advanced forgeries that bypass traditional detection systems (Dong *et al.*, 2023). Models trained on older datasets become ineffective against new, previously unseen deepfake styles.

3. **Limited and static augmentation strategies** – Many existing detection frameworks rely on predefined augmentations that fail to simulate real-world deepfake variations. This reduces their ability to adapt dynamically to new attack methodologies (Chen *et al.*, 2022; Dong *et al.*, 2023).

To effectively combat deepfake threats, there is a critical need for detection algorithms that can:

- Generalize well across multiple datasets by learning robust, domain-invariant features.
- Detect deepfakes manipulated using unknown post-processing techniques.
- Adapt dynamically to emerging deepfake generation methods.

By addressing these critical gaps, future deepfake detection systems can become more resilient, adaptable, and suitable for real-world deployment (Chen *et al.*, 2022; Stanciu and Ionescu, 2024).

2.4 Existing work

Even though deepfake detection has made much progress, maintaining consistent cross-dataset performance continues to be a difficult issue which demonstrates an important demand for new strategies including state-of-the-art data augmentation to further improve generalization. The surveyed sources study a broad area of Deepfake detection methods, limitations and potential future direction.

2.4.1 Recent advancements and challenges in cross-dataset generalization

A key challenge in Deepfake detection is the limited generalizability/transferability of trained models across datasets (Hu, Wang and Li, 2021; Chen et al., 2022; Wang et al., 2022; Dong et al., 2023). Models trained on one dataset often perform poorly on others, due to variations in deepfake generation methods, post-processing techniques, and dataset characteristics (Jain, Korshunov and Marcel, 2021; Korshunov and Marcel, 2022; Wang et al., 2022). For instance, detectors trained on datasets with specific artefacts might fail when tested with deepfakes from unseen sources or those with different post-processing (Chen et al., 2022; Wang et al., 2022; Dong et al., 2023; Stanciu and Ionescu, 2024).

Generalization has been severely limited by this issue, which results from variations in identities and creation techniques within datasets. (Guan et al., 2024; Lin et al., 2024; Yin et al., 2024). This gap arises because models often learn domain-specific features rather than focusing on universal deepfake characteristics (Guan et al., 2024; Yin et al., 2024). As a result, detectors struggle to generalize to unseen domains, particularly those with novel deepfake generation techniques (Korshunov, Jain and Marcel, 2022; Yin et al., 2024).

The rapid evolution of deepfake generation techniques presents a constant challenge for detectors (Masood et al., 2023; Guan et al., 2024; Stanciu and Ionescu, 2024; Yin et al., 2024). Deepfake techniques create increasingly lifelike deepfakes that are more difficult to tell apart from authentic information as they advance in sophistication. (Masood et al., 2023). This demands continuous development of new detection approaches that can adjust to new Deepfake generation techniques (Wang et al., 2022; Masood et al., 2023; Stanciu and Ionescu, 2024).

2.4.2 Existing deepfake detection methods

1. **Handcrafted features:** These techniques focus on identifying particular inconsistencies or faults present in Deepfakes, like head posture, eye blinking, and visual artefacts. But the usefulness of these techniques reduces when deepfake technology develops and successfully hides such objects. (Wang et al., 2022; Masood et al., 2023).
2. **Deep learning approaches:** Deepfake detection accuracy has increased thanks to DNN-based techniques, which have outperformed manually created feature-based methods (Yin et al., 2021). These techniques employ a variety of network designs, including CNNs, RNNs, and Capsule Networks to train discriminative features (Jain, Korshunov and Marcel, 2021; Nadimpalli and Rattani, 2022; Gupta et al., 2023; Masood et al., 2023; Smelyakov, Kitsenko and Chupryna, 2024). Popular pre-trained models that are often used for feature extraction are VGG16, ResNet50, and XceptionNet (Jain, Korshunov and Marcel, 2021; Wang et al., 2022; Smelyakov, Kitsenko and Chupryna, 2024). However, these models still struggle to generalize new data and they often overfit training data (Jain, Korshunov and Marcel, 2021; Korshunov, Jain and Marcel, 2022; Nadimpalli and Rattani, 2022).
3. **Focus on specific artefacts:** Some methods target specific artefacts introduced by certain deepfake generation methods, such as blending boundaries, facial inconsistencies, or frequency-domain artefacts (Chen et al., 2022; Wang et al., 2022; Dong et al., 2023). Still,

these methods lack generalizability to unseen deepfake methods that do not display these specific artefacts (Chen et al., 2022).

2.4.3 Novel augmentation-based algorithm for enhanced generalization

The sources highlight data augmentation as a promising path for improving the generalization of Deepfake detection models. The hypothesis that novel data augmentation techniques can significantly enhance generalization capabilities is supported by several studies.

1. **"Data farming":** This technique, proposed by Korshunov and Marcel (2022), involves merging randomly cropped background patches with enhanced faces to create authentic samples. Instead of depending on facial traits, this method requires the model to learn Deepfake-specific artefacts, which could enhance generalization.
2. **Adversarial attacks:** Stanciu and Ionescu (2024) present a framework that creates fresh deepfakes from real photos using adversarial approaches. This is done through improving the generalization of hidden deepfakes by including these artificial deepfakes (these are specifically created to challenge the detector) in the training set.
3. **Recurrent adversarial training:** Stanciu and Ionescu (2024) further suggest a recurrent training procedure that improves generalization by continuously training the detector on newly created adversarial deepfakes.

2.4.4 Optimizing augmentation techniques for improved generalization

1. **Multi-scale facial swap:** Dong et al. (2023) introduce Multi-Scale Facial Swap (MFS), a technique for augmenting data. The Implicit Identity Leakage (IIL) issue, in which models unintentionally pick up identity representations that limit generalization, is lessened by this method.
2. **Domain Refinement Data Augmentation (DRDA):** Yin et al. (2024) propose DRDA, which uses self-face-swapping and region-erasing techniques to suppress domain-specific features, encouraging the model to focus on domain-invariant characteristics. The goal of this method is to increase generalization across datasets from different areas.

2.4.5 Summary of existing work and connection to current research

Deepfake detection has advanced significantly in recent years; however, methods that overcome current limitations; particularly poor generalization across unseen datasets yet remain largely unaddressed in existing research. Existing literature highlights several promising avenues,

- **Reinforcement Learning-based augmentation:** Existing static augmentation methods (Korshunov and Marcel, 2022; Dong et al., 2023) are limited in their ability to adapt dynamically. Reinforcement learning (RL), specifically using methods like PPO, has proven effective in dynamically selecting augmentations in related domains, suggesting a potential improvement in robustness and adaptability.
- **Contrastive Learning for feature consistency:** Current detection approaches suffer from domain bias and identity leakage (Dong et al., 2023; Yin et al., 2024). Contrastive learning effectively encourages domain-invariant representations, enhancing generalization across diverse datasets and manipulation techniques.
- **Test-Time Training (TTT) for real-time adaptability:** While existing adversarial augmentation and recurrent training (Stanciu and Ionescu, 2024) demonstrate improved robustness, no current methods explicitly address real-time model adaptation without costly retraining. TTT offers a promising lightweight mechanism for on-the-fly adaptability, leveraging minimal data during inference.

While the literature highlights the individual potential of RL, CL, and TTT for improving generalization, no existing solution has unified these techniques into a single modular framework. This project addresses that gap by experimentally evaluating their isolated and combined effects on generalization and adaptability through the GenDF system which will be directly contributing to the research domain with new insights and practical implementation.

2.5 Technological review

Deepfake detection relies on a combination of machine learning algorithms, preprocessing techniques, and evaluation strategies to achieve high accuracy and generalization. This section reviews the key technological components used in state-of-the-art deepfake detection, including programming languages, machine learning models, preprocessing techniques, hyperparameter tuning, and evaluation strategies.

2.5.1 Programming Languages and libraries

Python is the dominant programming language used for deepfake detection due to its extensive support for machine learning and computer vision libraries (Korshunov, Jain & Marcel, 2022; Gupta et al., 2023). The primary deep learning frameworks include:

TensorFlow – A highly scalable and widely used open-source library for deep learning applications (Gupta et al., 2023; Masood et al., 2023).

Keras – A high-level API that simplifies deep learning model development, often used alongside TensorFlow (Masood et al., 2023).

These libraries provide pre-trained models, GPU acceleration, and extensive support for deepfake detection research, making them ideal for handling large-scale datasets.

2.5.2 Machine Learning algorithms

Deepfake detection primarily relies on neural networks, though other machine learning approaches are occasionally used. The most common architectures include,

1. Convolutional Neural Networks (CNNs)

CNNs are widely used for spatial feature extraction, identifying manipulation artifacts in deepfake images and videos (Gupta et al., 2023; Smelyakov, Kitsenko & Chupryna, 2024).

Common architectures:

- XceptionNet (Jain et al., 2021; Korshunov & Marcel, 2022).
- EfficientNet (Smelyakov et al., 2024).
- ResNet (Gupta et al., 2023).

2. Recurrent Neural Networks (RNNs)

RNNs process sequential data, making them effective for video-based deepfake detection by analyzing temporal inconsistencies (Gupta et al., 2023; Smelyakov, Kitsenko & Chupryna, 2024).

3. Capsule networks

Capsule Networks capture spatial hierarchies between features, enhancing robustness against adversarial attacks in deepfake detection (Jain, Korshunov & Marcel, 2021).

4. Decision Trees (DTs)

DT-based models provide interpretable deepfake detection but generally lack the accuracy needed for complex datasets (Humbird, Peterson & McClarren, 2017).

5. Hybrid models

Combining CNNs with handcrafted feature detection or decision trees can improve detection accuracy (Wang et al., 2022; Gupta et al., 2023).

2.5.3 Specific deepfake detection techniques

The various feature extraction techniques to differentiate real from fake images and videos,

1. Artifact-based detection

Detects anomalies introduced by deepfake generation, such as:

- Eye blinking inconsistencies – Deepfake videos often display unnatural blinking patterns (Dong et al., 2023).
- Head pose estimation – Misalignment between facial expressions and head movements (Wang et al., 2022).
- Visual artifact analysis – Blurring, unnatural lighting, and pixel inconsistencies (Masood et al., 2023).

2. Frequency-domain analysis

Examines the frequency spectrum of images/videos to detect deepfake-specific artifacts that are not visible in the spatial domain (Wang et al., 2022).

3. Data augmentation for generalization

Techniques such as adversarial attacks, data farming, and recurrent adversarial training improve model robustness against previously unseen deepfakes (Gupta et al., 2023).

2.5.4 Datasets and preprocessing

2.5.4.1 Datasets

Publicly available deepfake datasets play a crucial role in model training and evaluation (Wang et al., 2022). The most widely used datasets include:

- FaceForensics++ (FF++) – 5,000 deepfake videos using multiple generation methods (DeepFakes, FaceSwap, Face2Face, NeuralTextures).
- Celeb-DF–5,639 deepfake videos of celebrity impersonations.
- DeeperForensics-1.0(DF1.0) – 60,000 high-quality deepfake videos.
- Deepfake Detection Challenge (DFDC) – Over 100,000 fake videos.
- WildDeepfake (WDF) – Real-world deepfake content collected from the internet.

These datasets vary in generation techniques, compression levels, and resolution, making them ideal for evaluating model generalization.

2.5.4.2 Preprocessing techniques

Preprocessing is essential for enhancing training efficiency and accuracy (Gupta et al., 2023). Common techniques include:

- Face extraction – Uses face detection algorithms to crop and align facial regions for analysis.
- Data augmentation – Expands training data diversity via random cropping, color distortions, and adversarial transformations.
- Filtering and cleaning – Remove low-resolution or low-quality images to ensure consistent dataset quality.

2.5.5 Algorithm selection

Selecting the optimal detection model depends on several factors,

- Generalization Ability – How well the model performs on unseen datasets (Korshunov & Marcel, 2022).
- Dataset Characteristics – Matching model complexity to dataset size and diversity (Korshunov & Marcel, 2022).

- Computational Resources – Ensuring the model can run efficiently without excessive GPU demand.

For instance, CNN-based architectures (XceptionNet, EfficientNet) are ideal for image-based deepfake detection, while RNNs and Transformer models are more effective for video-based deepfakes (Gupta et al., 2023).

2.5.6 Hyperparameter Tuning

Hyperparameter tuning plays a critical role in optimizing deepfake detection models. Literature suggests the following best practices:

- Learning Rate – Typically set between 10^{-3} and 10^{-5} or 0.0002 for models like Xception and EfficientNet (Stanciu & Ionescu, 2024).
- Epochs – Models are usually trained for 20+ epochs, with early stopping to prevent overfitting (Jain, Korshunov & Marcel, 2021).
- Batch Size – Varies between 32 and 48, depending on dataset and model architecture (Yin et al., 2024).
- Optimizers – Adam optimizer is commonly used, with or without learning rate decay (Yin et al., 2024).
- Loss Functions – Cross-entropy loss, triplet loss, and ArcFace loss are commonly explored for improved performance (Guan et al., 2024).

Fine-tuning these parameters significantly impacts model accuracy, generalization, and training stability.

Deepfake detection relies on multiple technological components, including deep learning architectures, preprocessing techniques, and dataset selection strategies. While CNNs, RNNs, and Capsule Networks have shown promise, overfitting and dataset biases remain key challenges. Advanced augmentation techniques and robust hyperparameter tuning are crucial for improving cross-dataset generalization and real-world applicability.

2.6 Benchmarking and evaluation

Evaluating the performance of deepfake detection systems requires robust benchmarking using standard datasets and quantitative performance metrics. The literature consistently emphasizes the need for both in-dataset and cross-dataset testing to assess real-world generalizability (Korshunov, Jain & Marcel, 2022; Yin et al., 2024).

2.6.1 Evaluation Metrics

Several core metrics are used to evaluate model classification performance.

1. **Accuracy:** Measures the percentage of correctly predicted samples. While useful, it can be misleading in imbalanced datasets.
2. **Precision and Recall:** Precision measures how many predicted deepfakes are actually fake, while recall measures how many actual deepfakes were detected.
3. **F1 Score:** The harmonic mean of precision and recall; used to balance false positives and false negatives.
4. **Area Under the Curve (AUC):** Indicates the model's ability to distinguish between classes across various thresholds.
5. **Half Total Error Rate (HTER):** A more holistic metric combining False Acceptance Rate (FAR) and False Rejection Rate (FRR) into a single number, providing insight into the trade-off between type I and type II errors (Korshunov, Jain & Marcel, 2022).

These metrics provide complementary views of performance and are essential in high-stakes applications such as legal forensics or media verification.

2.6.2 Benchmark Datasets and Cross-Dataset Evaluation

Benchmarking deepfake detection models relies heavily on widely accepted datasets such as:

- FaceForensics++ (FF++)
- Celeb-DF
- DFDC
- WildDeepfake (WDF)

These datasets vary in compression levels, deepfake generation methods, and video quality, offering comprehensive coverage for evaluating model robustness (Yin et al., 2024). Evaluations using cross-dataset testing, where models are trained on one dataset and tested on another, are widely used to assess generalization performance (Korshunov, Jain & Marcel, 2022).Cross-dataset benchmarking is especially important as it simulates real-world scenarios where deepfakes encountered are not from the same distribution as training samples.

2.6.3 Adversarial Robustness and Generalization Evaluation

Recent studies evaluate models not just on traditional test sets but also on adversarially augmented data to test resilience (Stanciu & Ionescu, 2024). This includes,

- Data farming
- Recurrent adversarial training
- Test-time training (TTT)

These techniques are used in generalization benchmarking pipelines, where models are exposed to diverse deepfake variants to measure adaptability beyond standard datasets.

2.6.4 Explainability and Statistical Validation

In addition to detection performance, explainability is becoming an important evaluation criterion. Techniques from Explainable AI (XAI) allow interpretation of model decisions, building trust and transparency in critical use cases (Wang et al., 2022; Gupta et al., 2023).Moreover, statistical validation techniques like confidence intervals, random sampling, and stratified cross-validation are recommended to ensure reliability and reduce bias, particularly in sensitive domains such as legal investigations, journalism, and cybersecurity (Wang et al., 2022).

2.7 Chapter Summary

This chapter examined key deepfake detection approaches and highlighted challenges in generalization due to dataset bias and limited adaptability. This review highlights key gaps in adaptability, augmentation, and generalization, providing the foundation for experimentally evaluating RL, CL, and TTT within the GenDF framework and justifying its modular design and real-world benchmarking.

CHAPTER 03: METHODOLOGY

3.1 Chapter overview

This chapter describes the methodological process used to design, build, and evaluate a deepfake detection system that generalizes across datasets. It includes details on data handling, model experimentation, evaluation criteria, and iterative refinement. The research proceeds through three progressive system versions, each integrating advanced techniques; Reinforcement Learning, Contrastive Learning, and Test-Time Training to evaluate their individual and combined impact on generalization.

3.2 Research methodology (Saunders's research onion in Table Form)

Layer	Choice	Justification
Philosophy	Pragmatism (quantitative and qualitative data collection)	Research on deepfake detection mostly depends on quantifiable, objective data that is acquired by evaluating models on benchmark datasets. The objective is to use metrics like accuracy, AUC, and precision to measure these models' performance (Korshunov and Marcel, 2022; Korshunov, Jain and Marcel, 2022; Gupta et al., 2023) but qualitative input will also have to be taken.
Approach	Deductive (Hypothesis Testing)	This research tests whether new data augmentation techniques can improve the generalization of Deepfake detection models. By applying established theories in computer vision and machine learning, the author will be validating the constructed hypothesis through experiments, assessing how well the enhanced models perform across varied datasets.

Methodological Choice	Mixed methods (Quantitative and Qualitative)	Although discipline is dominated by quantitative statistics, qualitative insights is also useful (Wang et al., 2022). And since the philosophy chosen is Pragmatism the methodological choice should be mixed methods either way.
Strategy	Experimental (Model testing and evaluation)	Research on deepfake detection requires extensive testing, with multiple models being trained and assessed on diverse datasets. Controlled experiments that manipulate variables such as training data, model design, and data augmentation methods will be done.
Time Horizon	Cross-sectional	Data is analyzed at a single time point, using snapshots of current datasets for testing and comparing model accuracy.
Data Collection	Secondary data (Use of publicly available datasets)	Although benchmark datasets such as FaceForensics++, Celeb-DF, and DFDC were initially considered, the final dataset selected was the “Real and Fake Face Detection” dataset from Kaggle due to its simplicity, accessibility, and compatibility

Table 02: Research Methodology

3.3 Development methodology

This section outlines the overall development approach, and the supporting methodologies used to design, build, test, and refine the proposed deepfake detection system. The selected methodology is Prototyping, as it best supports the iterative experimentation required for this research.

3.3.1 Development methodology

The Prototyping development methodology is adopted for this project due to its alignment with the experimental and iterative nature of deepfake detection research. This approach supports the creation of incremental prototypes, allowing the researcher to iteratively improve model generalization by experimenting with various data augmentation techniques, model architectures, and evaluation strategies.

Compared to waterfall: Waterfall follows a linear process that lacks flexibility. Given the nature of this project, which requires frequent refinement based on evaluation results, a rigid approach would limit adaptability.

Compared to agile: While Agile supports iteration, it is more suited to feature-driven development for commercial products. This project is research-oriented, where model performance and experimentation take precedence over stakeholder-driven requirements.

Prototyping on the other hand enables rapid development, evaluation, and enhancement of model prototypes. Each version builds upon feedback and test results, which is ideal for incorporating methods such as:

- Reinforcement learning-based augmentation
- Test-time training (TTT)
- Cross-dataset evaluation
- Contrastive learning

3.3.2 Requirement elicitation methodology

Multiple elicitation techniques were used to define both functional and non-functional requirements:

- **Surveys:** Conducted to assess public understanding of deepfakes and gather feedback on usability.
- **Document analysis:** Academic literature and documentation from existing deepfake detection frameworks were analyzed to identify gaps and define technical benchmarks.
- **Prototyping:** Early versions of the system informed iterative refinements.
- **Brainstorming:** Sessions with peers and stakeholders were held to anticipate system needs and trends in deepfake generation.

This combination ensured that both technical feasibility and user relevance were considered in defining system requirements.

3.3.3 Design methodology

The project adopts the Object-Oriented Analysis and Design Methodology (OOADM) to structure the system architecture. OOADM was chosen due to its:

- Suitability for modular systems like deepfake detection pipelines
- Support for scalability as new detection features are added
- Ability to promote reusability and maintainability through encapsulated components.

System components such as preprocessing, augmentation selection, feature extraction, and prediction were designed as discrete, interoperable objects.

3.3.4 Programming paradigm

The Object-Oriented Programming (OOP) paradigm was selected to complement the OOADM design methodology. Python was used as the primary language due to its strong OOP capabilities and extensive support for deep learning libraries such as:

- PyTorch (for model development)
- TensorFlow and Keras (for baseline testing)
- Albumentations and OpenCV (for data augmentation and image processing)

OOP facilitated modular, testable code, which is crucial for building and evaluating successive prototypes efficiently.

3.3.5 Testing methodology

Model testing

Quantitative performance testing will be conducted using the following metrics:

- Accuracy, Precision, Recall, F1 Score, AUC, HTER
- Cross-dataset evaluation to assess generalization capability
- Robustness testing with manipulated inputs (noise, blur, lighting changes)

Although larger benchmarks were considered, testing was ultimately conducted using the "Real and Fake Face Detection" dataset from Kaggle. While smaller in size, it enabled rapid prototyping and multiple model version evaluations within Colab's constraints.

Prototype testing

- **Integration testing:** Verifies interactions between key modules (preprocessing, augmentation, inference).
- **Usability testing:** Ensures interface elements (e.g., visual result outputs) are intuitive and effective.
- **Stress testing:** Assesses system behavior under resource-intensive conditions.

3.3.6 Solution development workflow

The overall development pipeline followed these stages:

1. Dataset collection

Public datasets like FaceForensics++, Celeb-DF, and DFDC were used due to their diversity in manipulation techniques and real-world applicability.

2. Preprocessing and augmentation

- Standardized image resolutions
- Pixel normalization
- Augmentations: Gaussian noise, brightness adjustments, flipping, random rotations

3. Feature Engineering

Focused on identifying texture artifacts, facial inconsistencies and Edge distortions.

4. Model Selection

A hybrid model combining Convolutional Neural Networks (CNNs) and attention mechanisms was chosen to capture both local and contextual patterns in visual data.

5. Model Training

Training conducted over 20+ epochs with early stopping to prevent overfitting. Optimizer: Adam, with learning rate tuning between 0.0001 – 0.00001.

6. **Evaluation**

Models were tested on seen and unseen datasets, and under different manipulation intensities to evaluate both accuracy and generalization.

7. **Feedback loop**

Performance results informed refinements to augmentation strategies, hyperparameters, and model components in subsequent prototypes.

3.4 Project management methodology

This project adopts a hybrid Agile PRINCE2 methodology, which combines the structured project control mechanisms of PRINCE2 with the flexibility and adaptability of Agile. This hybrid approach is particularly well-suited for research and development projects like this, where experimentation, iterative refinement, and continuous feedback are critical.

By leveraging Agile's iterative development cycles, the project supports rapid prototyping and model validation, while PRINCE2 ensures clearly defined stages, milestone tracking, and controlled delivery of key outputs such as the PPRS, IPD, and FYP documentation.

3.4.1 Project scope

3.4.1.1 In-scope

The following elements are included within the scope of this project,

- Collection of publicly available deepfake datasets (Kaggle's "Real and Fake Face Detection" dataset was used; benchmark datasets such as FaceForensics++ and DFDC were considered but excluded due to video format constraints)
- Detection is limited to visual deepfakes involving **frontal human facial portraits**, as defined by the datasets used
- Data preprocessing and augmentation
- Modular model development using CNNs with reinforcement learning-based augmentation, contrastive learning, and test-time training

- Evaluation across multiple datasets to measure cross-dataset generalization
- Feedback loop for model refinement (based on test results)
- Integration of explainability techniques such as Grad-CAM for model transparency
- Domain expert interviews and user surveys for usability and perception evaluation
- Development of system documentation and delivery of interim and final demonstrations

3.4.1.2 Out-of-scope

The following are explicitly excluded from the project scope:

- Integration with real-time surveillance or third-party production systems
- Optimization for low-power or mobile devices
- Use of sensitive/private/non-public data sources
- Real-time video stream analysis or detection in moving scenes
- Audio deepfakes, full-body manipulation, or non-visual deepfake content
- Commercial deployment or monetization of the prototype

3.4.2 Schedule and planning

The project is structured around university-defined milestone submissions, interspersed with internal development cycles guided by Agile sprint-like reviews. Major components include requirement gathering, prototyping phases, testing, documentation, and viva preparation.

3.4.2.1 Gantt chart

A detailed Gantt chart that illustrates the full timeline, milestones, and dependencies is available in ***APPENDIX 03***

.

3.4.2.1 Key Deliverables and dates

Deliverable	Description	Deadline
Project Proposal	Initial requirements, project planning and risk analysis	11th November 2024
PPRS – Project Planning and Research Specification	Scope definition, initial literature review, and methodology planning	3rd February 2025
IPD – Interim Progress Demonstration	Prototype, early testing results, and presentation	1st April 2025
FPR and FPD – Final Project Report and Demonstration	Final model, evaluation, full report submission	7th April 2025
FPV – Final Project Viva	Presentation and defence of project	During viva window: April–May 2025

Table 03: Project deliverables and timeline

3.5 Resource requirements

The successful implementation of this research project relies on several hardware, software, data, and technical resources. Each category is outlined below:

3.5.1 Hardware resources

Training deep learning models on large datasets requires access to high-performance computing (HPC) resources. The key hardware requirements include:

- GPU-enabled machine (NVIDIA RTX 3060 or equivalent) for accelerated training of CNN-based models
- Minimum 16 GB RAM to handle high-resolution image processing and augmentation
- At least 250 GB SSD storage to accommodate multiple datasets, experiment logs, model checkpoints, and temporary augmentation outputs

Where required, cloud-based platforms such as Google Colab Pro or Kaggle Notebooks will be used for extended GPU access.

3.5.2 Software resources

The project will utilize the following software stack:

- **Programming language**: Python
- **Deep Learning libraries**:
    - TensorFlow and PyTorch – for model development and training
    - Keras – for rapid prototyping
- **Model evaluation**: Scikit-learn – for metrics such as AUC, precision, and recall
- **Data handling and preprocessing**: OpenCV and Albumentations – for image preprocessing and augmentation
- **Development environment**: Google Colab and Visual Studio Code for development and prototyping

3.5.3 Technical skills

1. Existing Skills
    - Python scripting, CNNs, image preprocessing, model evaluation metrics
2. New Skills Acquired
    - RL-based augmentation (PPO)
    - Test-Time Training (TTT)
    - Contrastive learning for domain-invariant features
    - XAI techniques (Grad-CAM)
    - Augmentation pipeline design for generalization

3.5.4 Data requirements

The following publicly available deepfake datasets are used for model training, validation, and benchmarking,

- FaceForensics++ – Offers manipulated videos generated using multiple deepfake methods
- Celeb-DF – High-quality celebrity deepfakes for cross-dataset testing
- Deepfake Detection Challenge (DFDC) – Large-scale dataset used for evaluating model generalization

These datasets provide diversity in manipulation techniques, compression levels, and face identities, ensuring comprehensive training and evaluation conditions.

3.6 Risk and mitigation

Risk	**Severity (1–5)**	**Frequency (1–5)**	**Mitigation Strategy**
Insufficient Data for Generalization	4	5	Use data augmentation to artificially expand dataset diversity; explore additional publicly available datasets for inclusion.
Poor Model Generalization Across Datasets	5	4	Implement cross-dataset training and evaluate models on varied datasets during each prototype iteration.
Overfitting	4	3	Use k-fold cross-validation, implement regularization (e.g., dropout, L2), and early stopping during training.
Hardware Constraints	3	5	Utilize cloud-based platforms (Google Colab, AWS) with GPU support; schedule training during off-peak hours to avoid resource throttling.
High False Positive/Negative Rates	4	4	Optimize model thresholds using ROC/AUC analysis; balance precision and recall through improved augmentation and training techniques.
Privacy and Ethical Concerns	5	2	Only use publicly available datasets; do not store or process any personal or sensitive user data; follow ethical AI practices.
Delays in Model Training	4	4	Optimize code and training pipelines; use batch processing and parallelism where possible to reduce training time.

Evolving Deepfake Techniques	5	4	Periodically re-train the model on newer datasets to adapt to novel deepfake generation methods.
Complexity in Implementation	3	3	Test new modules on small-scale subsets first
Lack of Real-World Robustness	4	5	Conduct robustness testing under diverse conditions (noise, compression, blur)

Table 04: Risks and mitigations

3.7 Chapter summary

This chapter outlined the methodological approach for designing and developing a deepfake detection system, focusing on iterative prototyping and experimental validation to improve cross-dataset generalization.

CHAPTER 04: SOFTWARE REQUIREMENTS SPECIFICATION

4.1 Chapter overview

This chapter presents a detailed analysis of the software requirements for the proposed deepfake detection system. It outlines the methodologies used to gather and analyze requirements, identifies key stakeholders and their viewpoints, and specifies both functional and non-functional requirements. Visual models such as the Rich Picture, Context Diagram, Stakeholder Onion Model, and Use Case Diagram are included to illustrate the system's structure and interactions. Finally, all requirements are prioritized using the MoSCoW method to ensure focus on critical functionalities essential to the system's performance and delivery.

4.2 Rich picture diagram

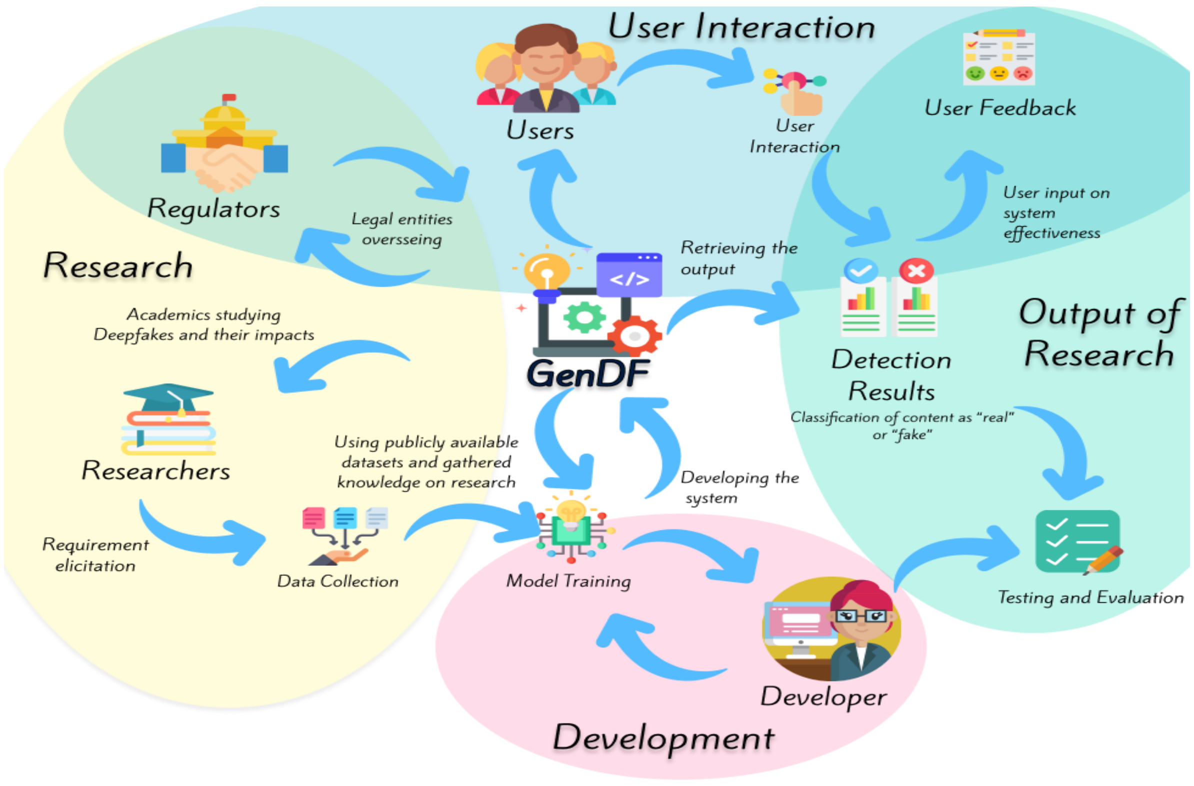


Figure 01: Rich Picture Diagram (Self-Composed)

This diagram illustrates the interaction between key stakeholders in the GenDF deepfake detection system. It shows how developers, researchers, users, and regulators contribute to and interact with the system across the stages of data collection, model development, output evaluation, and user feedback. It also highlights the research-driven and user-centric nature of the system, emphasizing iterative feedback loops and regulatory oversight.

4.3 Stakeholder analysis

The stakeholder analysis is done to properly identify the stakeholders of this research and their relationship with the research project.

4.3.1 Stakeholder onion model

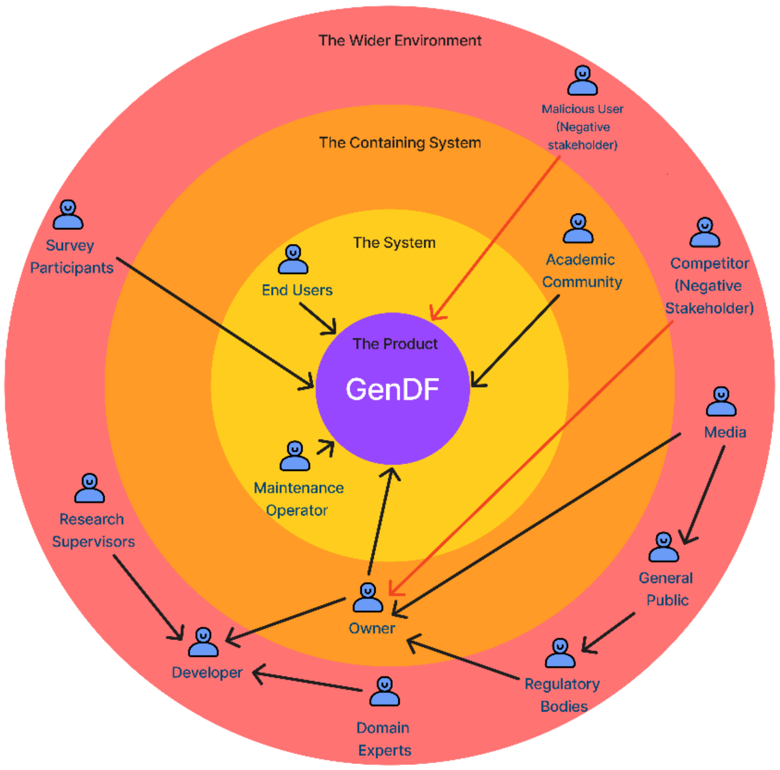


Figure 02: Stakeholder Onion Model Diagram (self-composed using Canva)

This diagram illustrates the layered relationship between GenDF and its internal, external, and environmental stakeholders, including system developers, users, regulators, and even potential adversaries.

4.3.2 Stakeholder viewpoints

Stakeholder	Viewpoints / Expectations
Developers	Requires efficient tools, datasets, and infrastructure to build a high-performing detection system with strong generalization.
Project owners	Expects a system that is innovative, academically impactful, and meets key performance metrics identified in the research gap.
Domain experts	Interested in a scientifically sound model that contributes to the field of AI-based media forensics, particularly generalizable detection.
Research Supervisors	Expect thorough documentation, clear methodology, and measurable contributions to academic knowledge.
Survey participants	Seek an intuitive, understandable system that raises awareness about deepfake risks and improves digital literacy.
Academic community	Focused on the novelty of the research approach (e.g., RL-based augmentation, contrastive learning) and its contributions to state-of-the-art detection methods.
End-users	Need a reliable tool that minimizes false positives/negatives and integrates seamlessly with existing media workflows.
General public	Concerned with the trustworthiness of online media and expect solutions that reduce misinformation and promote online safety.
Regulatory bodies	Require compliance with data privacy laws (e.g., GDPR), ethical AI standards, and transparent use of datasets.
Maintenance operator	Needs the system to be modular, maintainable, and well-documented to allow efficient updates and troubleshooting post-deployment.
Media	Interested in transparent and trustworthy insights into how the system identifies manipulated content in public discourse.

Competitor	May attempt to analyze or benchmark the system for competitive advantage; indirectly influences innovation and industry standards.
Malicious user	May try to bypass or mislead the detection model for misuse, fraud, or manipulation, representing an ongoing adversarial risk.

Figure 03: Stakeholder viewpoints

4.4 Selection of requirement elicitation methodologies

To ensure that the GenDF deepfake detection system meets both user needs and technical expectations, several requirement elicitation methodologies were considered. After evaluating their strengths, surveys were selected as the primary elicitation method, supported by document analysis, prototyping, and brainstorming sessions.

Selected method: Surveys

Surveys were chosen as the primary elicitation technique due to their ability to collect broad, quantifiable insights from a diverse group of stakeholders, including end users and members of the general public. This was crucial to understanding:

- User awareness of deepfake threats
- Expectations regarding detection accuracy
- Perceptions of trust and usability in AI-driven detection systems
- The demand for generalizability across different media types and manipulation methods

Justification over other methods:

- Compared to interviews, surveys provided wider reach, ensuring input from a more representative audience.
- Unlike document analysis, surveys captured real user opinions and subjective concerns which are not available in literature.
- Surveys were also quicker and easier to analyze at scale than focus groups, making them more practical within the project timeline.

Supporting methods:

- **Document analysis**

Used to extract functional and non-functional requirements from academic sources and system documentation of existing deepfake detection solutions. Helped identify technical challenges (e.g., overfitting, poor cross-dataset generalization) and ethical gaps (e.g., privacy, dataset bias).

- **Prototyping**

Early system prototypes helped validate assumptions and gather iterative feedback on detection performance and interface usability. This ensured requirements were grounded in real system behavior.

- **Brainstorming sessions**

Conducted with developers and academic mentors to collaboratively define technical features, such as adaptive augmentation techniques, UI features, and model evaluation strategies. These sessions generated actionable ideas that informed system design.

4.5 Findings from elicitation

For each methodology used, key findings are documented below to highlight insights gathered through various elicitation techniques and their relevance to the deepfake detection system's development.

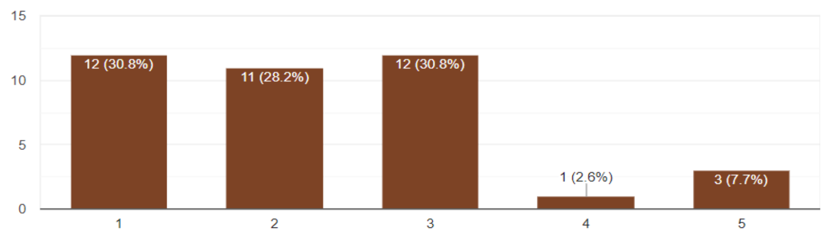
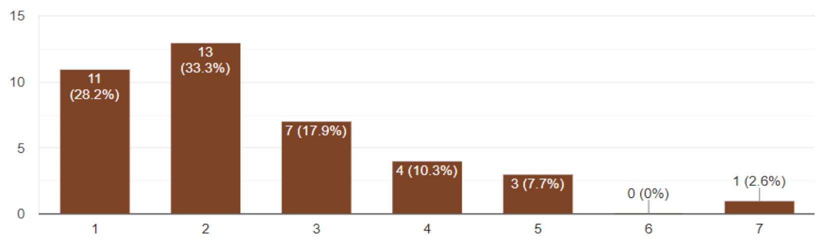
4.5.1 Literature review findings

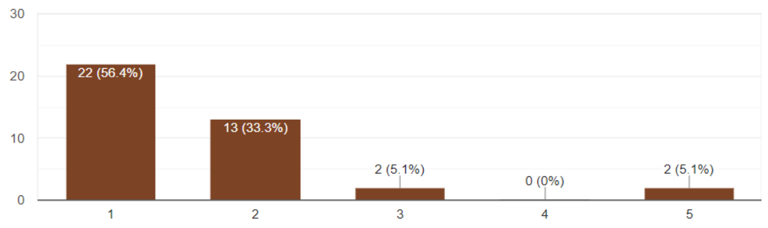
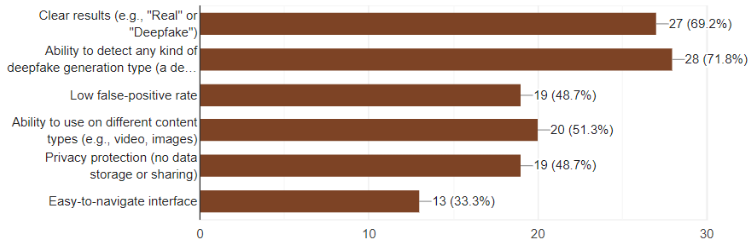
Finding: Existing Deepfake detection models often lack generalization capabilities, especially when applied across diverse datasets or with new manipulation techniques. Many models show strong performance on in-dataset testing but struggle with cross-dataset robustness.

Citation: (Korshunov and Marcel, 2022)

4.5.2 Survey findings

A survey was conducted to gather needs from the target audience to gather background awareness and to collect features to be included in the product.

Question	How concerned are you about the spread of Deepfake media in society?
Aim of question	To measure public concern on the spread of Deepfake media
Findings and conclusion  Most respondents rated themselves as “highly concerned” (1–2 on a scale of 5), indicating that the general public views deepfake content as a growing threat to trust and digital authenticity.	
Question	How concerned are you about a detection tool potentially mis-identifying real content as a deepfake due to not being updated on new kinds of Deepfake manipulations?
Aim of question	To measure public trust on current deepfake detection tools
Findings and conclusion  The majority selected between 1–3 on a 7-point scale, showing low confidence in current models’ reliability.	
Question	Would you feel more confident in using a detection tool if it had improved generalization on detecting more than one type of Deepfake manipulation technique?

Aim of question	To identify whether the public also feel the need of a system that has improved generalization in deepfake detection
Findings and conclusion  When asked if they would trust a detection tool more if it could handle multiple deepfake manipulation types, responses strongly supported this idea, with the majority selecting "Yes, definitely."	
Question	Which features are most important to you in Deepfake detection tool? (You can select more than one)
Aim of question	To find which features the public look forward to in a deepfake detection system.
Findings and conclusion  This finding shows that the public sees the ability to detect any kind of deepfake generation type is a very important feature they look forward to having.	

These findings directly validate the research motivation and project aim: to improve generalization in deepfake detection systems using advanced augmentation techniques.

4.5.3 Prototyping

Prototyping was used as an experimental method to verify the impact of augmentation strategies on generalization. Early results indicated that incorporating adversarial and real-world corruption-based augmentations led to measurable improvements in detection accuracy across unseen datasets. Full results and evidence will be detailed in the Evaluation chapter.

4.5.4 Brainstorming/observation

Brainstorming sessions with peers, project supervisors, and domain experts helped define key functional and non-functional requirements. Notably, they:

- Inspired the inclusion of *test-time training* to improve real-world adaptability,
- Shaped decisions around *user interface simplicity*, and
- Confirmed the need for *reinforcement learning-based augmentation selection* to reduce overfitting.

These collaborative sessions ensured the design reflected both technical feasibility and user expectations.

4.6 Summary of findings

The findings from all elicitation methods collectively guided the design and development of the GenDF deepfake detection system. They reinforced the project’s primary goals: improving cross-dataset generalization, optimizing model performance, and integrating explainable AI (XAI) where feasible. This consolidated understanding of stakeholder needs and system limitations serves as a strong foundation, ensuring the final solution is both technically robust and aligned with user and expert expectations.

4.7 Context diagram

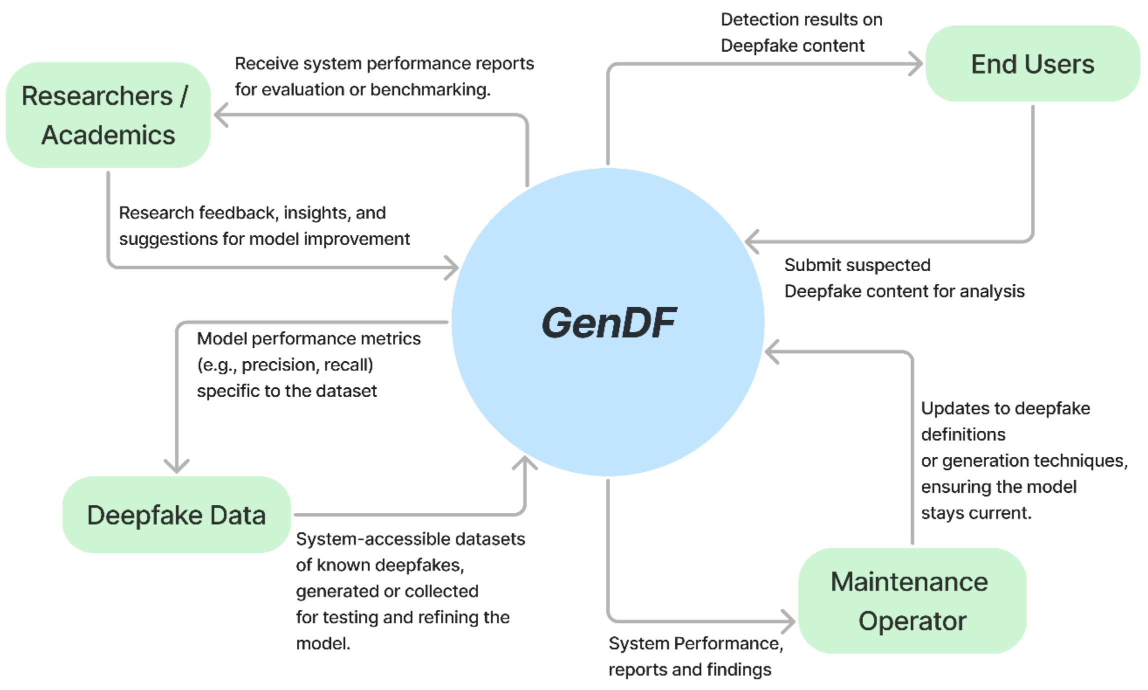

Researchers / Academics
Receive system performance reports for evaluation or benchmarking.
Detection results on Deepfake content
End Users
Research feedback, insights, and suggestions for model improvement
Submit suspected Deepfake content for analysis
GenDF
Model performance metrics (e.g., precision, recall) specific to the dataset
Updates to deepfake definitions or generation techniques, ensuring the model stays current.
Deepfake Data
System-accessible datasets of known deepfakes, generated or collected for testing and refining the model.
Maintenance Operator
System Performance, reports and findings

Figure 04: Context diagram (self-composed using Figma)

The context diagram illustrates the boundaries and key interactions of the GenDF deepfake detection system. It highlights how the system exchanges data with external entities such as users, maintenance operators, and dataset sources, providing a clear overview of information flow prior to detailed system development.

4.8 Use case diagram

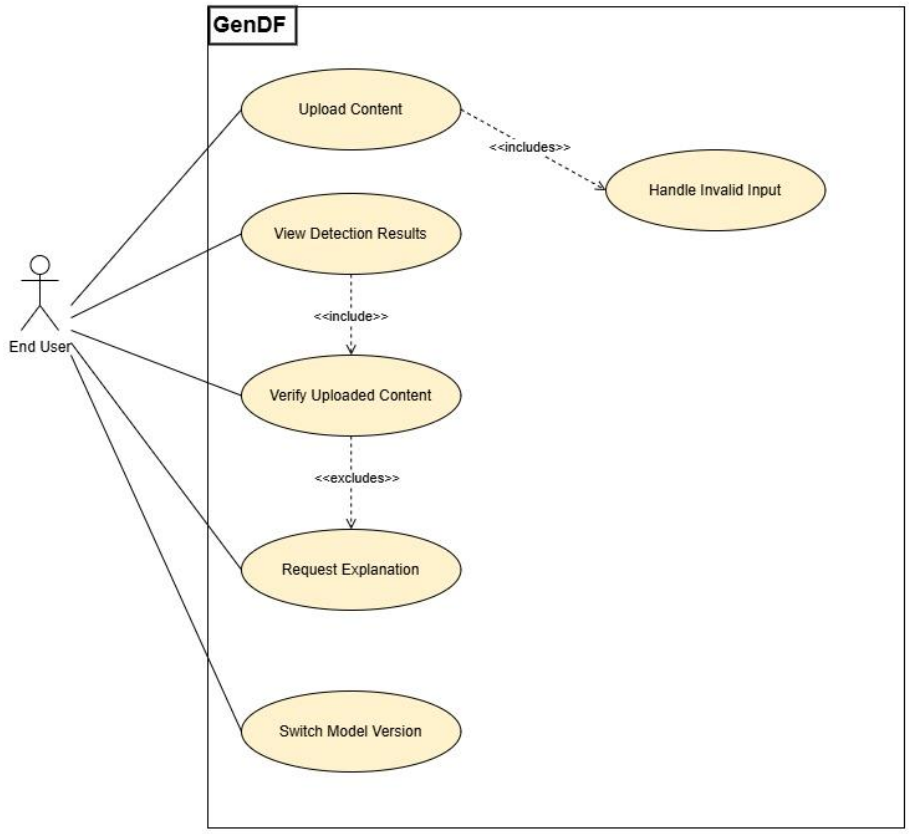


Figure 05: Use Case Diagram (Self-Composed using Figma)

This use case diagram reflects the core interactions within the final GenDF system. The primary actor, the End User, can upload media, verify its authenticity using one of three model versions, and view detection results. In Versions A and B, users can also request Grad-CAM–based explanations to interpret the model's decision; however, this feature is excluded in Version C due to the architectural limitations introduced by the TTT adapter. System operations such as model retraining or diagnostics are not part of the current implementation. Each use case corresponds directly to implemented functionality within the PyTorch backend and Streamlit frontend.

4.9 Use case descriptions

Use Case	Upload Content	Verify Uploaded Content
Goal	Upload media files for verification	Determine authenticity (real or fake)
Primary Actor	End User	End User
Preconditions	• User is authenticated (if required) • System is operational	• Content is uploaded • Detection model is active
Postconditions	Content is uploaded and queued for analysis	User receives authenticity result
Main Flow	1. User selects file 2. System checks format/size 3. File stored for processing	1. User selects file 2. Detection runs 3. Result displayed
Alternative Flows	A1. Invalid format > show error A2. File too large > prompt for smaller file	A1. Detection fails > show error A2. File unreadable > prompt re-upload

Use case descriptions for 'Request Explanation', 'View Detection Results', 'Generate System Reports and Diagnostics', and 'Update Model and Retrain' are provided in the ***APPENDIX 04***

.

4.10 Requirements

The MoSCoW prioritization technique is applied to organize the system requirements into four categories: “Must Have”, “Should Have”, “Could Have”, and “Will Not Have”. This ensures the core functionalities are implemented while identifying additional features that can enhance the system if time and resources permit.

4.10.1 Functional requirements

ID	Requirement	Priority (MoSCoW)
FR01	The system must support generalization across image-based deepfake manipulation techniques (e.g., facial image forgeries), particularly within human portrait datasets.	Must have
FR02	Flag uploaded content as either real or deepfake.	Must have
FR03	The user should have the ability to upload a picture for detection	Should have
FR04	The system could have a feature for users to submit feedback on detection results for system improvement.	Could have
FR05	The system should have an explainability module (e.g., Grad-CAM) to provide visual insights into the detection process.	Could have
FR06	The system could integrate additional datasets for comprehensive deepfake identification.	Could have
FR07	Integration with real-time content monitoring pipelines (e.g., social media scanners) will not be included.	Will not have
FR08	The system supporting real-time video deepfake detection in this phase.	Will not have

Table 05: Functional requirements

4.10.2 Non-functional requirements

ID	Requirement	Priority (MoSCoW)
NFR01	The system must process Deepfake image detection requests as fast as possible.	Must have
NFR02	The system must ensure data security and privacy by not storing any information.	Must have
NFR03	The system should maintain an accuracy rate of at least 75% through generalization tested using internal augmentations and controlled dataset variation.	Should have
NFR04	The system should provide an up-time of at least 90% for reliable access.	Should have
NFR05	The system could provide a user-friendly interface for non-technical users to interact with the results.	Could have
NFR06	The system could offer a dashboard with performance metrics on detection accuracy and model efficiency.	Could have
NFR07	The system handling real-time Deepfake video processing.	Will Not have

Table 06: Non-functional requirements

4.11 Chapter summery

This chapter outlined the core functional and non-functional requirements of the deepfake detection system, structured using the MoSCoW prioritization method. It also detailed the requirement elicitation techniques used, ensuring the system is aligned with user needs and technical goals. The requirements support building a reliable, secure, and generalizable model while allowing flexibility for future enhancements.

CHAPTER 05: SOCIAL, LEGAL, ETHICAL & PROFESSIONAL ISSUES

5.1 Chapter overview

This chapter addresses the Social, Legal, Ethical, and Professional (SLEP) issues relevant to the development and deployment of the GenDF deepfake detection system. It explores potential risks and responsibilities encountered throughout the project and reflects on how each issue was mitigated, with reference to the BCS Code of Conduct where applicable.

5.2 SLEP issues and mitigation

5.2.1 Social issues

GenDF addresses social risks associated with deepfake content, such as misinformation, erosion of trust, and cyberbullying. A key concern was bias due to limited dataset diversity. While large-scale datasets like DFDC were initially explored, the final model used Kaggle's Real and Fake Face Detection dataset. To improve fairness, image augmentations simulated visual variation, while XAI outputs and confidence scores promoted transparency. This aligns with **BCS Code of Conduct Section 1c** on fairness and inclusion.

5.2.2 Legal issues

Legal compliance was ensured by using a publicly available Kaggle dataset released for academic use. No personal or copyrighted data was used. The system runs entirely in memory, ensuring no data is stored, reused, or logged to support GDPR compliance. Survey responses were anonymous, with informed consent implied through voluntary participation. This reflects **BCS Code of Conduct Section 2d** on data privacy and legal responsibility.

5.2.3 Ethical issues

Ethical concerns included the risk of false positives and system misuse. Mitigation strategies included explainability features (Grad-CAM), clear accuracy reporting, and no real-time deployment. Dataset limitations were acknowledged, and the system is restricted to research purposes. The system is not intended for public deployment to prevent malicious misuse or reputational harm caused by false claims about media authenticity.

These considerations reflect BCS Code of Conduct Sections 1a (“The public interest”) and 1d (“Avoid harming others”), ensuring the system is built and used ethically, with respect for user rights, societal impact, and potential harms.

5.2.4 Professional issues

Professional conduct was maintained by documenting limitations, reporting realistic performance metrics, and avoiding exaggerated claims. All models, parameters, and results were version-controlled and reproducible.

The project maintained ongoing communication with supervisors and stakeholders, providing regular updates and highlighting system limitations during evaluation stages. This ensured realistic expectations and informed discussions about the project's capabilities and boundaries.

These actions align with the BCS Code of Conduct Section 3a, which stresses the importance of maintaining professional competence and integrity, especially when working on high-impact technologies like deepfake detection.

5.3 Chapter summary

This chapter explored the social, legal, ethical, and professional considerations associated with the GenDF project. Measures were taken to uphold public trust, data privacy, and fairness while adhering to ethical AI principles and the BCS Code of Conduct. Risks such as bias, misuse, and overstatement of performance were addressed through careful system design, dataset selection, transparency, and documentation.

CHAPTER 06: DESIGN

6.1 Chapter overview

This chapter presents the architectural, algorithmic, and interface design of the GenDF system, a deepfake detection framework developed in three progressively enhanced versions. It translates theoretical models into structured design decisions that enable generalization, robustness, and adaptability. The design phase outlines the modular structure, component interactions, and data flow strategies that form the foundation for implementing reinforcement learning, contrastive learning, and test-time training in a scalable and maintainable system.

6.2 Design goals

The design of the GenDF system is guided by a set of key quality attributes tailored to meet the challenges of robust and generalizable deepfake detection. The design goals are detailed in ***APPENDIX 05***

, which outlines the critical architectural principles that informed system development.

6.3 System architecture design

6.3.1 High-level three-tiered architecture

The system is structured using the three-tiered architecture pattern, which is a well-established approach for building complex applications in a maintainable and scalable manner. Each tier is responsible for a specific group of responsibilities, reducing the interdependency between components and facilitating modular development.

- **Presentation Tier**: Handles all interactions with the end user via a web interface powered by Streamlit.
- **Logic Tier**: Performs all the application logic, including deep learning inference, reinforcement learning-based augmentation, and contrastive learning.
- **Data Tier**: Manages storage and access to datasets and trained model weights

This separation allows for independent development and debugging of each tier, improving the overall reliability and extensibility of the system.

The architecture supports three distinct model configurations (Versions A, B, and C), each integrating different combinations of reinforcement learning (RL), contrastive learning (CL), and test-time training (TTT) while sharing a common execution framework.

6.3.2 Component interaction flow

The system initiates at the Presentation Tier, where users upload an image and select a model via the Streamlit UI. Although all components run within the same runtime environment, logical separation is maintained: the UI handles input/output, while internal modules handle data processing, augmentation, and model inference.

Presentation Tier	1. Image upload and preview	Users upload an image via the web interface, which is previewed in real time.
Data Tier	2. Model and dataset handling	The system dynamically loads the selected version's model weights (gendf_versionA/B/C.pth). Dataset structure is accessed using PyTorch's ImageFolder API for consistent label mapping and augmentation support.
Logic Tier	3. Preprocessing and augmentation	The uploaded image is resized, normalized, and passed through a PPO-based RL module that selects augmentations dynamically for all versions.
	4. Contrastive feature consistency (Version B and C only)	Contrastive learning minimizes intra-class variation and maximizes inter-class separation by comparing differently augmented views.
	5. Test-Time Training (Version C only)	A lightweight adapter applies KL-divergence regularization to adapt feature representations at inference time, improving robustness to unseen manipulations.

	6. Prediction Generation	The system generates the final classification (Real or Fake) using the model pipeline corresponding to the selected version.
Presentation Tier	7. Result display	The classification result, confidence score, and Grad-CAM explanation (supported in Version A and B) are returned to the user within the Streamlit interface.

Figure 06: Component interaction flow

6.3.3 System architecture diagram

The following diagram presents the three-tiered architecture of the Deepfake Detection System, showing how various components interact across the tiers.

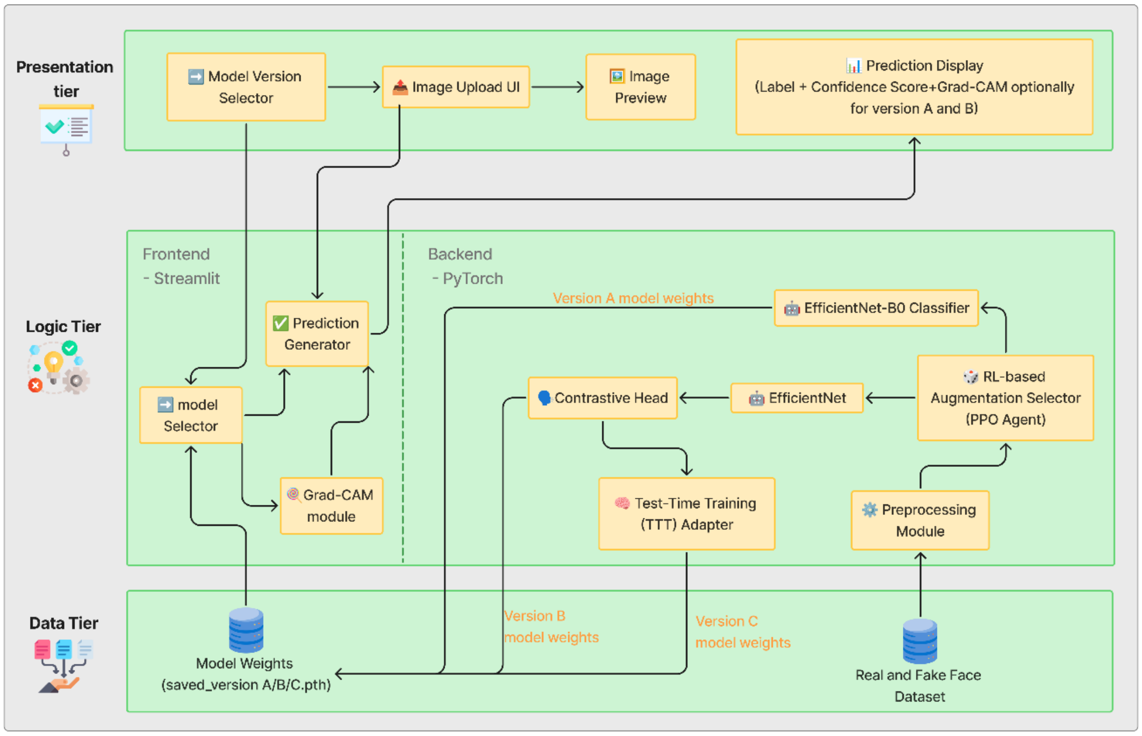


Figure 07 Three-tiered Architecture (self-composed using Figma)

6.4 Detailed design

6.4.1 System components and their roles

The proposed system uses a modular, object-oriented architecture split into Presentation, Logic, and Data tiers. Below are the key components and their roles.

In this system:

- Model training happens offline (in Colab / backend).
- UI is only responsible for loading the model for inference.

Tier	Component	Role
Presentation Tier	Model Version Selector	Allows users to choose between Versions A, B, or C.
	Image Upload UI	Facilitates image input from the user.
	Image Preview	Shows the uploaded image before prediction.
	Prediction Display	Shows classification result, confidence, and (for Versions A and B) Grad-CAM overlay.
Logic Tier	Preprocessing Module	Resizes and normalizes the image (224×224).
	RL-based Augmentation Selector	PPO agent selects optimal augmentations
	EfficientNet-B0 Classifier	Core model for real/fake prediction
	Contrastive Learning Module *(B and C only)*	Improve feature separation
	Test-Time Training Adapter *(C only)*	Enhances inference robustness
	Grad-CAM Module *(A and B only)*	Highlights decision-influencing regions
Data Tier	Real and Fake Face Dataset	Used for training/testing the model
	Model Weights (A/B/C)	Loaded dynamically based on user selection

Table 07: System components and their roles

6.4.2 Component interaction and workflow

The workflow starts with the user uploading an image and selecting a model version through the Streamlit UI. After preprocessing, PPO-driven augmentation is optionally applied. The image then flows through the model pipeline,

- **Version A**: EfficientNet-only.
- **Version B**: Adds contrastive consistency.
- **Version C**: Adds test-time adaptation. The predicted label and confidence score are then displayed, with Grad-CAM activated in Versions A and B for interpretability. All models load pretrained weights and reference the shared dataset.

(Figure 08 illustrates this interaction visually.)

6.4.3 Design paradigm

The design paradigm selected for this system is **Object-Oriented Analysis and Design (OOAD)**. This choice is justified by the modular nature of the project, which includes components like the classifier, augmentation logic, feature learners, and adapters that are each encapsulated as an object or module with specific responsibilities and methods.

The decision to use OOAD is further supported by:

- Use of class-based model wrappers (e.g., EfficientNet modification, feature extractors, adapters).
- Clear inheritance and method overriding (e.g., contrastive loss, TTT adaptation).
- Reusability and extensibility of modules for future improvements.

6.4.4 Diagrams

1. Component diagram

The component diagram as shown in ***APPENDIX 06***

illustrates the modular structure of the GenDF system, highlighting the interaction between the frontend, backend, and data layers. The frontend UI, built with Streamlit in VS Code, interacts with a pre-trained model for real-time inference. This model is produced by the backend training engine, developed in Python and executed on Google Colab. The training engine leverages a dataset of real and fake faces and integrates three learning modules: reinforcement learning (RL) for augmentation (via Stable-Baselines3), contrastive learning for feature consistency, and test-time training (TTT) for adaptability during inference. These modules communicate with the core EfficientNet classifier, which is responsible for both training and prediction. The diagram clearly maps the data flow and component dependencies, ensuring a modular and scalable system architecture.

2. Class diagram

The class diagram in ***APPENDIX 07*** presents the core architecture of the GenDF system, highlighting the main classes and their interactions. ModelTrainer handles training and evaluation of all three EfficientNet-based model versions, using data from the Dataset class and optionally applying RL-based augmentations via ReinforcementLearningAgent. ImageProcessor manages image resizing and transformations. Trained models are used by DeepfakeDetector for inference, while StreamlitApp provides the user interface, handling image uploads, displaying results, and supporting Grad-CAM visualizations (used as a utility, not shown) and internal components like the TTT adapter and projection head are abstracted here.

3. Sequence diagram

The diagram in ***APPENDIX 08*** depicts the interaction flow from image upload to result display in the GenDF system. The user uploads an image via the StreamlitApp, which validates and forwards it to the DeepfakeDetector. If valid, the image is transformed by the ImageProcessor and passed to the EfficientNet-based Saved Model for prediction. The result (Real/Fake with confidence) is returned to the frontend and displayed. Invalid inputs trigger appropriate feedback. This sequence reflects the actual system logic implemented using PyTorch and Streamlit.

6.5 Algorithm design

6.5.1 Key algorithms

The GenDF deepfake detection system integrates four core algorithmic components: EfficientNet-B0, Reinforcement Learning (RL), Contrastive Learning (CL), and Test-Time Training (TTT) to ensure high generalization across unseen manipulation styles and datasets.

1. EfficientNet-B0 for feature extraction

EfficientNet-B0 serves as the foundational feature extractor across all GenDF model versions (A, B, and C). Pretrained (Jain, Korshunov and Marcel, 2021) on ImageNet and fine-tuned on real-vs-fake face images, it transforms input images into a 1280-dimensional feature vector. For binary classification, the final layer is modified to output two logits representing real and fake classes.In Version A, EfficientNet directly predicts the class label (Smelyakov, Kitsenko and Chupryna, 2024). In Versions B and C, this output is routed through additional modules (projector, contrastive loss, and classifier) for enhanced generalization. Its lightweight architecture makes it suitable for real-time use on constrained hardware while capturing fine-grained manipulations such as facial warping or compression artifacts (Wang *et al.*, 2022).

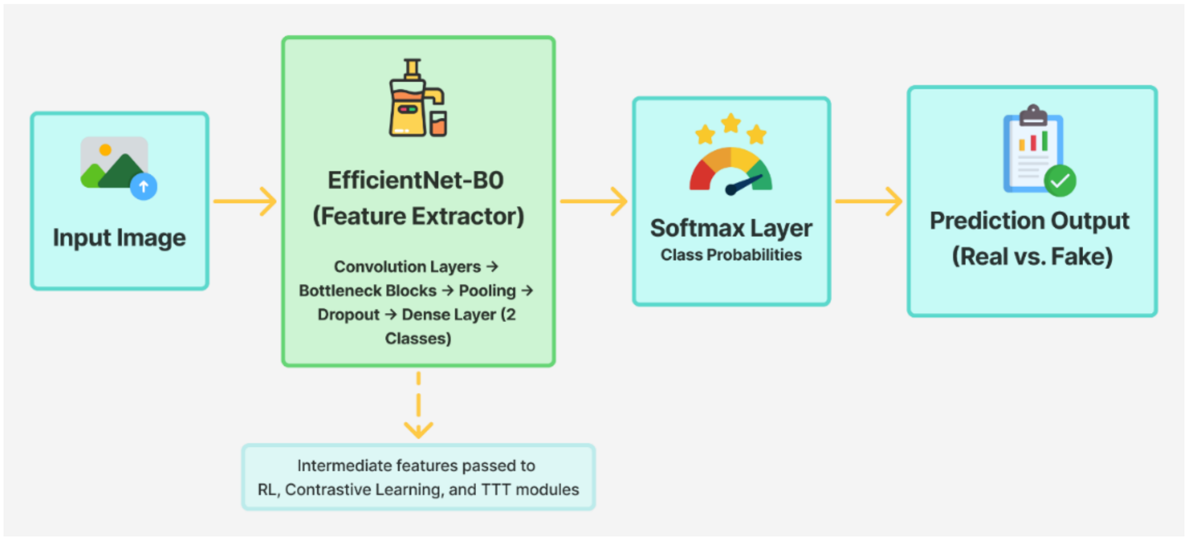

Input Image
EfficientNet-B0 (Feature Extractor)
Convolution Layers → Bottleneck Blocks → Pooling → Dropout → Dense Layer (2 Classes)
Softmax Layer
Class Probabilities
Prediction Output (Real vs. Fake)
Intermediate features passed to RL, Contrastive Learning, and TTT modules

Figure 08: Feature Extraction with EfficientNet-B0

As was discussed above EfficientNet is the core CNN model, but just training it normally on a dataset:

- Can make it overfit (Stanciu and Ionescu, 2024)
- Can make it memorize dataset-specific artifacts (Guan *et al.*, 2024)
- Can make it fail on new deepfake styles (Gupta *et al.*, 2023)

This is where RL, CL, and TTT come in. These supporting modules enhance the learning process, not just the output. The following sections will discuss these supporting modules in depth.

2. Reinforcement Learning for Adaptive Augmentation (PPO Agent)

To prevent overfitting (Das *et al.*, 2021) and dataset bias during training, GenDF integrates a PPO-based RL agent that dynamically selects image augmentations (e.g., rotation, flip, sharpness). The agent operates in a custom environment, receiving a reward based on the feature activation strength from EfficientNet for each augmented image (Nadimpalli and Rattani, 2022).

This dynamic strategy replaces fixed augmentation pipelines and encourages the model to learn features robust to diverse manipulations. The augmented dataset produced by the PPO agent is used to train all model versions, improving their resistance to overfitting and enhancing cross-dataset resilience.

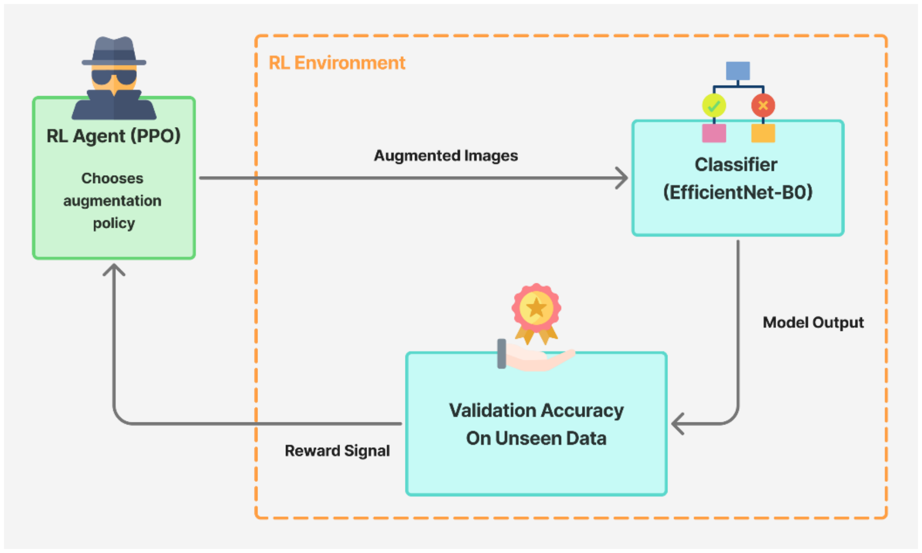

RL Agent (PPO)
Chooses augmentation policy
RL Environment
Augmented Images
Classifier (EfficientNet-B0)
Model Output
Validation Accuracy On Unseen Data
Reward Signal

Figure 09: Reinforcement Learning (self-composed using Figma)

3. Contrastive learning for cross-dataset feature consistency (versions B and C)

To encourage domain-invariant learning, versions B and C incorporate contrastive learning. Each training image is transformed into two distinct augmented views (e.g., flipped, noised). The model minimizes the distance between same-class features (positive pairs) and maximizes it for different-class features (negative pairs) using a cosine similarity-based loss (Lin *et al.*, 2024).

A two-layer projection head (1280 → 512 → 128) converts EfficientNet's features into compact embeddings for contrastive comparison. This prevents shortcut learning and ensures the model focuses on manipulation-specific patterns rather than dataset-specific artifacts.

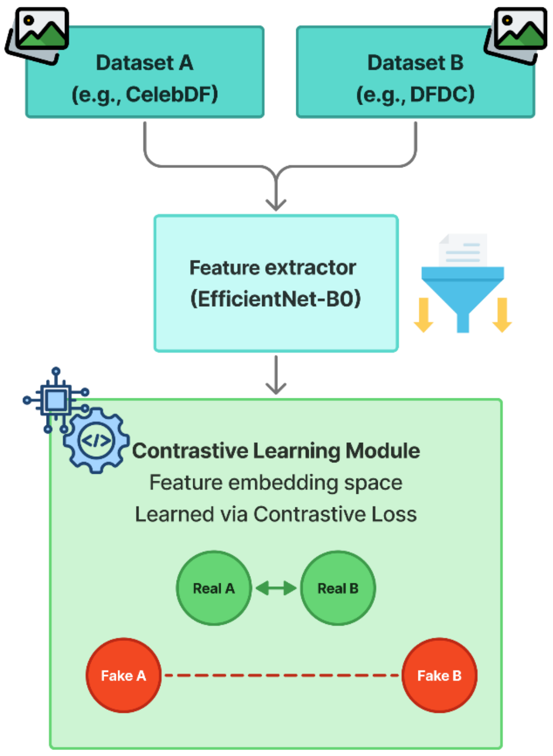


Figure 10: Contrastive learning (self-composed using Figma)

4. Test-Time Training (TTT) with KL-Divergence for real-time adaptation (version C only)

Version C further introduces a lightweight TTT adapter module to fine-tune predictions during inference. When an unfamiliar deepfake style is encountered, the model self-supervises using its initial output as a pseudo-label (Chen *et al.*, 2022). It then minimizes KL-divergence between the original and adapted predictions via a 128D fully connected adapter.

TTT is only activated during inference and allows real-time adaptation without labeled data, which is crucial for unpredictable media environments. This boosts robustness against distribution shifts and ensures stable performance even on novel attacks.

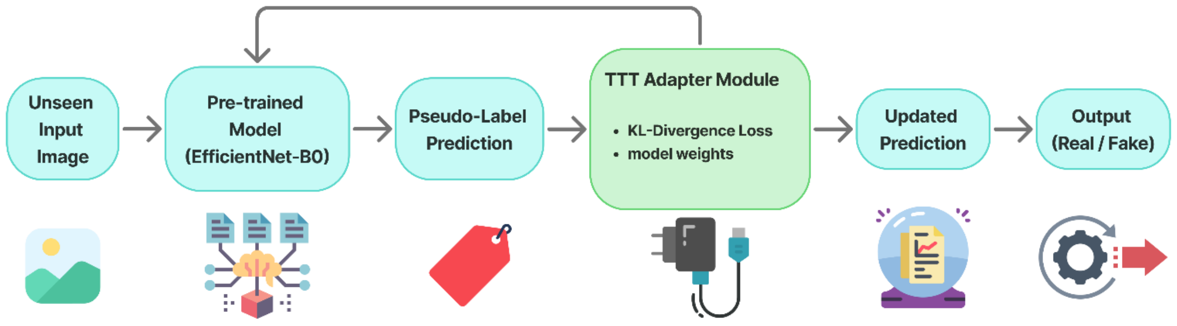


Figure 11: Test-Time Training (self-composed using Figma)

5. Grad-CAM for explainable inference (versions A and B)

To enhance interpretability, GenDF integrates Grad-CAM in Versions A and B. During inference, Grad-CAM hooks into the final convolutional layers of EfficientNet to generate heatmaps (Yin *et al.*, 2024) highlighting regions that most influenced the prediction. This visual explanation is presented to users alongside the classification result via the Streamlit interface.

This transparency feature not only improves trustworthiness for end-users but also aids in validating model behavior across varying deepfake types.

6.5.2 Flow of execution

The execution pipeline of GenDF consists of six streamlined stages that vary slightly across the three model versions, all built around EfficientNet-B0.

Step 1: Image ingestion and preprocessing

An input image is uploaded or loaded from the dataset. It is resized to 224×224 and normalized to match the model's expected input format.

Step 2: Adaptive augmentation via Reinforcement Learning (training only)

During training, a PPO agent selects augmentation combinations (e.g., rotation, sharpness) that maximize validation performance. These augmentations are applied dynamically, ensuring feature extractor exposure to varied distortions and enhancing cross-dataset generalization.

Step 3: Feature extraction (EfficientNet-B0)

The processed image is passed through EfficientNet-B0 to extract deep features. This forms the foundation for classification in all three versions:

- Version A: Features are directly classified.
- Version B: Features are passed through a contrastive head before classification.
- Version C: Features undergo both contrastive projection and real-time adaptation via a TTT adapter.

Step 4: Contrastive Learning (versions B and C)

To improve domain-invariant learning, contrastive loss is applied during training by comparing feature pairs (same vs. different class). This forces the model to learn manipulation-based representations instead of dataset-specific cues.

Step 5: Test-Time Training adaptation (version C Only)

During inference, Version C activates a lightweight adapter module. It uses the model's initial prediction as a pseudo-label and adjusts the final features using KL-divergence loss. This enables real-time self-adjustment on novel deepfakes without requiring labeled data.

Step 6: Classification and explainability output

The final classifier predicts whether the image is real or fake, outputting a confidence score. In Versions A and B, Grad-CAM visualizations highlight the most influential regions, providing interpretability through overlayed heatmaps.

The following diagram visually illustrates the step-by-step flow of GenDF's execution pipeline, from image input to final prediction.

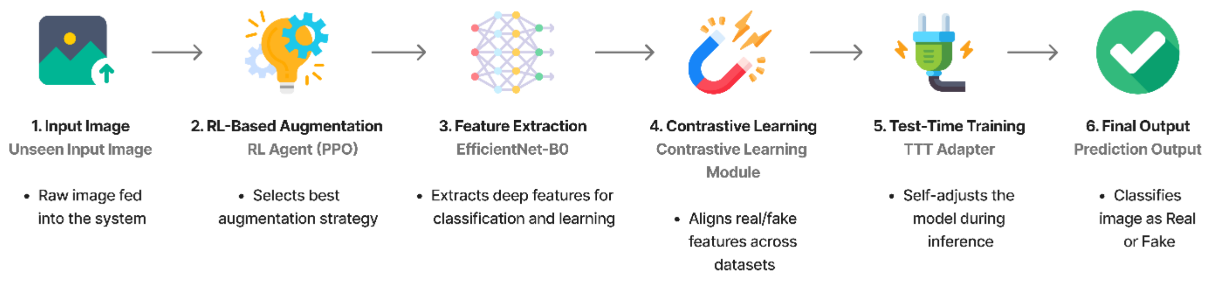

1. Input Image
Unseen Input Image
• Raw image fed into the system
2. RL-Based Augmentation
RL Agent (PPO)
• Selects best augmentation strategy
3. Feature Extraction
EfficientNet-B0
• Extracts deep features for classification and learning
4. Contrastive Learning
Contrastive Learning Module
• Aligns real/fake features across datasets
5. Test-Time Training
TTT Adapter
• Self-adjusts the model during inference
6. Final Output
Prediction Output
• Classifies image as Real or Fake

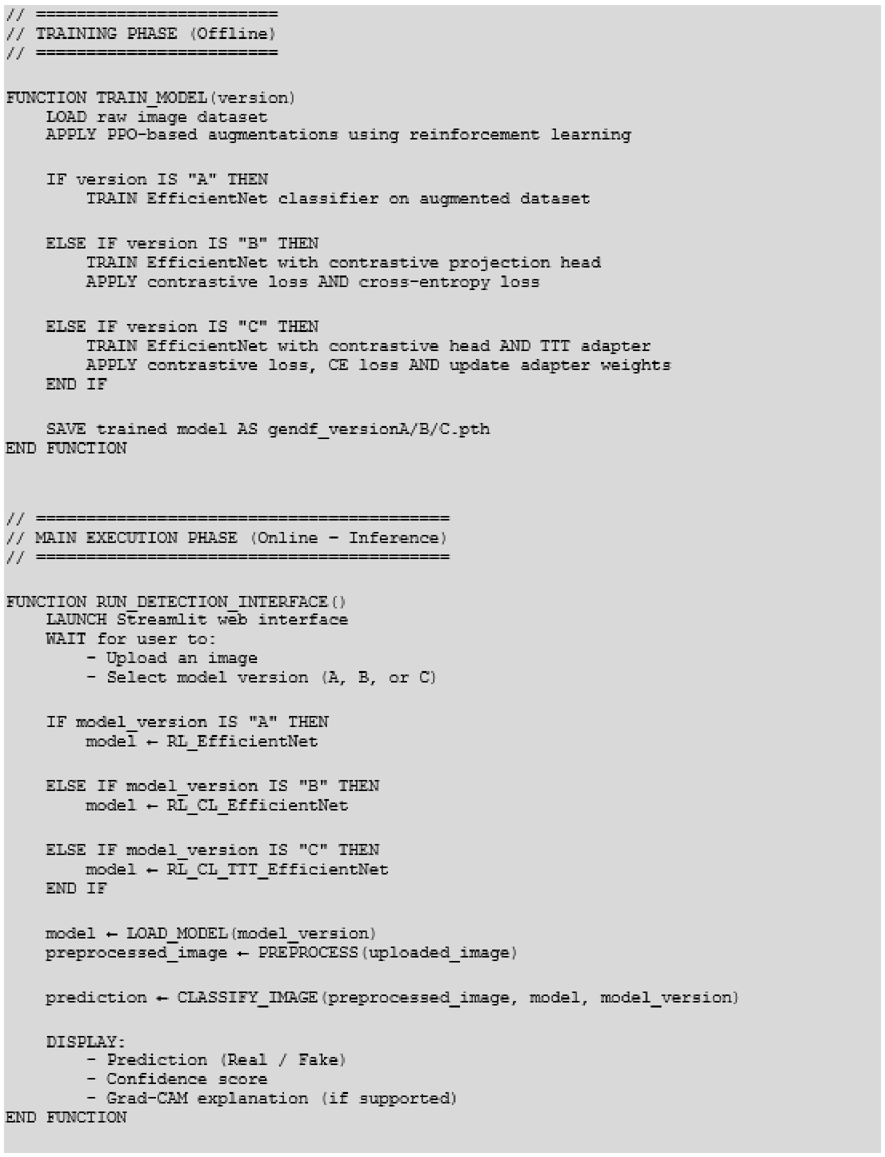
Figure 12: Linear overview of flow of execution (self-composed using Figma)

While the flow is presented sequentially for clarity, EfficientNet-B0 serves as the central backbone throughout training and inference. Auxiliary modules such as Reinforcement Learning (RL), Contrastive Learning (CL), and Test-Time Training (TTT) interact with it at different stages to guide augmentation, optimize features, and enable adaptation but all predictions ultimately come from EfficientNet.

Together, these components form a robust and intelligent pipeline that goes beyond traditional deepfake detection by learning, generalizing, and adapting, which is the very core purpose of the GenDF framework.

6.5.3 Pseudocode of the deepfake detection framework

This pseudocode outlines how the GenDF system is trained and deployed using standard pseudocode conventions.



```
// ========================
// TRAINING PHASE (Offline)
// ========================

FUNCTION TRAIN_MODEL(version)
    LOAD raw image dataset
    APPLY PPO-based augmentations using reinforcement learning

    IF version IS "A" THEN
        TRAIN EfficientNet classifier on augmented dataset

    ELSE IF version IS "B" THEN
        TRAIN EfficientNet with contrastive projection head
        APPLY contrastive loss AND cross-entropy loss

    ELSE IF version IS "C" THEN
        TRAIN EfficientNet with contrastive head AND TTT adapter
        APPLY contrastive loss, CE loss AND update adapter weights
    END IF

    SAVE trained model AS gendf_versionA/B/C.pth
END FUNCTION


// ==========================================
// MAIN EXECUTION PHASE (Online - Inference)
// ==========================================

FUNCTION RUN_DETECTION_INTERFACE()
    LAUNCH Streamlit web interface
    WAIT for user to:
        - Upload an image
        - Select model version (A, B, or C)

    IF model_version IS "A" THEN
        model ← RL_EfficientNet

    ELSE IF model_version IS "B" THEN
        model ← RL_CL_EfficientNet

    ELSE IF model_version IS "C" THEN
        model ← RL_CL_TTT_EfficientNet
    END IF

    model ← LOAD_MODEL(model_version)
    preprocessed_image ← PREPROCESS(uploaded_image)

    prediction ← CLASSIFY_IMAGE(preprocessed_image, model, model_version)

    DISPLAY:
        - Prediction (Real / Fake)
        - Confidence score
        - Grad-CAM explanation (if supported)
END FUNCTION
```

Figure 13: Pseudocode

6.6 UI design

The UI design of the GenDF system focuses on delivering a clean, intuitive, and accessible user experience for deepfake detection. Users can upload images and receive classification results alongside Grad-CAM visual explanations, all within a lightweight web interface.

6.6.1 Low-fidelity wireframe

The full low-fidelity UI design is available in the ***APPENDIX 09***

.

6.6.2 High-fidelity prototype

The fully functional interface was implemented using Streamlit, with the following features:

- Minimalist layout with image preview, prediction result, and Grad-CAM support.
- Device responsiveness, supporting both desktop and mobile resolutions.
- Progress bar and visual cues for confidence score interpretation.
- Model switcher: Users can choose between Version A, B, or C dynamically.

This interface reflects the final deployed version used in testing and evaluation (*see Chapter 7*).

6.6.3 Usability and accessibility considerations

- Simple navigation with radio buttons and checkboxes.
- High-contrast colors, clear fonts, and screen-reader friendly elements.
- Keyboard navigation and alternative text for Grad-CAM overlays.
- Instant feedback after upload, including result, confidence score, and explanation.

6.6.4 UI flow

The step-by-step interaction from the user's perspective is,

1. Upload an image (.jpg, .jpeg, or .png).
2. Select a model version (A, B, or C).
3. The model performs inference and returns a classification result, a confidence score, an optional Grad-CAM overlay (if selected), and a brief explanation to help interpret the prediction.

6.7 System process workflow

6.7.1 Workflow overview

The GenDF system follows a streamlined workflow to perform deepfake image classification. It covers image input, preprocessing, model inference, and output delivery through a user-friendly interface.

6.7.2 Step-by-step process flow

1. **Image Upload:** Users upload an image through the Streamlit frontend
2. **Preprocessing:** The image is resized to 224×224 pixels, converted to a tensor, and normalized to match the model input format
3. **Model Inference:** Based on the selected model version (A, B, or C), the image is passed through the corresponding PyTorch model
4. **Result Visualization:** The interface displays the classification result, confidence score (as a percentage and progress bar), and an optional Grad-CAM overlay to highlight influential regions of the image. An explanation text helps interpret the prediction

6.7.3 Key considerations

The system is designed for efficiency with minimal latency, ensures a seamless user experience with immediate and clear feedback, and supports scalability for multiple datasets and deepfake detection models.

6.7.4 Activity diagram

The following activity diagram illustrates the end-to-end system workflow, from the user uploading an image to receiving the prediction result. This diagram encapsulates the core operations of the detection pipeline.

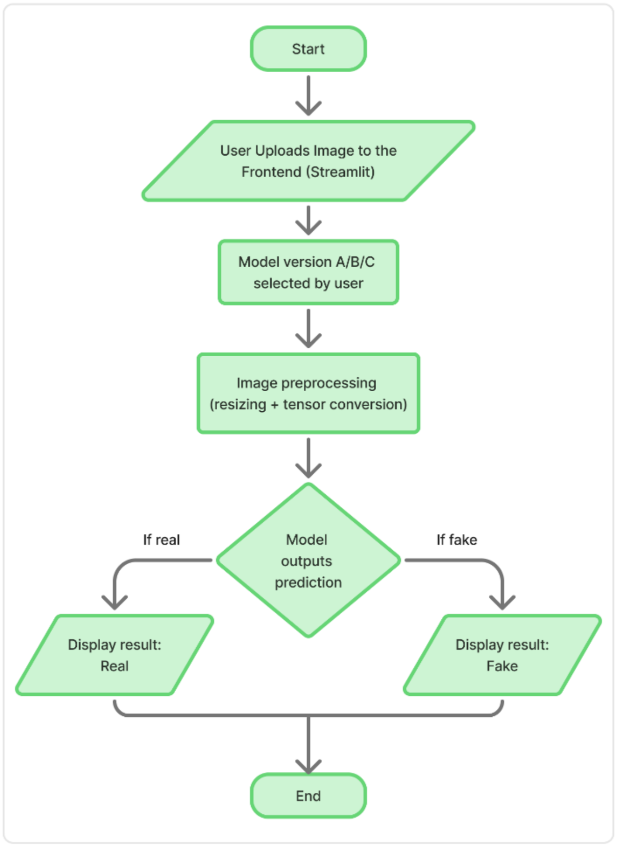

Start
User Uploads Image to the Frontend (Streamlit)
Model version A/B/C selected by user
Image preprocessing (resizing + tensor conversion)
Model outputs prediction
If real
If fake
Display result: Real
Display result: Fake
End

Figure 14: Activity diagram (self-composed using Figma)

6.8 Chapter summary

This chapter discussed how the system is designed for modularity, scalability, and usability, integrating reinforcement learning, contrastive learning, and test-time training to enhance deepfake detection. Its tiered architecture enables efficient processing and real-time inference, while the accessible UI ensures smooth user experience. Together, these choices create a robust and adaptable framework suited for real-world use.

CHAPTER 07: IMPLEMENTATION

7.1 Chapter overview

This chapter presents the practical implementation of the proposed GenDF deepfake detection framework. It outlines how the system was developed by translating the design architecture into a fully functional prototype. The core functionalities are implemented across three progressively enhanced model versions. The transition from architectural design to backend code and an interactive frontend application is highlighted using code snippets and interface visuals. This implementation phase plays a vital role in validating the feasibility of the proposed design and achieving the project's research objectives.

7.2 Technology selection

7.2.1 Technology stack

The diagram below illustrates the technology stack used to implement the GenDF system, structured into three tiers based on its layered architecture.

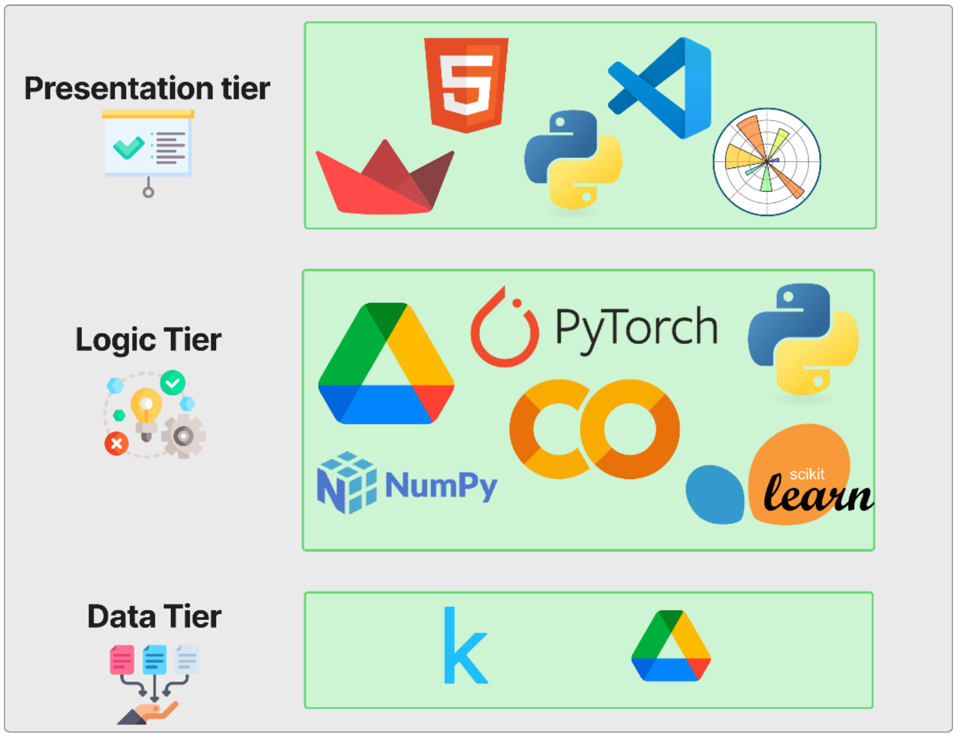


Figure 15: GenDF Technology Stack (self-composed via Figma)

7.2.2 Data selection

Several deepfake benchmark datasets were initially considered for this project, including FaceForensics++, Celeb-DF, DFDC, and WildDeepfake. While these offered high scale and realism, they presented practical challenges such as:

- Large video-based formats unsuitable for image-based models
- Restricted access or licensing constraints
- Missing or ambiguous labels in some cases

Due to these limitations, the “Real and Fake Face Detection” dataset on Kaggle was selected. This dataset consists of 2,041 images (1,083 real and 958 fake) organized in a clean folder-based structure, directly compatible with PyTorch’s ImageFolder class. The fake images were expertly crafted using Photoshop, offering a different detection challenge than common GAN-generated deepfakes. Its small size (225MB) ensured smooth integration with Google Colab’s memory and training limitations. Additionally, the dataset has over 26,000 downloads and 200,000 views, indicating its widespread use and reliability in the ML community (Maqbool and Nawaz, 2021).

While this dataset lacks GAN-generated or video-based deepfakes found in larger benchmarks like DFDC, it was deemed ideal for rapid prototyping, lightweight model training, and demonstrating the effectiveness of the proposed GenDF architecture under constrained conditions.

7.2.3 Selection of programming language

Python was selected as the sole programming language for both the backend and frontend due to its comprehensive deep learning ecosystem, seamless integration, and cross-platform support. It enabled the development of the project’s reinforcement learning, contrastive learning, and test-time training components using libraries like PyTorch and stable-baselines3.

Using Python end-to-end eliminated the need for API bridging layers, allowing the trained models to be directly integrated into the interface. Its compatibility with both Google Colab (for GPU training) and VS Code (for local deployment) further validated its suitability. Alternatives like R or JavaScript were not chosen due to limited deep learning support or unnecessary overhead for a unified ML pipeline.

7.2.4 Libraries used

These libraries were selected for their performance, community support, ease of integration, and compatibility with Google Colab and VS Code development environments. The selected libraries and along with their corresponding justification are provided in the table included in ***APPENDIX 10***.

7.2.5 Frameworks used

To support the modular development of GenDF, several lightweight yet powerful frameworks were selected.

- **Backend framework:** PyTorch was chosen as the core deep learning framework due to its dynamic computation graph, ease of use, and extensive ecosystem supporting model customization. For reinforcement learning, Stable-Baselines3 was used, providing a reliable PPO implementation and integration with custom Gym environments.
- **User Interface (UI) framework:** Streamlit was adopted for its simplicity and speed in building data-driven UIs. It enabled rapid prototyping and seamless deployment of the trained model, allowing end-users to upload images, view predictions, and interpret Grad-CAM visualizations in real time.
- **API layer:** No external API framework was required. As the trained models were internally loaded and executed within the Streamlit app, avoiding unnecessary overhead or architectural complexity for this project.

7.2.6 Integrated Development Environments (IDEs)

Two IDEs were used throughout the GenDF system development.

IDE	Justification
Google Colab	Used as the primary environment for model training and experimentation due to its free access to GPU resources, seamless integration with Google Drive, and compatibility with PyTorch-based workflows. It enabled efficient training of all three GenDF versions (RL, RL+CL, RL+CL+TTT) under constrained hardware conditions.
Visual Studio Code (VS Code)	Chosen for frontend development and local testing of the Streamlit application. Its extension ecosystem, Python debugging tools, and version control support significantly streamlined the deployment and UI refinement process.

Table 08: IDE justification

7.2.7 Summary of technology selection

Component	Tool(s) / Technology(ies)
Programming Language	Python
Development Frameworks	PyTorch, Stable-Baselines3, Streamlit
Libraries	Torch, Torchvision, NumPy, Scikit-learn, Matplotlib, Pillow (PIL), Gymnasium
Model Versions	Version A: RL + EfficientNet Version B: RL + CL + EfficientNet Version C: RL + CL + TTT + EfficientNet
Explainability	Grad-CAM (custom implementation using PyTorch and OpenCV)
IDE / Environments	Google Colab (training), Visual Studio Code (frontend)
Dataset	Real and Fake Face Detection (Kaggle)
Deployment Platform	Local (Streamlit Web App in VS Code)

Table 09: Summary of technology selection

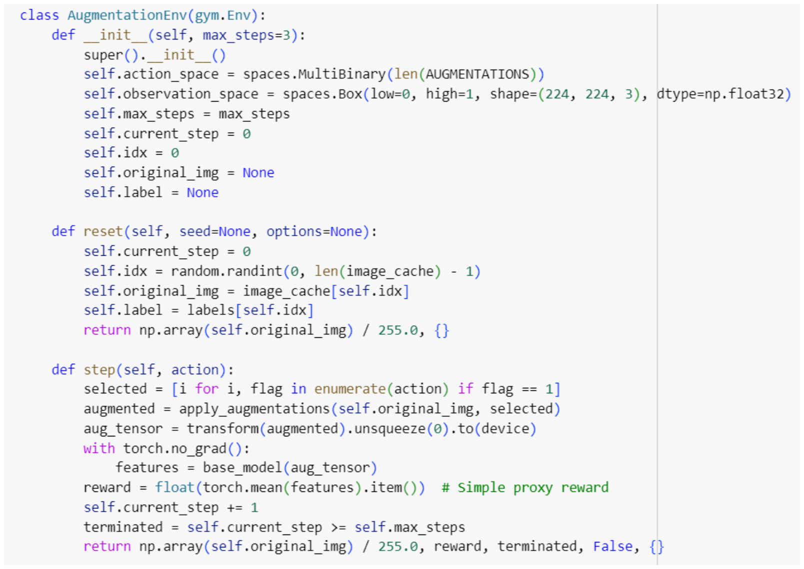
7.3 Implementation of core functionalities

This section presents the core implementation logic behind the three progressively enhanced versions of the GenDF system as Version A, Version B and Version C. Each version demonstrates a distinct architectural extension, emphasizing reinforcement learning (RL), contrastive learning (CL), and test-time training (TTT) respectively.

7.3.1 Version A (RL + EfficientNet)

PPO-based augmentation selection

A custom Gym environment was designed where each image is treated as a reinforcement learning (RL) episode. The PPO agent selects augmentations (e.g., flip, sharpness, rotation), which are applied to the image. The modified image is passed through a frozen EfficientNet feature extractor, and the mean feature value is used as the reward. This reward structure encourages selection of transformations that yield richer representations.



```python
class AugmentationEnv(gym.Env):
    def __init__(self, max_steps=3):
        super().__init__()
        self.action_space = spaces.MultiBinary(len(AUGMENTATIONS))
        self.observation_space = spaces.Box(low=0, high=1, shape=(224, 224, 3), dtype=np.float32)
        self.max_steps = max_steps
        self.current_step = 0
        self.idx = 0
        self.original_img = None
        self.label = None

    def reset(self, seed=None, options=None):
        self.current_step = 0
        self.idx = random.randint(0, len(image_cache) - 1)
        self.original_img = image_cache[self.idx]
        self.label = labels[self.idx]
        return np.array(self.original_img) / 255.0, {}

    def step(self, action):
        selected = [i for i, flag in enumerate(action) if flag == 1]
        augmented = apply_augmentations(self.original_img, selected)
        aug_tensor = transform(augmented).unsqueeze(0).to(device)
        with torch.no_grad():
            features = base_model(aug_tensor)
        reward = float(torch.mean(features).item())  # Simple proxy reward
        self.current_step += 1
        terminated = self.current_step >= self.max_steps
        return np.array(self.original_img) / 255.0, reward, terminated, False, {}
```

Figure 16: AugmentationEnv class and step() method implementation

Dataset generation using PPO

The trained PPO agent is then used to generate an augmented training dataset. Each image is transformed based on the agent's chosen actions as seen in ***APPENDIX 11***.

These images are saved and organized using class-wise folder structures compatible with PyTorch's ImageFolder.

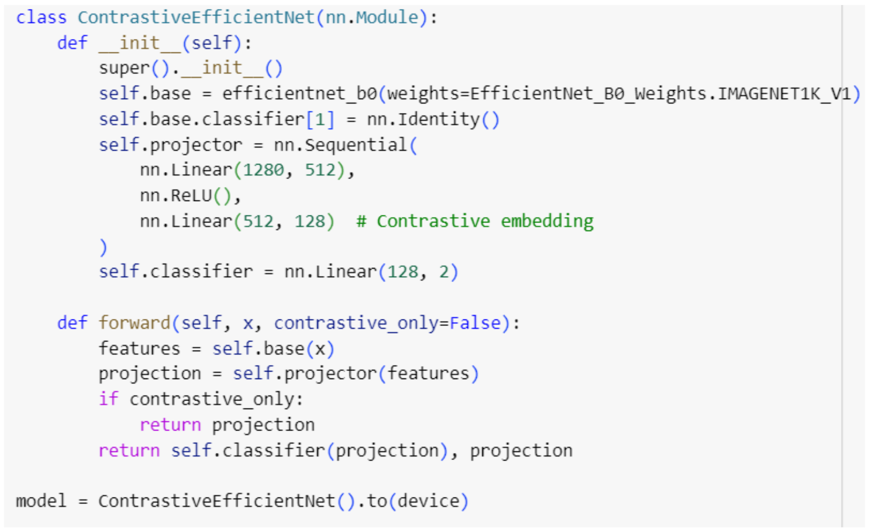
EfficientNet classification

A fresh EfficientNet-B0 is fine-tuned using the PPO-generated dataset. The final layer is replaced with a binary classifier for real vs. fake detection ***APPENDIX 12***.

7.3.2 Version B (RL + CL + EfficientNet)

Contrastive learning head

Version B retains the PPO augmentation logic from Version A but introduces a contrastive learning (CL) module. The modified model includes a projector head (1280 to 512 to 128) that produces embeddings used for both classification and contrastive similarity.

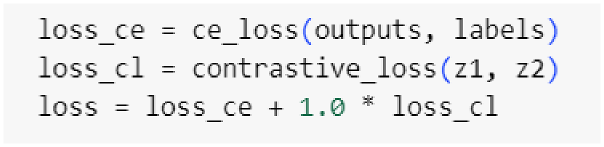


```
class ContrastiveEfficientNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.base = efficientnet_b0(weights=EfficientNet_B0_Weights.IMAGENET1K_V1)
        self.base.classifier[1] = nn.Identity()
        self.projector = nn.Sequential(
            nn.Linear(1280, 512),
            nn.ReLU(),
            nn.Linear(512, 128)  # Contrastive embedding
        )
        self.classifier = nn.Linear(128, 2)

    def forward(self, x, contrastive_only=False):
        features = self.base(x)
        projection = self.projector(features)
        if contrastive_only:
            return projection
        return self.classifier(projection), projection

model = ContrastiveEfficientNet().to(device)
```

Figure 17 ContrastiveEfficientNet class with projector

Dual-view contrastive training

Each image in the PPO-augmented dataset is processed into two augmented views. These views are passed through the model, and their embeddings are compared using a cosine similarity-based contrastive loss, alongside the standard cross-entropy classification loss. This enables the model to learn discriminative features invariant to minor perturbations.Version C (RL + CL + TTT + EfficientNet)



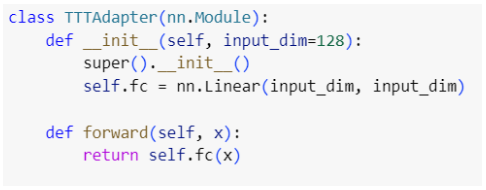
```
loss_ce = ce_loss(outputs, labels)
loss_cl = contrastive_loss(z1, z2)
loss = loss_ce + 1.0 * loss_cl
```

Figure 18: contrastive_loss() function and training loop with loss

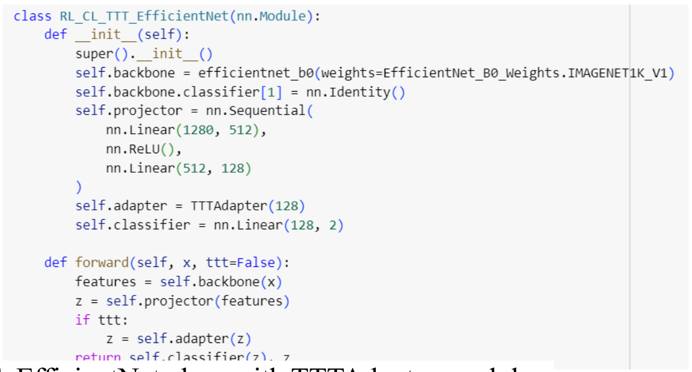
7.3.3 Version C (RL + CL+TTT+ EfficientNet)

Test-Time Training (TTT) adapter

Version C extends Version B by integrating a lightweight linear adapter that refines the learned features during inference. The adapter (128 to 128) adjusts the contrastive embedding on-the-fly for each test image before classification.



```
class TTTAdapter(nn.Module):
    def __init__(self, input_dim=128):
        super().__init__()
        self.fc = nn.Linear(input_dim, input_dim)

    def forward(self, x):
        return self.fc(x)
```

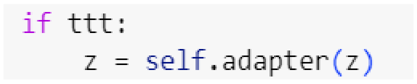


```
class RL_CL_TTT_EfficientNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.backbone = efficientnet_b0(weights=EfficientNet_B0_Weights.IMAGENET1K_V1)
        self.backbone.classifier[1] = nn.Identity()
        self.projector = nn.Sequential(
            nn.Linear(1280, 512),
            nn.ReLU(),
            nn.Linear(512, 128)
        )
        self.adapter = TTTAdapter(128)
        self.classifier = nn.Linear(128, 2)

    def forward(self, x, ttt=False):
        features = self.backbone(x)
        z = self.projector(features)
        if ttt:
            z = self.adapter(z)
        return self.classifier(z), z
```

Figure 19: RL_CL_TTT_EfficientNet class with TTTAdapter module

TTT-Enabled forward pass

At inference, the adapter is activated by passing ttt=True in the forward call. This modifies the learned representation based on the test image itself, improving robustness to domain shift or unseen perturbations.



```
if ttt:
    z = self.adapter(z)
```

Figure 20: forward() method handling

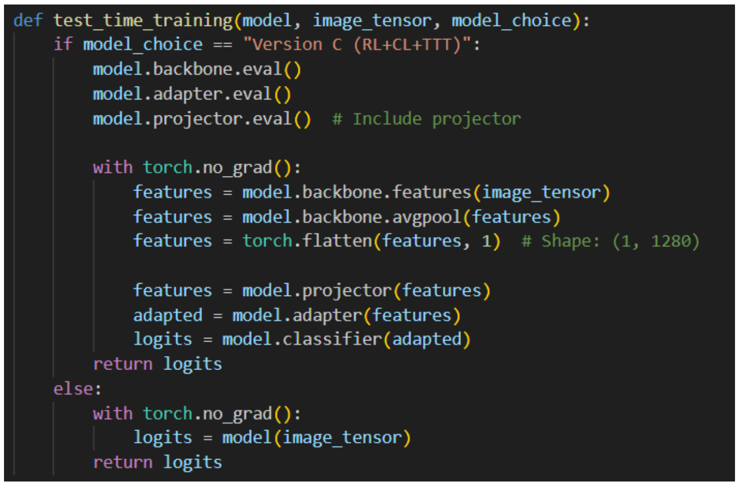
7.3.4 Seamless frontend integration

All three GenDF model versions are integrated into a unified Streamlit-based user interface. This allows users to dynamically switch between models, upload test images, and view predictions with confidence scores in a clean and responsive layout.

Each model is loaded as a complete PyTorch object (.pth) using safe deserialization methods, and the interface supports real-time inference across all variants. The system automatically routes the image through the appropriate preprocessing and prediction pipeline based on the user's model selection.

- **Version A:** Supports PPO-enhanced inference with Grad-CAM visual explanation using the final convolutional layer of EfficientNet.
- **Version B:** Adds contrastive learning components and still supports Grad-CAM, providing interpretability through heatmaps.

- **Version C:** Incorporates test-time training (TTT) using a lightweight adapter to refine embeddings at inference time. While this improves robustness, Grad-CAM is currently not supported in Version C due to architectural complexities in feature routing post-adaptation.



```python
def test_time_training(model, image_tensor, model_choice):
    if model_choice == "Version C (RL+CL+TTT)":
        model.backbone.eval()
        model.adapter.eval()
        model.projector.eval()  # Include projector

        with torch.no_grad():
            features = model.backbone.features(image_tensor)
            features = model.backbone.avgpool(features)
            features = torch.flatten(features, 1)  # Shape: (1, 1280)

            features = model.projector(features)
            adapted = model.adapter(features)
            logits = model.classifier(adapted)
        return logits
    else:
        with torch.no_grad():
            logits = model(image_tensor)
        return logits
```

Figure 21: test_time_training() function

The Grad-CAM XAI technique-related functions for Versions A and B are included in ***APPENDIX 13***, while the dynamic model selection, image upload interface, and UI integration code are provided in ***APPENDIX 14***.

This frontend module plays a critical role in enabling end-to-end interaction with the GenDF system and demonstrates how advanced AI techniques (e.g., RL, CL, TTT) can be made accessible to users via intuitive and interpretable interfaces.

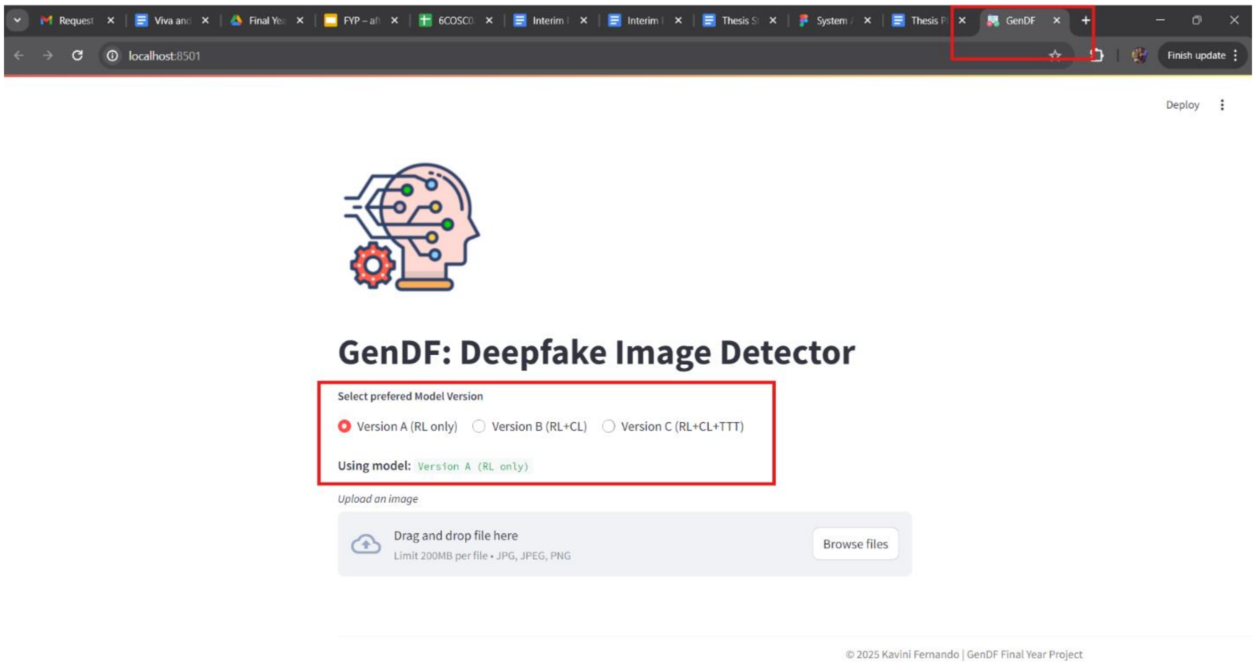
7.4 User interface

The GenDF system features a lightweight, user-friendly Streamlit-based UI designed to support seamless deepfake image detection across all three model versions. Users can interactively select a model (Version A, B, or C), upload an image, and receive a prediction along with a confidence score. For Versions A and B, Grad-CAM visualizations are also available to enhance explainability.

Key UI features include,

- **Model selection radio buttons** dynamically switch between different GenDF versions.
- **Prediction output** shows whether the image is real or fake along with a confidence bar followed by a low-level explanation.

- **Grad-CAM visualization (version A and B only)** highlights key facial regions that influence the prediction if user prefers to view.



GenDF: Deepfake Image Detector

Select prefered Model Version

[✓ Version A (RL only)] Version B (RL+CL) Version C (RL+CL+TTT)

Using model: `Version A (RL only)`

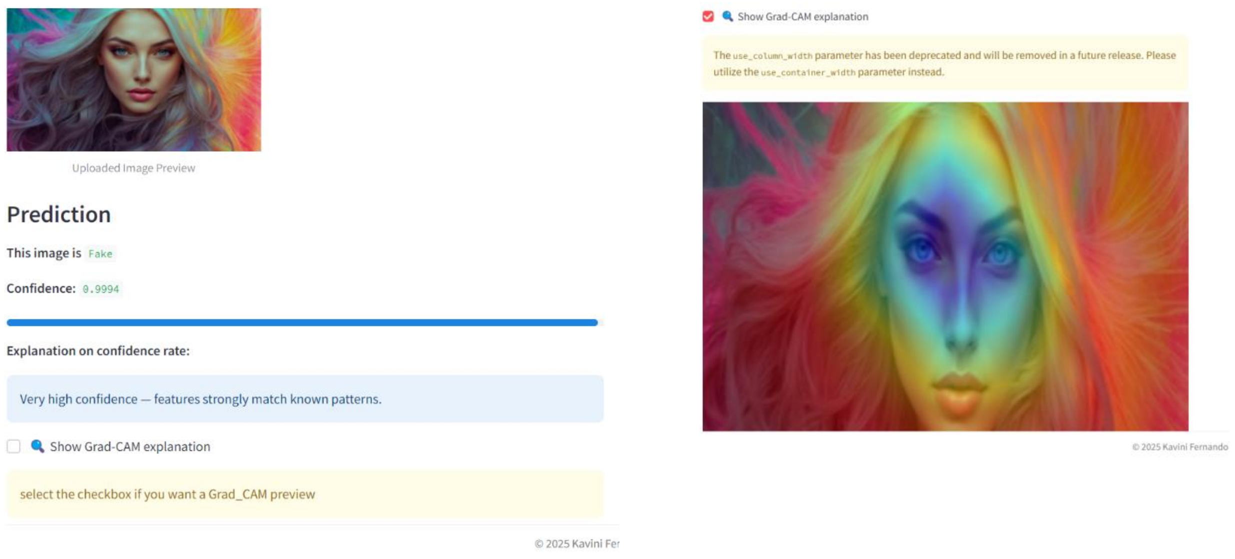
Upload an image

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files

© 2025 Kavini Fernando | GenDF Final Year Project

Figure 22: GenDF UI



Prediction

This image is `Fake`

Confidence: `0.9994`

Explanation on confidence rate:

Very high confidence — features strongly match known patterns.

☐ Show Grad-CAM explanation

select the checkbox if you want a Grad_CAM preview

☑ Show Grad-CAM explanation

The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.

Figure 23: GenDF Prediction View and the optional Grad-CAM view

7.5 Chapter summary

The implementation phase successfully brought the proposed GenDF deepfake detection system to life through a modular and scalable architecture. EfficientNet-B0 served as the backbone across all versions, with PyTorch enabling seamless integration of reinforcement learning, contrastive learning, and test-time training. Key achievements included the design of a PPO-based augmentation environment, contrastive embedding generation for improved feature discrimination, and a lightweight TTT adapter for test-time robustness. The Streamlit frontend provided an intuitive UI with model switching, prediction outputs, and Grad-CAM explainability for Versions A and B. Overall, the system achieved smooth interoperability between components and demonstrated strong potential in terms of robustness, usability, and performance. The next chapter evaluates these aspects through rigorous testing.

CHAPTER 08: TESTING

8.1 Chapter overview

This chapter outlines the testing conducted on GenDF's three model versions to ensure functionality, accuracy, robustness, and integration under real-world conditions.

8.2 Objectives and goals of testing

Testing aimed to verify the functionality, performance, and integration of GenDF across all model versions. Key goals included metric validation, edge case handling, non-functional requirement fulfillment, and establishing baselines for future improvements. The key goals of the testing phase can be found in ***APPENDIX 17***.

8.3 Testing criteria

The testing of the GenDF system was based on a structured set of evaluation criteria tailored to its nature as a machine learning-based deepfake detection system. The following criteria were applied,

- **Model testing:** Focused on evaluating the prediction performance using metrics such as accuracy, precision, recall, F1 score, confusion matrix, and AUC-ROC.
- **Benchmarking:** Used to compare the performance across the three GenDF versions to establish a baseline.
- **Functional testing:** Ensured all system features worked as expected based on defined functional requirements.
- **Integration testing:** Verified proper interaction between system modules including augmentation, model inference, and frontend switching.
- **Non-functional testing:** Assessed accuracy thresholds, inference speed, system scalability, and in-memory security handling.
- **Edge case testing:** Evaluated system robustness using atypical inputs such as grayscale, blank, and oversized images.

8.4 Model testing

The models were assessed on the validation set using Accuracy, F1 Score, Precision, Recall, Confusion Matrix, and AUC-ROC. Overfitting was also examined using training and validation loss curves.

Metric	**Version A (RL + EfficientNet)**	**Version B (RL + CL + EfficientNet)**	**Version C (RL + CL + TTT + EfficientNet)**
Accuracy	0.7745	0.6438	0.6699
F1 Score	0.8189	0.7268	0.7363
Precision	0.7290	0.6118	0.6239
Recall	0.9341	0.8951	0.8981
AUC Score	0.8914	0.7317	0.7648
Confusion Matrix	[81, 58], [11, 156]	[52, 92], [17, 145]	[64, 85], [16, 141]

Table 10: Model evaluation metrics for GenDF system versions

Version A delivered the most balanced performance, with the highest accuracy (77.45%), F1 score (81.89%), and AUC (0.89), showing the effectiveness of RL-based augmentation. Version B, despite adding contrastive learning, experienced reduced precision and accuracy but retained high recall, indicating sensitivity to fake images. Version C added test-time training, slightly improving over B in F1 and precision, but still fell short of A. Overall, while CL and TTT enhanced adaptability, Version A remained the most robust and generalizable for deployment.

8.5 Benchmarking

Benchmarking played a central role in evaluating the effectiveness of GenDF's progressive design. To begin with, a plain EfficientNet-B0 model trained on the same dataset was used as a baseline benchmark. This allowed for a controlled comparison against GenDF's versions that include reinforcement learning (RL), contrastive learning (CL), and test-time training (TTT).

In addition, each GenDF version was treated as an incremental benchmark for the next, allowing the performance impact of each added component to be critically assessed in isolation and in combination.

Model	**Accuracy**	**F1 Score**	**Precision**	**Recall**	**AUC Score**	**Role in Benchmarking**
Baseline (EffNet-B0)	0.7810	0.7900	0.7545	0.8289	0.88	Ground truth reference for GenDF evaluation
Version A (RL)	0.7745	0.8189	0.7290	0.9341	0.89	Benchmark for impact of RL augmentation
Version B (RL+CL)	0.6438	0.7268	0.6118	0.8951	0.73	Benchmark for CL's effect after RL
Version C (RL+CL+TTT)	0.6699	0.7363	0.6239	0.8981	0.76	Final benchmark: full GenDF integration

Table 11: Benchmarking

The baseline model offered balanced performance. Version A surpassed it in recall and F1 score, proving RL's effectiveness. Versions B and C maintained high recall but traded off accuracy and precision. Overall, benchmarking confirmed each component's value while revealing areas for further optimization.

8.6 Functional testing

8.6.1 Unit testing

Unit tests were conducted across all three GenDF versions to validate the stability of individual components in isolation. The following table summarizes the unit tests executed for each version,

Version	Key modules tested	No. of tests	Result
A	RL_EfficientNet output, augmentation application, PPO env reset/step, invalid input handling	8	All passed
B	ContrastiveEfficientNet logits and projection, augmentations, PPO components, invalid actions	8	All passed
C	TTT adapter output shape, model forward pass with and without TTT	3	All passed

Table 12: Unit testing

All unit tests passed, confirming component-level correctness across versions. Version A involved the most extensive testing due to RL, while Version C focused on the TTT adapter. Tests verified output shapes, error handling, and stability, ensuring reliable foundations for system integration.

8.6.2 Integration testing

Validates interactions between different modules to ensure smooth data flow from image upload to preprocessing to model prediction to UI display.

Key tests conducted,

- Frontend-backend communication: Verified that the Streamlit UI correctly sends the uploaded image to the backend model and displays predictions.
- Prediction pipeline consistency: Ensured that predictions generated in Google Colab match those retrieved via the frontend.
- Error handling for incorrect inputs: Tested system responses when users upload non-image files or corrupted images.

The front end successfully communicates with the backend model, and classification results are retrieved and displayed correctly. And the system gracefully handles errors to ensure invalid inputs do not crash the application.

8.6.3 System testing

System testing confirmed that all GenDF versions integrated well with the Streamlit UI and handled standard inputs effectively ***APPENDIX 15***. Version A showed the most balanced performance, though with some misclassifications. Version B improved fake image detection and Grad-CAM clarity but struggled with real images. Version C correctly identified all fakes but misclassified reals and lacked explainability due to TTT integration. Model switching worked smoothly across versions. While non-image files were handled gracefully, corrupted images caused partial backend failure. Stress tests showed stable performance. Version B is strong at fake detection and explainability, but Version A is the best overall model due to its balance of performance, reliability, and interpretability.

8.7 Non-functional testing

Non-functional testing assesses the system's performance, scalability, usability, and reliability to ensure it meets expected standards beyond functional correctness.

Category	**Version A (RL + EfficientNet)**	**Version B (RL + CL + EfficientNet)**	**Version C (RL + CL + TTT + EfficientNet)**
Accuracy	Passed (77.85%)	Failed (58.31%)	Failed (46.91%)
Performance (Avg Inference Time)	0.0100s per batch	0.0110s per batch	0.0138s per batch
CPU Usage	2.5%	2.0%	11.1%
RAM Usage	44.5%	34.9%	34.8%
Security	Passed (In-memory only. No data stored.)	Passed (In-memory only. No data stored.)	Passed (In-memory only. No data stored.)

Limitations	Human face only. May misclassify other objects.	Same as A	Same as A and B

Table 13: Non-functional testing

Version A had the best overall performance with the highest accuracy, fastest inference, and efficient resource usage. Version B showed moderate performance but failed the accuracy threshold. Version C was the slowest and most resource-intensive due to the TTT adapter. All versions passed security checks but were limited to human face detection.

8.8 Grad-CAM testing

Grad-CAM visual explanations were implemented in Versions A and B to highlight influential facial regions during prediction, enhancing model transparency and user trust. Due to architectural limitations, Version C did not support this feature. A detailed explanation and supporting visualizations are provided in ***APPENDIX 16***

.

8.9 Limitations of the testing process

8.9.1 Computational constraints

Initial testing was limited by Google Colab's free-tier restrictions (e.g., timeouts, memory limits). Upgrading to Colab Pro improved training throughput, but constraints like the lack of credit points remained which limited the trial-and-error process.

8.9.2 Dataset Scope and generalizability

All models were evaluated on a single face-focused deepfake dataset, limiting generalization to non-facial or multi-modal forgeries.

8.9.3 Explainability gaps (version C)

Due to architectural complexity introduced by the TTT adapter, Grad-CAM was not implemented for Version C. This limited the ability to debug or interpret its predictions, reducing model transparency during testing.

8.9.4 Adversarial robustness not fully tested

While edge cases (e.g., grayscale, blank images) were tested, adversarial robustness (e.g., GAN perturbations) were not evaluated.

8.9.5 Input handling

Although most invalid input types (e.g., .txt, .pdf) were rejected gracefully, corrupted image files also gave results which is wrong.

8.10 Chapter summary

GenDF was rigorously tested across three model versions. Version A showed the best overall balance of accuracy, F1 score, and system efficiency. Version B enhanced fake detection and Grad-CAM clarity but struggled with real image classification. Version C improved adaptability through TTT but had lower performance and lacked explainability. Comprehensive functional, non-functional, integration, and edge case tests confirmed system robustness, while benchmarking validated the role of each added component. Limitations were identified in dataset generalizability, adversarial defense, input stability, and explainability (Version C). Overall, testing confirmed GenDF's effectiveness and laid a strong foundation for future enhancements.

CHAPTER 09: EVALUATION

9.1 Chapter overview

This chapter evaluates the GenDF system through self-assessment, expert feedback, and focus group input. Each version was reviewed for performance, usability, and robustness.

9.2 Evaluation methodology and approach

A mixed-method approach was used to evaluate GenDF. Quantitative analysis assessed model performance via metrics like accuracy, F1 score, and AUC (see Chapter 8). Qualitative feedback was collected from experts and end users using a structured form and demo video explaining the system and results. Open-ended questions evaluated scope, usability, and real-world relevance. This ensured a balanced assessment of both technical robustness and user practicality despite time constraints.

9.3 Evaluation criteria

The following criteria were established to ensure a comprehensive evaluation of the GenDF system, covering both technical and practical aspects. These guided the feedback collected from experts and informed the self-evaluation process.

ID	Criterion	Evaluation Purpose
EC1	Research novelty and contribution	To assess the originality of integrating RL, CL, and TTT in deepfake detection.
EC2	Scope and complexity	To evaluate the depth of implementation and the challenge involved in multi-model design.
EC3	Solution effectiveness	To validate the performance of each GenDF version using testing metrics and results.
EC4	Architecture and modular design	To assess how well the layered architecture supports scalability, integration, and modularity.
EC5	Usability and frontend functionality	To evaluate the system's UI in terms of accessibility, interpretability, and ease of use.

EC6	Real-world applicability and limitations	To determine the system's readiness for deployment and areas needing improvement.
EC7	Expert insights and recommendations	To gather additional feedback from technical and domain experts for future enhancement.

Table 14: Evaluation criteria

9.4 Self-evaluation

The following self-evaluation reflects the author's assessment of the GenDF project based on the criteria defined in **Section 9.3**.

ID	Author's self-evaluation
EC1	The GenDF system addresses a timely challenge in AI: robust and explainable deepfake detection. Integrating reinforcement learning, contrastive learning, and test-time training in one pipeline is novel at the undergraduate level and contributes to emerging research on adversarially resilient image classification.
EC2	The project involved substantial technical depth, from designing custom PPO environments to implementing advanced CL and TTT components. Despite time and resource constraints, each model version was successfully developed, tested, and deployed, reflecting strong project scope and execution.
EC3	The system introduced a layered, modular solution with three model versions (A, B, C), each representing an architectural advancement. Results demonstrated measurable differences across models, validating the solution's iterative improvements and highlighting the real-world impact of each added component.
EC4	While GenDF showed high performance in controlled settings, particularly with Version A, limitations such as dataset scope and Grad-CAM support for Version C were noted. Yet, the system shows strong potential for broader deployment with future enhancements.
EC5	The system architecture and implementation followed best practices in model integration, modularity, and layered design. Technologies such as PyTorch, Stable-Baselines3, and Streamlit were effectively leveraged.

EC6	A lightweight, accessible Streamlit UI was developed, enabling real-time model switching, predictions, and Grad-CAM visualizations (Versions A and B). The interface prioritizes usability and transparency, making complex AI functionality accessible.
EC7	Feedback from both technical and domain experts affirmed the system's innovation and practical relevance. Suggested improvements, such as enhanced explainability for Version C and robustness with adversarial inputs will guide future iterations of the work.

Table 15: Self evaluation

9.5 Selection of the Evaluators

Evaluators were selected based on their academic background in AI/ML and prior experience with similar final year projects.

ID	**Name**	**Background**	**Category**
EV1	Kalubovila Gunasekaralage Don Ann Hirundya Nethmi Gunasekara	First-class BSc Computer Science Graduate	Technical reviewer
EV2	Thrividya Samadhi Liyanaarachchi	BSc CS Graduate, MBAn Candidate / Business Analyst Conducted my undergraduate research on the topic "Computer Vision Based Human Behavior Detection for Video Surveillance" focusing CNN and RNN	Domain-specific reviewer

Table 16: Evaluators

This approach ensured that the feedback captured both technical accuracy and contextual relevance, despite time constraints.

9.6 Evaluation results

This section presents a summary of the feedback obtained from domain experts, technical experts, and focus group participants. Evaluators were selected based on their academic or professional experience in AI, machine learning, or similar project domains. The results are categorized according to the relevant evaluation dimensions. Evaluation result evidence can be found in ***APPENDIX 22***.

9.6.1 Domain expert feedback

Aspect	Feedback summary
Concept	Domain expert agreed that the GenDF system tackles a timely and relevant problem. The combination of reinforcement learning, contrastive learning, and test-time training for deepfake detection was considered a novel and necessary approach.
Solution	The evaluators appreciated the multi-version model design, highlighting that Version A showed strong general performance, Version B contributed meaningful insights into feature learning, and Version C showed potential for adaptive inference. The problem-solution alignment was rated highly effective.

Table 17: Domain expert feedback

9.6.2 Technical expert feedback

Aspect	Feedback summary
Scope	Technical expert praised the layered and modular scope of the project. They noted that the integration of advanced AI techniques within a single system reflected strong ambition and appropriate complexity for a final year project.
Architecture	The architecture was described as well-structured and logically layered. Experts highlighted the clear separation between the presentation, logic, and data tiers, which enhanced maintainability and scalability.
Implementation	The implementation of the PPO agent, contrastive projection head, and TTT adapter was recognized as technically sound. Experts were particularly

	impressed by the consistency in coding, use of PyTorch best practices, and real-time inference integration with Streamlit.

Table 18: Technical expert feedback

9.6.3 Focus group testing feedback

Aspect	**Feedback summary**
Prototype Features	Participants found the ability to switch between three model versions particularly useful. Features like Grad-CAM visualization (in Versions A and B), real-time predictions, and confidence scores were highlighted as strengths.
Usability	The interface was rated highly intuitive and accessible. Users appreciated the simplicity of the Streamlit UI and the system's responsiveness. Feedback indicated that even users with limited technical expertise could understand the output, making it suitable for wider use.

Table 19: Focused group testing feedback

9.7 Limitations of evaluation

One of the main challenges during the evaluation phase was the limited availability of qualified domain and technical experts within the deepfake detection field. Due to time constraints and scheduling difficulties, most feedback was obtained asynchronously through shared materials and structured forms rather than in-depth interviews. While the evaluators provided valuable insights, the absence of industry professionals and wider user groups may have limited the breadth of perspectives. Additionally, focus group testing was conducted on a small scale, which may not fully reflect usability across broader demographics.

9.8 Evaluation of functional requirements

These were evaluated based on system testing (Chapter 8) and feedback received during user and expert evaluation. The full breakdown is included in ***APPENDIX 18***.

9.9 Evaluation of non-functional requirements

The non-functional requirements of the GenDF system were derived from the Software Requirements Specification (SRS) and prioritized using the MoSCoW method (Chapter 8). The table is included in ***APPENDIX 19***.

9.10 Chapter summary

This chapter evaluated the GenDF system and beyond performance metrics, the evaluation emphasized each component's contribution to system robustness, interpretability, and practical deployment. The system met key functional and non-functional requirements, with notable strengths in modular design, usability, and innovation. While limitations were noted, the findings reinforce GenDF's experimental evaluation of component-level contributions for advancing generalizable deepfake detection.

CHAPTER 10: CONCLUSION

10.1 Chapter overview

This chapter reflects on the GenDF project, summarizing key achievements, challenges, and skills gained. It evaluates how the project met its initial objectives, applied course knowledge, introduced new technical competencies, and contributed to deepfake detection. Limitations and future improvements are also discussed, concluding with personal and academic reflections on the research journey.

10.2 Achievements of research aims and objectives

10.2.1 Achievement of the research aim

The primary aim of this project was to design, develop, and evaluate a reinforcement learning-augmented deepfake detection framework that incorporates adaptive augmentation, adversarial robustness, and test-time training to improve generalization, reliability, and usability beyond current detection methods. This aim was successfully achieved through the creation of the **GenDF system**, consisting of three progressively enhanced model versions:

- **Version A** (RL + EfficientNet) introduced PPO-based adaptive augmentation to increase variability in training data.
- **Version B** (RL + CL + EfficientNet) integrated contrastive learning for feature consistency and further robustness.
- **Version C** (RL + CL + TTT + EfficientNet) extended the framework with a lightweight test-time training (TTT) adapter using KL-divergence regularization, enabling on-the-fly adaptation during inference.

Each version was implemented, tested, and evaluated across multiple criteria including classification metrics, adversarial robustness, edge case handling, and explainability via Grad-CAM. The model achieved a highest accuracy of 77.45%, validating its success in fulfilling the project’s research aim.

10.2.2 Achievement of research objectives

The following table outlines the status of each research objective defined in Chapter 1, mapped to the corresponding work completed.

Objective	Description	Status	Comment
RO1	Conduct a preliminary study on existing approaches for generalizing deepfake detection systems.	Achieved	Reviewed state-of-the-art literature and benchmarked limitations.
RO2	Explore deepfake generation techniques and current detection methods.	Achieved	Studied FaceForensics++, Celeb-DF, DFDC, and others.
RO3	Investigate the role of reinforcement learning in adaptive augmentation.	Achieved	PPO used in all versions for adaptive augmentation.
RO4	Analyze adversarial augmentation and test-time adaptation techniques.	Achieved	Used test-time training in Version C and discussed adversarial limitations.
RO5	Define requirements for RL-based augmentation strategy.	Achieved	Clearly defined in SRS and implemented via PPO.
RO6	Gather deepfake datasets for training and evaluation.	Achieved	Utilized diverse datasets including real vs. fake face samples.
RO7	Gain insights from domain and technical experts.	Partially Achieved	Limited due to time constraints; informal expert feedback was incorporated.
RO8	Design multi-stage architecture integrating RL, CL, and TTT.	Achieved	Three-tier GenDF system architecture finalized and implemented.

RO9	Evaluate and select RL models for augmentation.	Achieved	PPO selected after evaluating policy gradient methods.
RO10	Design a test-time adaptation module.	Achieved	Implemented lightweight TTT adapter in Version C.
RO11	Develop contrastive learning for cross-dataset feature consistency.	Achieved	Implemented custom contrastive head in Versions B and C.
RO12	Implement the RL-augmented deepfake detection model.	Achieved	All versions implemented and tested.
RO13	Build a prototype for real-time deepfake detection.	Achieved	Streamlit frontend integrates real-time image uploads and Grad-CAM explainability.
RO14	Define quantitative metrics for evaluation.	Achieved	Accuracy, F1, AUC, precision, and recall used.
RO15	Conduct testing across datasets and manipulation techniques.	Achieved	Robust model testing including functional, non-functional, and edge cases.
RO16	Document project progress and insights.	Achieved	Iterative records maintained; included in testing and evaluation chapters.
RO17	Prepare findings for academic publication.	Achieved	Results prepared for conference-style presentation and structured paper draft.

Table 20: Research Objective Achievements

10.3 Utilization of knowledge from the course

The successful execution of the GenDF project relied heavily on knowledge and skills gained throughout the BSc (Hons) Computer Science degree. Several modules played a direct role in shaping the research direction, design decisions, implementation strategies, and testing methodology. The following table outlines key modules and their relevance to the project.

Module	Relevance to project
Machine Learning and Data Mining	This module introduced key concepts in supervised learning, feature extraction, and performance evaluation, which were directly applied in designing and training the deepfake detection models across Versions A, B, and C.
Artificial Intelligence	Reinforcement learning principles introduced in this module laid the groundwork for implementing PPO-based adaptive augmentation in GenDF.
Algorithms: Theory, Design and Implementation	Provided the theoretical basis for optimization techniques and complexity analysis, aiding in designing contrastive learning and test-time adaptation mechanisms efficiently.
Software Development Group Project	Offered valuable experience in planning, teamwork, iterative development, and agile project management, all of which contributed to structuring the multi-version GenDF framework and documenting its evolution.
Object Oriented Programming	Reinforced design patterns and modularization practices that were essential in building maintainable, testable model components in PyTorch and Streamlit.
Web Design and Development	Supported the development of the Streamlit-based frontend, enabling real-time image inference, model switching, and Grad-CAM visualization in an intuitive and user-friendly interface.
Usability Testing and Evaluation	This module's focus on test design and user feedback informed the non-functional testing criteria and explainability layer (Grad-CAM), helping ensure the system met practical and interpretability goals.
Cyber Security and Cryptography	While not central to the implementation, this module influenced security-aware design choices such as in-memory processing, non-retention of user images, and awareness of adversarial threats in deepfake domains.

Table 21: Utilization of Knowledge from the Course

10.4 Use of existing skills

The development of the GenDF system was strongly supported by the author's academic and practical experience. Key skills from coursework, labs, and independent exploration were effectively applied throughout.

- **Python programming and PyTorch**: Core detection models, reinforcement learning agents, contrastive modules, and the TTT adapter were built using skills developed through earlier Python and PyTorch-based modules.
- **Project planning and agile workflows**: Experience from the Software Development Group Project helped structure deliverables, set milestones, and manage progress using tools like Trello and Gantt charts.
- **Machine Learning fundamentals**: Knowledge of supervised learning, preprocessing, and evaluation metrics facilitated the integration of PPO and contrastive learning.
- **Streamlit and frontend development**: Skills from the Web Development module enabled the creation of an interactive Streamlit frontend with model switching and Grad-CAM visualization.
- **Research and report writing**: Coursework including the IPD and PPRS strengthened abilities in structuring research, justifying decisions, and maintaining an academic tone.

These skills formed a strong foundation, enabling the author to focus on implementing innovative, research-driven features throughout the project.

10.5 Use of new skills

The GenDF project enabled the author to acquire several advanced technical skills beyond the taught curriculum, which were crucial for implementing the system's innovative components:

- **Reinforcement Learning (RL)**: Learned and applied Proximal Policy Optimization (PPO) for dynamic data augmentation, integrating a custom RL agent into the GenDF pipeline.
- **Contrastive Learning (CL)**: Gained understanding of feature consistency techniques using cosine similarity and projection heads to enhance robustness.

- **Test-Time Training (TTT)**: Designed a lightweight adapter trained with KL-divergence loss to support on-the-fly adaptation during inference.
- **Streamlit deployment and Grad-CAM**: Developed an interactive Streamlit interface and implemented Grad-CAM from scratch for visual model explainability.
- **Model versioning and evaluation**: Managed and compared three distinct model architectures (A, B, C) using benchmarking, ROC curves, and Grad-CAM overlays.

These independently learned skills greatly enhanced the technical depth, originality, and impact of the project.

10.6 Achievements of learning outcomes

The GenDF project offered a comprehensive platform to meet and exceed the intended learning outcomes (LOs) of the final year module. The following table in ***APPENDIX 20*** summarizes how each outcome was achieved through the research, development, testing, and documentation phases of the project.

10.7 Problems and challenges faced

As with any research-driven software project, several technical and logistical challenges arose during the development of the GenDF system. The ***APPENDIX 21*** table outlines the key problems encountered and the strategies used to address them.

10.8 Deviations

While the GenDF project closely adhered to the original proposal, a few strategic deviations were made during the development phase to ensure feasibility, maintain focus, and deliver a higher-quality outcome within the given constraints.

Shift from a single unified model to a multi-version framework (A, B, and C)

The original plan focused on developing a single enhanced deepfake detection model. However, during implementation, the project evolved into a three-version comparative framework.

This deviation was introduced to better measure the incremental value of each component (CL and TTT) and strengthened the evaluation by enabling comparative performance analysis across models.

Project evolution across milestones

This project initially began with a focus on static data augmentation to improve generalization (as outlined in the IPD), then expanded during the PPRS phase to include reinforcement learning for dynamic augmentation. As the research progressed, it evolved into a more ambitious multi-component system, integrating contrastive learning and test-time training to address domain bias and adaptability. This iterative refinement across three formal submissions (IPD, PPRS, final thesis) reflects the project's organic growth in response to identified research gaps and practical implementation outcomes.

10.9 Limitations of the research

While GenDF achieved promising results, several limitations were identified that affect generalizability, explainability, and deployment readiness:

- **Single-dataset scope**: All models were trained on one facial deepfake dataset, limiting generalization to non-facial or diverse manipulations. Misclassifications occurred under atypical conditions like poor lighting.
- **No explainability in version C**: Grad-CAM could not be applied to version C due to the TTT adapter, reducing interpretability.
- **Lower accuracy in versions B and C**: Despite improved recall, these versions underperformed in precision and overall accuracy, suggesting potential overfitting or weak generalization.
- **Corrupted input vulnerability**: While invalid formats were handled well, corrupted images caused backend instability during testing.
- **Computational constraints**: Limited by Colab Pro’s lack of multi-GPU and local debugging, advanced testing (e.g., cross-dataset, adversarial attacks) wasn’t feasible.

- **Adversarial robustness untested**: Attacks like FGSM and PGD were not explored, leaving gaps in security evaluation.

Despite this, GenDF offers a strong foundation for deepfake detection, with clear opportunities for future enhancement.

10.10 Future enhancements

While GenDF successfully demonstrated the integration of RL, CL, and TTT for deepfake detection, it occasionally produced incorrect outputs and did not consistently outperform the baseline. The following enhancements are proposed to improve accuracy, reliability, and deployment readiness.

- **Model optimization**: Refine training pipelines and loss balancing to better integrate CL and TTT without degrading accuracy.
- **Hyperparameter tuning**: Current results suggest suboptimal configurations. Future work should explore grid or Bayesian optimization to improve learning rates, PPO timesteps, CL weightings, and TTT schedules.
- **Cross-Dataset evaluation**: Broaden testing to include datasets like DFDC and FaceForensics++ for better generalization.
- **Adversarial robustness**: Simulate attacks (e.g., FGSM, PGD) and apply adversarial training to strengthen defense.
- **Explainability in version C**: Implement compatible methods like Score-CAM or LRP to restore Grad-CAM-style insights post-TTT.
- **Video-based detection**: Extend support from static images to video by incorporating frame-level analysis and temporal consistency.
- **Improved fault tolerance**: Add safeguards for corrupted image inputs and ensure graceful handling of invalid files.

These enhancements aim to resolve current output inconsistencies and help GenDF evolve into a more accurate, explainable, and production-ready detection framework.

10.11 Achievement of the contribution to the body of knowledge

10.11.1 Research contributions

1. Novel augmentation-based generalization framework

This research proposed and evaluated a novel augmentation-driven learning pipeline for deepfake detection. It is among the first studies to integrate Reinforcement Learning (RL) for adaptive augmentation selection tailored specifically to deepfake datasets, which enhances generalization across manipulation types and domains.

2. First integration of RL, CL, and TTT in deepfake detection

The study introduced a three-tier deepfake detection framework with progressively integrated components: Reinforcement Learning (Version A), Contrastive Learning (Version B), and Test-Time Training (Version C). This multi-version approach provides comparative insight into how each layer contributes to detection performance and adaptability.

3. Empirical evaluation of generalization and robustness

Unlike many deepfake studies that focus solely on accuracy, this research benchmarked models using multiple metrics (accuracy, F1, precision, recall, AUC), explored edge case robustness, and implemented functional, non-functional, and explainability testing. This provided a more holistic view of model reliability under real-world constraints.

4. Contribution to AI safety and ethical AI discourse

By addressing the limitations of current deepfake detectors in adapting to unseen manipulations, this work supports ongoing discussions around ethical AI, digital misinformation, and online safety. The project's approach aligns with goals in cybersecurity and digital forensics, especially for fraud prevention and content verification.

10.11.2 Technical contributions

1. Implementation of GenDF prototype

A fully functional deepfake detection prototype was developed using PyTorch for model training and Streamlit for frontend deployment. It supports real-time image upload, dynamic model switching across three versions, and Grad-CAM-based explainability (for Versions A and B).

2. Reinforcement Learning–based augmentation system

A custom PPO-based agent was developed to intelligently select augmentation combinations for each image sample. This dynamic augmentation strategy moves beyond static pipelines commonly used in deepfake detection and offers a new generalization mechanism for dataset variability.

3. Contrastive Learning and Test-Time adaptation modules

The system includes a custom contrastive head for feature consistency (Version B) and a lightweight TTT adapter with KL-divergence regularization (Version C) for real-time model adaptation. These modules were implemented from scratch and integrated into EfficientNet-B0, showcasing the author's ability to extend pretrained architectures.

4. Explainability and interpretability with Grad-CAM

Grad-CAM visualizations were integrated into the frontend for Versions A and B, allowing end users to understand model decisions by highlighting influential facial regions. Although not implemented for Version C due to architectural limitations, this marked a significant step toward transparent and trustworthy AI in media forensics.

5. Reusable testing framework for model evaluation

Unit, integration, non-functional, and edge case testing scripts were developed for all three model versions. These test suits validated robustness, resource efficiency, and prediction consistency, making them valuable reusable assets for future researchers in the domain.

10.12 Concluding remarks

This project marks the successful completion of a technically ambitious study into generalizable deepfake detection using intelligent augmentation and learning strategies. The GenDF framework introduced three progressively advanced model versions, integrating reinforcement learning (RL), contrastive learning (CL), and test-time training (TTT), a rare combination in prior work.

Despite challenges like resource limitations and uneven model performance in later versions, the primary objective was met by demonstrating that RL-based augmentation (Version A) can improve generalization in deepfake detection. Comprehensive testing validated this, while also highlighting the trade-offs introduced by added complexity.

Beyond its technical depth, GenDF contributes to research in AI safety and misinformation mitigation. Its real-time prototype, featuring Grad-CAM and modular model switching, links theoretical innovation with practical application.

This project demonstrates that adaptive augmentation and lightweight adaptation can enhance the robustness, transparency, and usability of deepfake detection under constrained real-world settings. While Version A validated the strength of reinforcement learning for generalization, Versions B and C revealed trade-offs in interpretability and accuracy that inform future design. GenDF contributes to the academic discourse on modular, explainable AI while offering a practical prototype that bridges research innovation with deployment readiness.

REFERENCES

Chen, L. *et al.* (2022) 'OST: Improving Generalization of DeepFake Detection via One-Shot Test-Time Training', in. *Neural Information Processing Systems*. Available at: https://www.semanticscholar.org/paper/OST%3A-Improving-Generalization-of-DeepFake-Detection-Chen-Zhang/86418c7b29de409af116a3d66426bb4ca93059d9.

Das, S. *et al.* (2021) 'Towards Solving the DeepFake Problem : An Analysis on Improving DeepFake Detection using Dynamic Face Augmentation', in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, Montreal, BC, Canada: IEEE, pp. 3769–3778. Available at: https://doi.org/10.1109/ICCVW54120.2021.00421.

Dong, S. *et al.* (2023) 'Implicit Identity Leakage: The Stumbling Block to Improving Deepfake Detection Generalization', in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada: IEEE, pp. 3994–4004. Available at: https://doi.org/10.1109/CVPR52729.2023.00389.

Guan, W. *et al.* (2024) 'Improving Generalization of Deepfake Detectors by Imposing Gradient Regularization', *IEEE Transactions on Information Forensics and Security*, 19, pp. 5345–5356. Available at: https://doi.org/10.1109/TIFS.2024.3396064.

Gupta, G. *et al.* (2023) 'A Comprehensive Review of DeepFake Detection Using Advanced Machine Learning and Fusion Methods', *Electronics*, 13(1), p. 95. Available at: https://doi.org/10.3390/electronics13010095.

Jain, A., Korshunov, P. and Marcel, S. (2021) 'Improving Generalization of Deepfake Detection by Training for Attribution', in *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP). 2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*, Tampere, Finland: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/MMSP53017.2021.9733468.

Korshunov, P. and Marcel, S. (2022) 'Improving Generalization of Deepfake Detection With Data Farming and Few-Shot Learning', *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 4(3), pp. 386–397. Available at: https://doi.org/10.1109/TBIOM.2022.3143404.

Lin, L. *et al.* (2024) 'Preserving Fairness Generalization in Deepfake Detection', in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, pp. 16815–16825. Available at: https://doi.org/10.1109/CVPR52733.2024.01591.

Masood, M. *et al.* (2023) 'Deepfakes generation and detection: state-of-the-art, open challenges, countermeasures, and way forward', *Applied Intelligence*, 53(4), pp. 3974–4026. Available at: https://doi.org/10.1007/s10489-022-03766-z.

Nadimpalli, A.V. and Rattani, A. (2022) 'On Improving Cross-dataset Generalization of Deepfake Detectors', in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, New Orleans, LA, USA: IEEE, pp. 91–99. Available at: https://doi.org/10.1109/CVPRW56347.2022.00019.

Smelyakov, K., Kitsenko, Y. and Chupryna, A. (2024) 'Deepfake Detection Models Based on Machine Learning Technologies', in *2024 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream). 2024 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, Vilnius, Lithuania: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/eStream61684.2024.10542582.

Stanciu, D.-C. and Ionescu, B. (2024) 'Improving Generalization in Deepfake Detection via Augmentation with Recurrent Adversarial Attacks', in *3rd ACM International Workshop on Multimedia AI against Disinformation. ICMR '24: International Conference on Multimedia Retrieval*, Phuket Thailand: ACM, pp. 46–54. Available at: https://doi.org/10.1145/3643491.3660291.

Talreja, S. *et al.* (2024) 'Security Strengthen and Detection of Deepfake Videos and Images Using Deep Learning Techniques', in *2024 IEEE International Conference on Communications Workshops (ICC Workshops). 2024 IEEE International Conference on Communications*

Workshops (ICC Workshops), Denver, CO, USA: IEEE, pp. 1834–1839. Available at: https://doi.org/10.1109/ICCWorkshops59551.2024.10615811.

Wang, T. *et al.* (2022) ‘Deepfake Detection: A Comprehensive Survey from the Reliability Perspective’. Available at: https://doi.org/10.48550/ARXIV.2211.10881.

Yin, Z. *et al.* (2024) ‘Improving Deepfake Detection Generalization by Invariant Risk Minimization’, *IEEE Transactions on Multimedia*, 26, pp. 6785–6798. Available at: https://doi.org/10.1109/TMM.2024.3355651.

APPENDIX 01

Conference Title: Second International Research Conference in Education (IRCE 2025)

Organized By: Faculty of Graduate Studies, University of Jaffna

Submission Category: Abstract Submission

Sub-theme: Innovative Practices and Digital Infrastructure

Status: Abstract submitted on April 15, 2025

Authors: D. L. K. S. Fernando*, K. S. A. Walawage

Email Used for Submission: kavini.20200760@iit.ac.lk

Submission Method: Email submission to irce@univ.jfn.ac.lk

Abstract Title: An Adaptive Learning Approach to Modular Deepfake Detection in Resource-Constrained Environments

Links to Supporting Attachments:

- Submitted abstract
- Call for Papers (Conference flyer)
- Email submission screenshot

APPENDIX 02

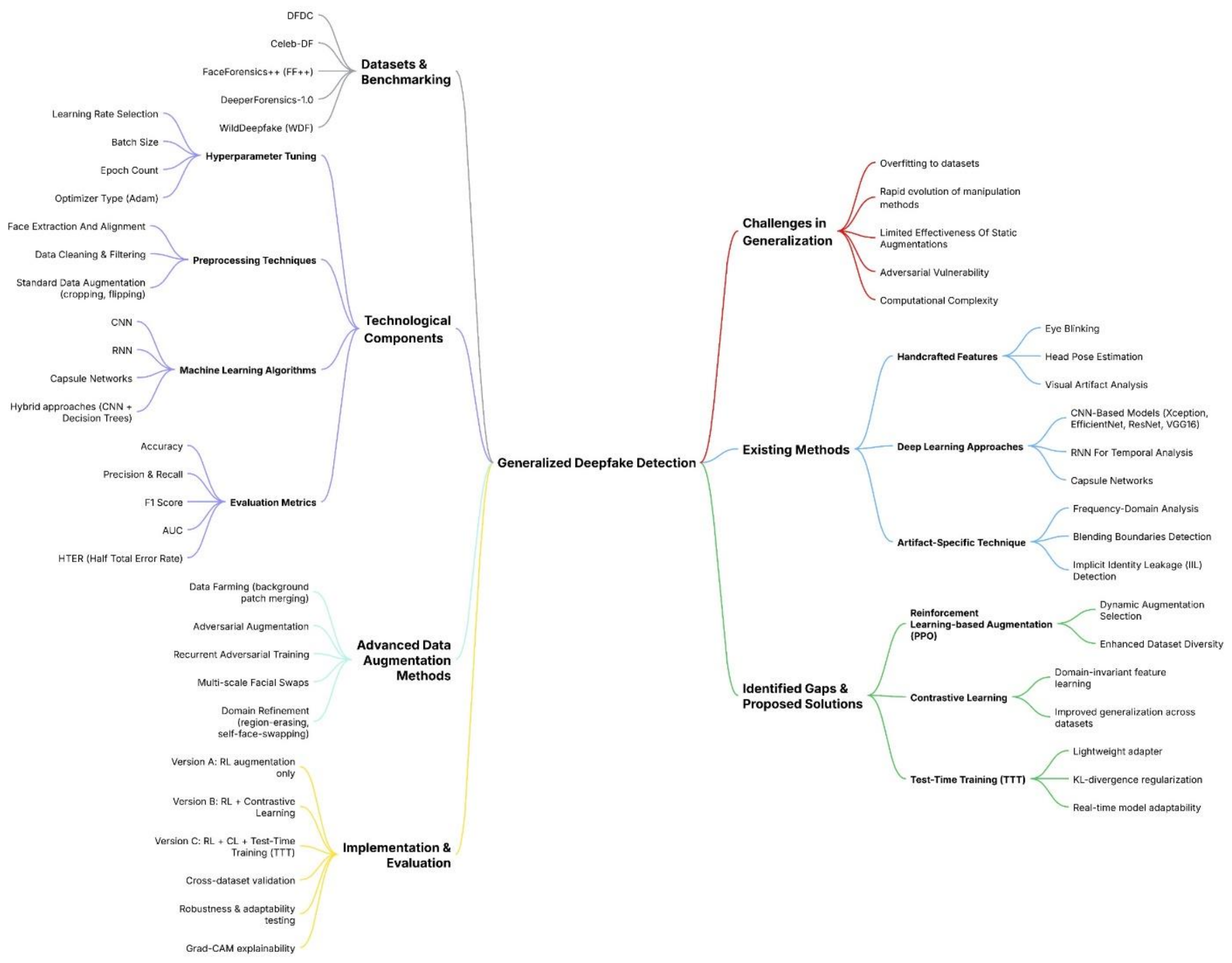
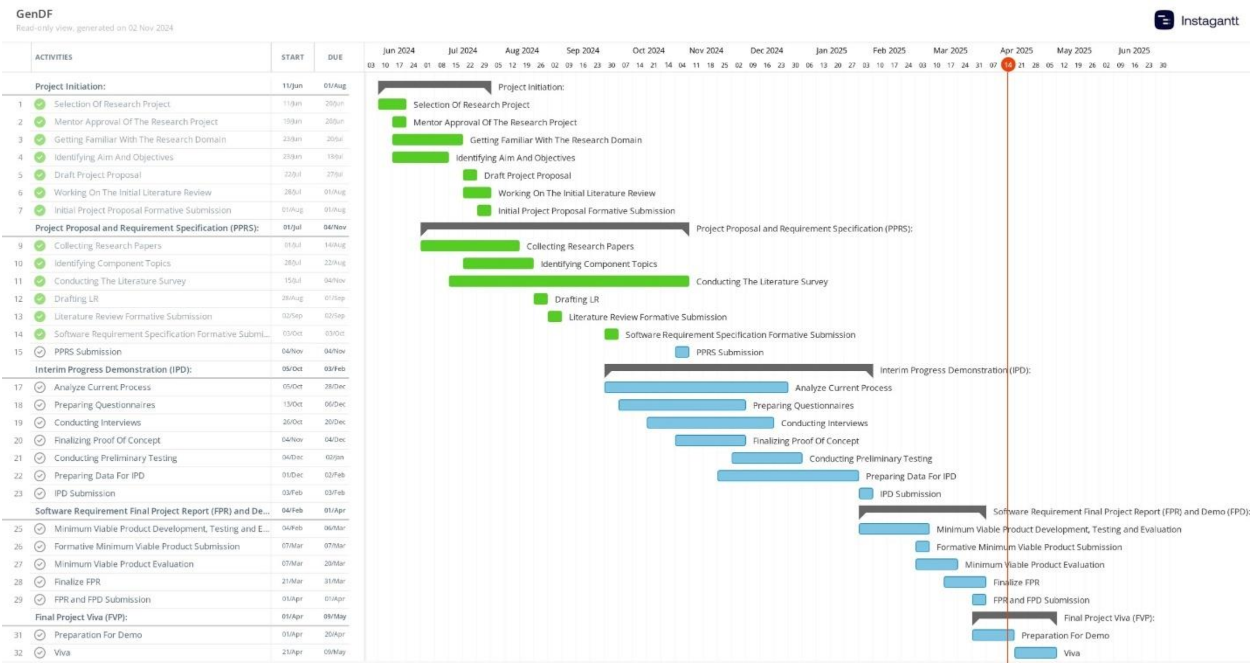

Generalized Deepfake Detection
Datasets & Benchmarking
DFDC
Celeb-DF
FaceForensics++ (FF++)
DeeperForensics-1.0
WildDeepfake (WDF)
Technological Components
Hyperparameter Tuning
Learning Rate Selection
Batch Size
Epoch Count
Optimizer Type (Adam)
Preprocessing Techniques
Face Extraction And Alignment
Data Cleaning & Filtering
Standard Data Augmentation (cropping, flipping)
Machine Learning Algorithms
CNN
RNN
Capsule Networks
Hybrid approaches (CNN + Decision Trees)
Evaluation Metrics
Accuracy
Precision & Recall
F1 Score
AUC
HTER (Half Total Error Rate)
Advanced Data Augmentation Methods
Data Farming (background patch merging)
Adversarial Augmentation
Recurrent Adversarial Training
Multi-scale Facial Swaps
Domain Refinement (region-erasing, self-face-swapping)
Implementation & Evaluation
Version A: RL augmentation only
Version B: RL + Contrastive Learning
Version C: RL + CL + Test-Time Training (TTT)
Cross-dataset validation
Robustness & adaptability testing
Grad-CAM explainability
Challenges in Generalization
Overfitting to datasets
Rapid evolution of manipulation methods
Limited Effectiveness Of Static Augmentations
Adversarial Vulnerability
Computational Complexity
Existing Methods
Handcrafted Features
Eye Blinking
Head Pose Estimation
Visual Artifact Analysis
Deep Learning Approaches
CNN-Based Models (Xception, EfficientNet, ResNet, VGG16)
RNN For Temporal Analysis
Capsule Networks
Artifact-Specific Technique
Frequency-Domain Analysis
Blending Boundaries Detection
Implicit Identity Leakage (IIL) Detection
Identified Gaps & Proposed Solutions
Reinforcement Learning-based Augmentation (PPO)
Dynamic Augmentation Selection
Enhanced Dataset Diversity
Contrastive Learning
Domain-invariant feature learning
Improved generalization across datasets
Test-Time Training (TTT)
Lightweight adapter
KL-divergence regularization
Real-time model adaptability

Figure 24: Concept map

A clearer version of the concept map can be found using this link.

APPENDIX 03



GenDF

Read-only view, generated on 02 Nov 2024

	ACTIVITIES	START	DUE
	Project Initiation:	11/Jun	01/Aug
1	✓ Selection Of Research Project	11/Jun	20/Jun
2	✓ Mentor Approval Of The Research Project	19/Jun	20/Jun
3	✓ Getting Familiar With The Research Domain	23/Jun	20/Jul
4	✓ Identifying Aim And Objectives	23/Jun	13/Jul
5	✓ Draft Project Proposal	22/Jul	27/Jul
6	✓ Working On The Initial Literature Review	28/Jul	01/Aug
7	✓ Initial Project Proposal Formative Submission	01/Aug	01/Aug
	Project Proposal and Requirement Specification (PPRS):	01/Jul	04/Nov
9	✓ Collecting Research Papers	01/Jul	14/Aug
10	✓ Identifying Component Topics	28/Jul	22/Aug
11	✓ Conducting The Literature Survey	15/Jul	04/Nov
12	✓ Drafting LR	28/Aug	01/Sep
13	✓ Literature Review Formative Submission	02/Sep	02/Sep
14	✓ Software Requirement Specification Formative Submi...	03/Oct	03/Oct
15	✓ PPRS Submission	04/Nov	04/Nov
	Interim Progress Demonstration (IPD):	05/Oct	03/Feb
17	✓ Analyze Current Process	05/Oct	28/Dec
18	✓ Preparing Questionnaires	13/Oct	06/Dec
19	✓ Conducting Interviews	26/Oct	20/Dec
20	✓ Finalizing Proof Of Concept	04/Nov	04/Dec
21	✓ Conducting Preliminary Testing	04/Dec	02/Jan
22	✓ Preparing Data For IPD	01/Dec	02/Feb
23	✓ IPD Submission	03/Feb	03/Feb
	Software Requirement Final Project Report (FPR) and De...	04/Feb	01/Apr
25	✓ Minimum Viable Product Development, Testing and E...	04/Feb	06/Mar
26	✓ Formative Minimum Viable Product Submission	07/Mar	07/Mar
27	✓ Minimum Viable Product Evaluation	07/Mar	20/Mar
28	✓ Finalize FPR	21/Mar	31/Mar
29	✓ FPR and FPD Submission	01/Apr	01/Apr
	Final Project Viva (FVP):	01/Apr	09/May
31	✓ Preparation For Demo	01/Apr	20/Apr
32	✓ Viva	21/Apr	09/May

Figure 25: Gantt Chart

APPENDIX 04

3. Request Explanation

Goal: To offer a confidence score and reasoning when content is flagged as a deepfake.

Primary Actor: End User

Preconditions:

Verification must be completed.

Detection results are available.

Postconditions: The user receives an explanation of the result, including a confidence score.

Main Flow:

- User clicks “Request Explanation”.
- System retrieves relevant metadata and confidence score.
- Explanation is generated and shown to the user.

Alternative Flows:

A1: If the explanation module is unavailable, user is shown a fallback message.

4. View Detection Results

Goal: Allow users to see the outcome of deepfake verification.

Primary Actor: End User

Preconditions: Verification must be completed.

Postconditions: Detection results are visible to the user.

Main Flow:

- User navigates to results.
- System displays whether content is real or fake.

- If applicable, user is prompted to request an explanation.

Alternative Flows:

A1: If the result is not yet available, a message indicates processing is in progress.

5. Generate System Reports and Diagnostics

Goal: Generate performance metrics and evaluation reports for auditing or academic review.

Primary Actor: Maintenance Operator

Preconditions: Detection logs and system usage data are available.

Postconditions: Reports are generated and optionally exported.

Main Flow:

- The operator initiates report generation.
- The system compiles performance indicators (accuracy, recall, precision).
- Reports are saved and optionally shared.

Alternative Flows:

A1: If metrics are missing, a partial report is generated with a warning.

6. Update Model and Retrain

Goal: Incorporate new data and techniques to keep the model current and robust.

Primary Actor: Maintenance Operator

Preconditions: Updated datasets or model components are ready.

Postconditions: Model is retrained and deployed.

Main Flow:

- Operator uploads new datasets or configuration.
- Model is retrained using defined training loop.
- System replaces the old model with the new version.

Alternative Flows:

A1: If training fails, the system reverts to previous model and logs the issue.

APPENDIX 05

Key Design Attributes

1. Modularity

The system follows a modular architecture, allowing components to be independently developed, tested, and extended. This improves code reusability, maintainability, and simplifies future integration with new datasets or augmentation strategies.

2. Scalability

The architecture is designed to scale horizontally, supporting parallel processing of large datasets and diverse manipulation techniques. Reinforcement learning-driven augmentation enables the model to adapt to new deepfake types without full retraining, supporting scalable learning.

3. Efficiency

To reduce training time and computational costs, the system uses optimized CNN architectures (e.g., EfficientNet, XceptionNet) and lightweight augmentation pipelines. Reinforcement learning ensures dynamic augmentation selection, improving generalization without unnecessary resource use.

4. Usability

The user interface is implemented using Streamlit, offering a clean and intuitive front-end for both technical and non-technical users. Minimal manual input is required, and the system supports test-time adaptation, automatically adjusting to new manipulation types in real time.

5. Security and Reliability

To ensure trust and robustness, the system incorporates:

- Adversarial augmentations and corruptions during training
- Test-time fine-tuning for real-world adaptability
- Data integrity safeguards to prevent manipulation and ensure output consistency

APPENDIX 06

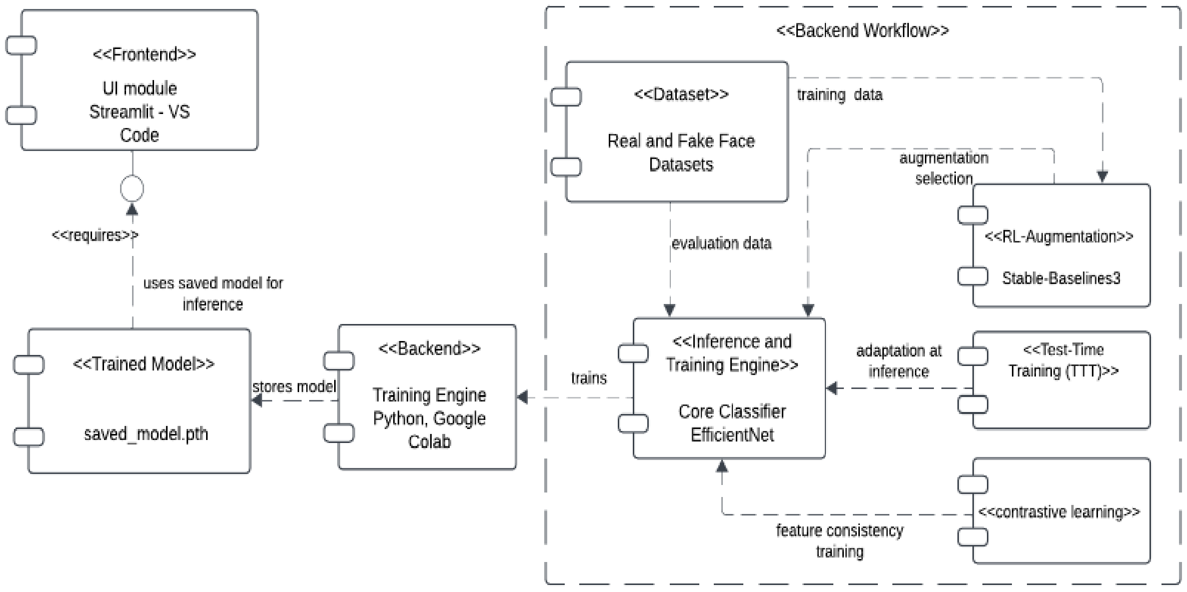


Figure 26: Component diagram (self-composed using Lucidchart)

APPENDIX 07

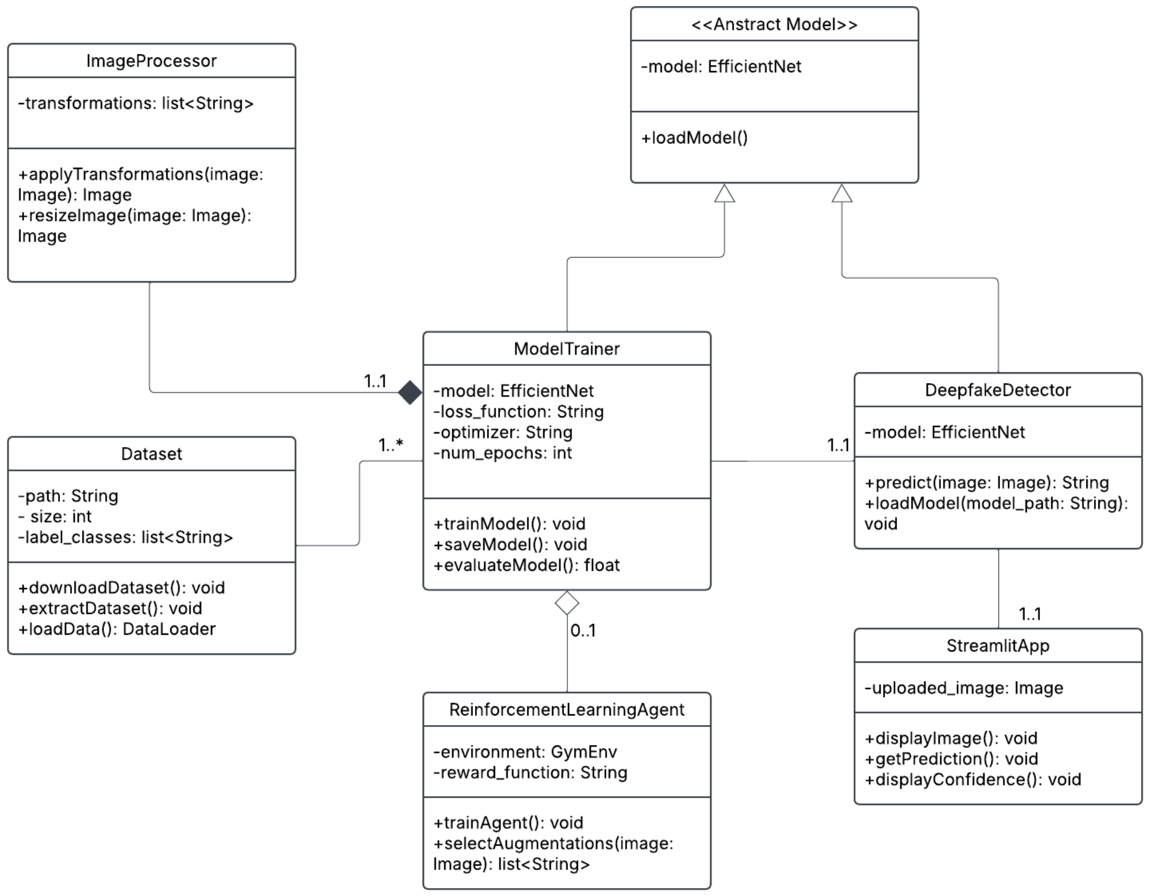


Figure 27: Class diagram (self-composed using Lucidchart)

APPENDIX 08

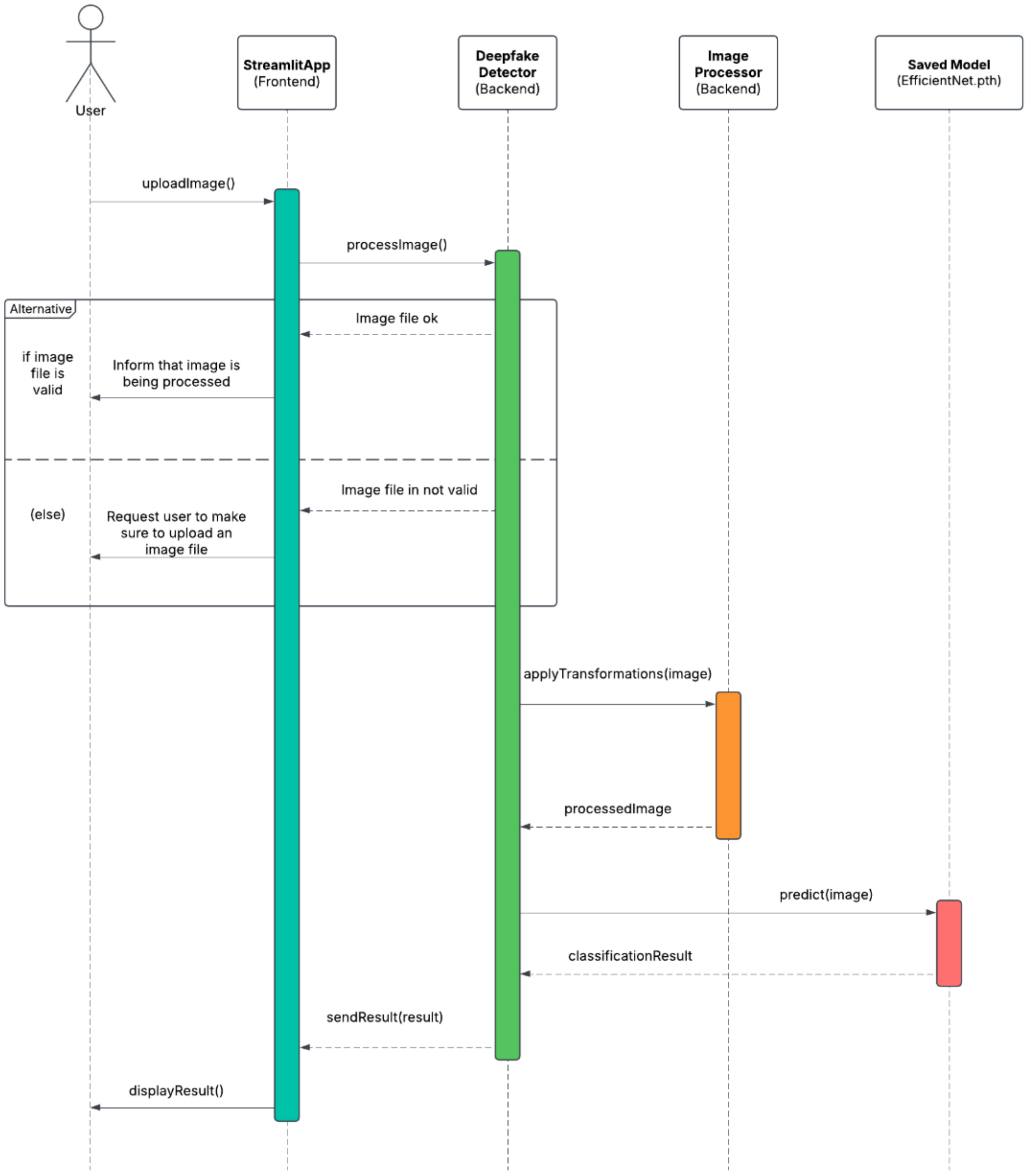


Figure 28: Sequence diagram (self-composed using Lucidchart)

APPENDIX 09

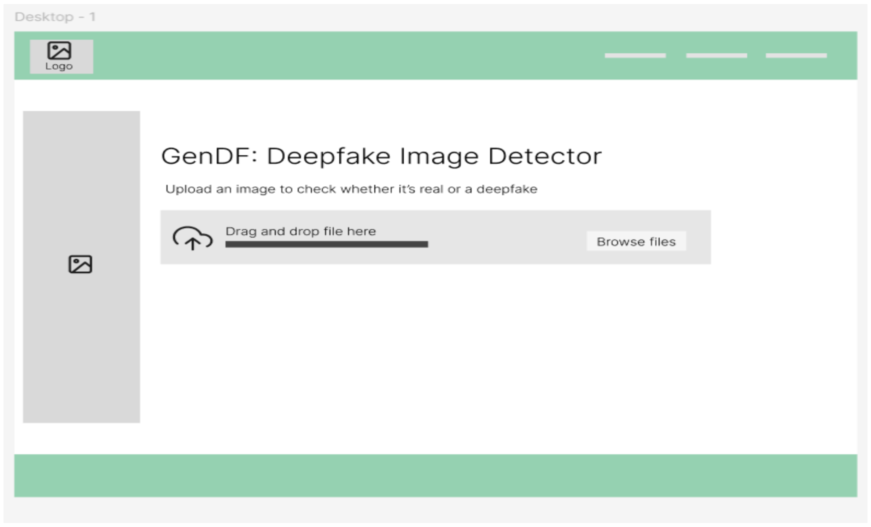

Desktop - 1
Logo
GenDF: Deepfake Image Detector
Upload an image to check whether it's real or a deepfake
Drag and drop file here
Browse files

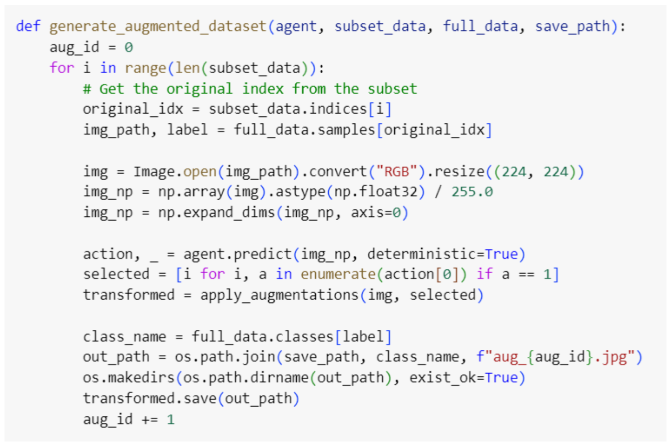
Figure 29: Low-fidelity wireframe (self-composed using Figma)

APPENDIX 10

Library	Justification
PyTorch (torch)	Core deep learning framework used to build and train the EfficientNet-based models. Enabled custom architectures for RL, CL, and TTT modules.
Torchvision	Provided access to pre-trained EfficientNet models and image transformation utilities essential for training and preprocessing.
Stable-Baselines3 (PPO)	Used for implementing the Proximal Policy Optimization agent to dynamically select data augmentations during reinforcement learning.
Gymnasium	Offered a flexible environment interface to implement the custom image augmentation RL environment required by PPO.
NumPy	Supported numerical operations across all training, evaluation, and augmentation phases.
Scikit-learn (sklearn)	Used for model evaluation via accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC computations.
Matplotlib	Enabled plotting of training/validation loss curves and Grad-CAM heatmaps during evaluation.
Pillow (PIL)	Used for image loading, resizing, and augmentation processes within the data pipeline.
Streamlit	Powered the interactive frontend application, allowing users to upload images, receive predictions, and view Grad-CAM visualizations.
Google Colab	Provided a cloud-based, GPU-enabled environment for training resource-intensive models without hardware constraints.
VS Code	Used as the primary IDE for local development, debugging, and frontend integration with the trained model.

Table 22: Libraries used

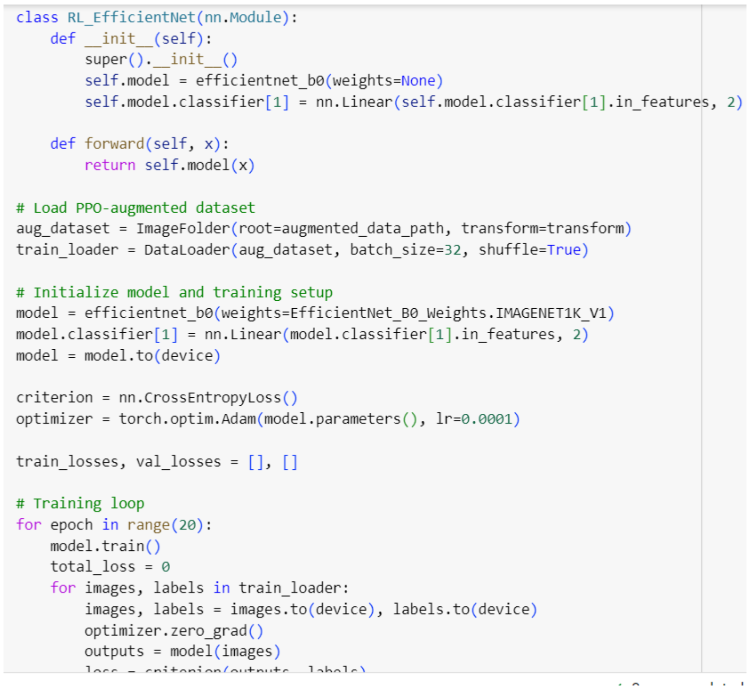
APPENDIX 11



```python
def generate_augmented_dataset(agent, subset_data, full_data, save_path):
    aug_id = 0
    for i in range(len(subset_data)):
        # Get the original index from the subset
        original_idx = subset_data.indices[i]
        img_path, label = full_data.samples[original_idx]

        img = Image.open(img_path).convert("RGB").resize((224, 224))
        img_np = np.array(img).astype(np.float32) / 255.0
        img_np = np.expand_dims(img_np, axis=0)

        action, _ = agent.predict(img_np, deterministic=True)
        selected = [i for i, a in enumerate(action[0]) if a == 1]
        transformed = apply_augmentations(img, selected)

        class_name = full_data.classes[label]
        out_path = os.path.join(save_path, class_name, f"aug_{aug_id}.jpg")
        os.makedirs(os.path.dirname(out_path), exist_ok=True)
        transformed.save(out_path)
        aug_id += 1
```

Figure 30: generate_augmented_dataset() function

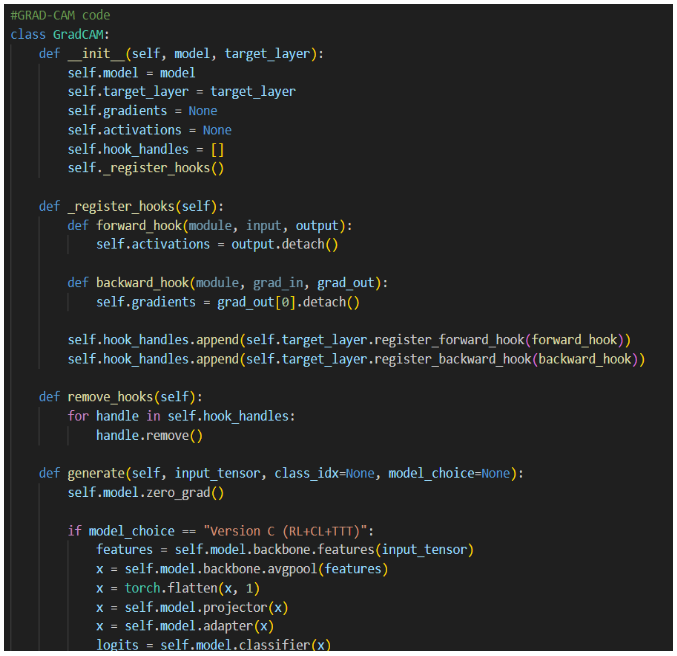
APPENDIX 12



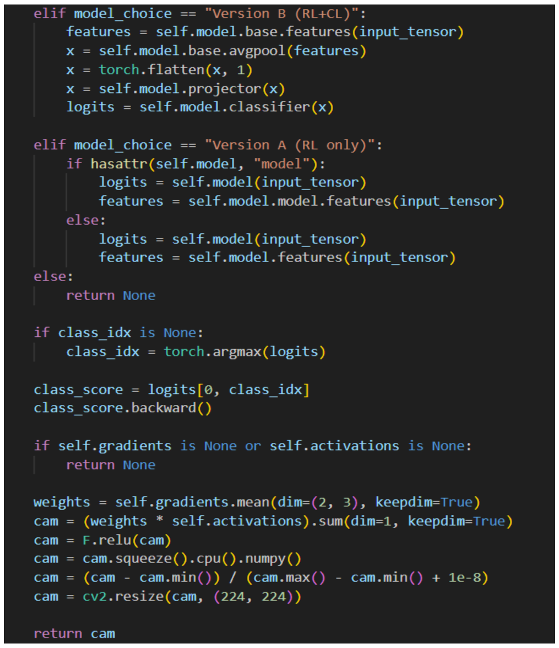
```
class RL_EfficientNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = efficientnet_b0(weights=None)
        self.model.classifier[1] = nn.Linear(self.model.classifier[1].in_features, 2)

    def forward(self, x):
        return self.model(x)

# Load PPO-augmented dataset
aug_dataset = ImageFolder(root=augmented_data_path, transform=transform)
train_loader = DataLoader(aug_dataset, batch_size=32, shuffle=True)

# Initialize model and training setup
model = efficientnet_b0(weights=EfficientNet_B0_Weights.IMAGENET1K_V1)
model.classifier[1] = nn.Linear(model.classifier[1].in_features, 2)
model = model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)

train_losses, val_losses = [], []

# Training loop
for epoch in range(20):
    model.train()
    total_loss = 0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
```

Figure 31: RL_EfficientNet class definition and training loop

APPENDIX 13



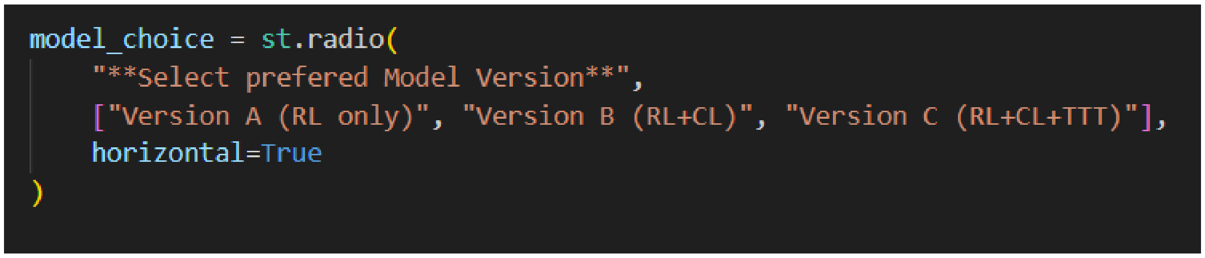
```python
#GRAD-CAM code
class GradCAM:
    def __init__(self, model, target_layer):
        self.model = model
        self.target_layer = target_layer
        self.gradients = None
        self.activations = None
        self.hook_handles = []
        self._register_hooks()

    def _register_hooks(self):
        def forward_hook(module, input, output):
            self.activations = output.detach()

        def backward_hook(module, grad_in, grad_out):
            self.gradients = grad_out[0].detach()

        self.hook_handles.append(self.target_layer.register_forward_hook(forward_hook))
        self.hook_handles.append(self.target_layer.register_backward_hook(backward_hook))

    def remove_hooks(self):
        for handle in self.hook_handles:
            handle.remove()

    def generate(self, input_tensor, class_idx=None, model_choice=None):
        self.model.zero_grad()

        if model_choice == "Version C (RL+CL+TTT)":
            features = self.model.backbone.features(input_tensor)
            x = self.model.backbone.avgpool(features)
            x = torch.flatten(x, 1)
            x = self.model.projector(x)
            x = self.model.adapter(x)
            logits = self.model.classifier(x)
```



```python
        elif model_choice == "Version B (RL+CL)":
            features = self.model.base.features(input_tensor)
            x = self.model.base.avgpool(features)
            x = torch.flatten(x, 1)
            x = self.model.projector(x)
            logits = self.model.classifier(x)

        elif model_choice == "Version A (RL only)":
            if hasattr(self.model, "model"):
                logits = self.model(input_tensor)
                features = self.model.model.features(input_tensor)
            else:
                logits = self.model(input_tensor)
                features = self.model.features(input_tensor)
        else:
            return None

        if class_idx is None:
            class_idx = torch.argmax(logits)

        class_score = logits[0, class_idx]
        class_score.backward()

        if self.gradients is None or self.activations is None:
            return None

        weights = self.gradients.mean(dim=(2, 3), keepdim=True)
        cam = (weights * self.activations).sum(dim=1, keepdim=True)
        cam = F.relu(cam)
        cam = cam.squeeze().cpu().numpy()
        cam = (cam - cam.min()) / (cam.max() - cam.min() + 1e-8)
        cam = cv2.resize(cam, (224, 224))

        return cam
```

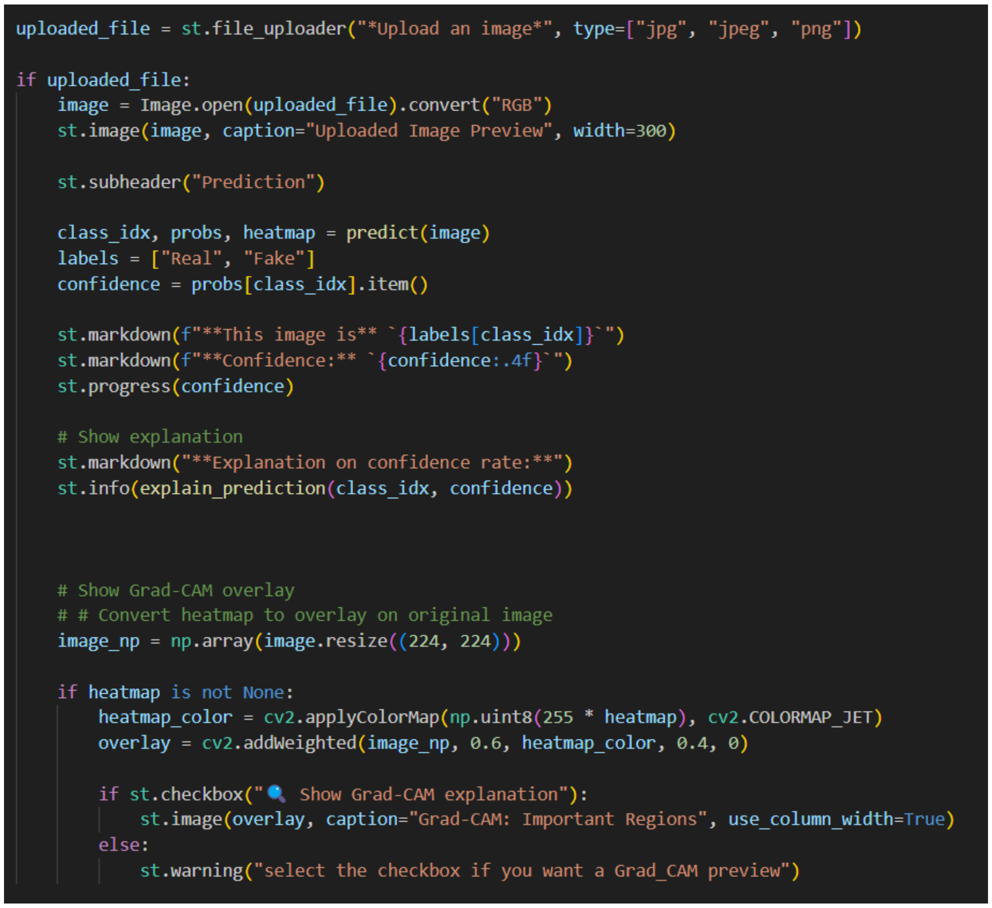
Figure 32 Grad-CAM visualizations for Versions A and B

APPENDIX 14



```python
model_choice = st.radio(
    "**Select prefered Model Version**",
    ["Version A (RL only)", "Version B (RL+CL)", "Version C (RL+CL+TTT)"],
    horizontal=True
)
```

Figure 33 Model selection radio buttons



```python
uploaded_file = st.file_uploader("*Upload an image*", type=["jpg", "jpeg", "png"])

if uploaded_file:
    image = Image.open(uploaded_file).convert("RGB")
    st.image(image, caption="Uploaded Image Preview", width=300)

    st.subheader("Prediction")

    class_idx, probs, heatmap = predict(image)
    labels = ["Real", "Fake"]
    confidence = probs[class_idx].item()

    st.markdown(f"**This image is** `{labels[class_idx]}`")
    st.markdown(f"**Confidence:** `{confidence:.4f}`")
    st.progress(confidence)

    # Show explanation
    st.markdown("**Explanation on confidence rate:**")
    st.info(explain_prediction(class_idx, confidence))



    # Show Grad-CAM overlay
    # # Convert heatmap to overlay on original image
    image_np = np.array(image.resize((224, 224)))

    if heatmap is not None:
        heatmap_color = cv2.applyColorMap(np.uint8(255 * heatmap), cv2.COLORMAP_JET)
        overlay = cv2.addWeighted(image_np, 0.6, heatmap_color, 0.4, 0)

        if st.checkbox("🔍 Show Grad-CAM explanation"):
            st.image(overlay, caption="Grad-CAM: Important Regions", use_column_width=True)
        else:
            st.warning("select the checkbox if you want a Grad_CAM preview")
```

Figure 34 Image upload code logic

APPENDIX 15

Test scenario	Expected output	Model version	Real image/Fake image	Status
Uploading real and fake images	Correct prediction label ('Real' or 'Fake') with confidence shown for all versions.	A	Real 01	Pass
			Real 02	Fail
			Real 03	Fail
			Fake 01	Pass
			Fake 02	Fail
			Fake 03	Fail
		B	Real 01	Pass
			Real 02	Fail
			Real 03	Fail
			Fake 01	Pass
			Fake 02	Pass
			Fake 03	Fail
		C	Real 01	Fail
			Real 02	Fail
			Real 03	Fail
			Fake 01	Pass
			Fake 02	Pass
			Fake 03	Pass
Grad-CAM explainability	Grad-CAM heatmap is generated and overlays correctly on the image.	A	Real	Pass
			Fake	Pass
		B	Real	Pass (better)
			Fake	Pass

				(better)
		C	Real	N/A
			Fake	N/A
Model switching	Models switch smoothly and give consistent results without crash.	A B C	All versions switch correctly and maintain state integrity.	Pass
Invalid inputs	Friendly error or warning shown; app should not crash.	All	Non-image files (e.g., .txt, .pdf) cannot be selected due to file type restrictions in the Streamlit uploader.	Pass
			The corrupted image gives an error but the app doesn't completely crash.	Fail
Stress test(Larger images)	The system slows down but does not crash. Streamlit recovers smoothly.	All	The app slows momentarily but recovers; no memory crashes observed.	Pass

Figure 35: System testing table

APPENDIX 16

To validate the interpretability of GenDF predictions, Grad-CAM (Gradient-weighted Class Activation Mapping) was implemented across all model versions. This visual explainability technique highlights the facial regions that most influenced the model's decision, offering insights into model behavior and transparency.

Version A (RL + EfficientNet) and **Version B (RL + CL + EfficientNet)** were both evaluated with Grad-CAM overlays, using a test image from the dataset. The system successfully generated heatmaps with clear attention on relevant facial features (e.g., eyes, nose, mouth). This demonstrated that both models were learning meaningful spatial cues when making predictions.

For **Version C (RL + CL + TTT + EfficientNet)**, Grad-CAM was not implemented manually via backend notebooks due to architectural complexity involving the adapter layer.

These explainability visuals increase trust in the system by revealing how the model processes visual cues during classification. Furthermore, they serve as a useful tool for debugging misclassifications or confirming model behavior under edge cases.

APPENDIX 17

- To verify the performance of all three GenDF versions using standard evaluation metrics.
- To ensure that the implemented functionalities meet the ‘Must Have’ and ‘Should Have’ requirements defined in the SRS using the MoSCoW prioritization.
- To confirm that essential non-functional requirements such as accuracy, scalability, and security are satisfied.
- To identify any bugs, design flaws, or performance bottlenecks during model inference or component interaction.
- To ensure proper integration between modules including augmentation, model inference, and the user-facing frontend.
- To test the robustness of the system through edge case inputs and unexpected usage scenarios.
- To provide baseline results that future research can build on or compare against.

APPENDIX 18

The table below outlines the functional requirements defined during the specification phase and their current implementation status.

FR ID	Requirement Description	Priority	Status
FR1	The user must be able to upload an image to the system.	M	Implemented
FR2	The system must validate image type and reject unsupported or corrupted files.	M	Partially Implemented (Fails on corrupted images)
FR3	Uploaded images must be preprocessed before model inference.	M	Implemented
FR4	The system must use a trained model to predict real vs fake labels.	M	Implemented
FR5	Grad-CAM visual explanations must be available for supported model versions.	S	Implemented (A & B only)
FR6	The user must be able to switch between different model versions.	M	Implemented
FR7	Prediction results must be displayed with confidence scores.	M	Implemented
FR8	The system should display an error message for non-images.	M	Implemented
FR9	A preview of the uploaded image should be shown before classification.	S	Implemented
FR10	Users should be able to optionally view Grad-CAM visualizations.	C	Implemented (A & B only)
FR11	The system should not support video-based deepfakes at this stage.	W	Not Considered
FR12	The system could support saving or exporting prediction results.	C	Not Implemented

Table 23: Evaluation of functional requirements

Completion Rate: 10 out of 12 = **83.3%**

APPENDIX 19

NFR ID	Requirement Description	Priority	Status
NFR1	The system must provide accurate predictions to ensure trust in results.	M	Implemented
NFR2	The interface must be user-friendly and intuitive for non-technical users.	M	Implemented
NFR3	Inference time should be low to support efficient prediction and responsiveness.	M	Implemented
NFR4	The system must handle corrupted or invalid inputs gracefully without crashing.	S	Partially Implemented
NFR5	The model should generalize well across unseen inputs and slight domain shifts.	S	Implemented
NFR6	No user data should be stored; the system must function entirely in-memory for security.	M	Implemented
NFR7	The application must maintain performance across model versions without memory crashes.	S	Implemented
NFR8	The UI should adapt well on standard desktop environments (tested via Streamlit).	C	Implemented
NFR9	The system is not designed for mobile compatibility at this stage.	W	Not Considered

Table 24: Evaluation of non-functional requirements

Completion Rate: 8 out of 9 = **88.9%**

APPENDIX 20

Learning Outcome	**Achievement**
LO1	Achieved through the integration of PPO-based reinforcement learning, contrastive learning, and test-time training techniques within a unified framework. The chosen tools (PyTorch, Streamlit, PPO, Grad-CAM) were clearly justified in the Methodology and Implementation chapters.
LO2	A detailed plan was devised and updated regularly. The project was divided into three implementable model versions (A, B, and C) to maintain scope control and time efficiency. Gantt charts and milestone reviews ensured steady progress.
LO3	Functional and non-functional requirements were gathered using literature analysis and informal expert insights. Techniques such as MoSCoW prioritization and use case diagrams were used during the requirements elicitation stage.
LO4	The project required extensive literature review on deepfake generation, detection, RL, CL, and TTT. Each chosen technique was benchmarked against alternatives and its relevance justified within the commercial context of image-based verification systems.
LO5	New skills such as PPO implementation, contrastive loss design, Grad-CAM explainability, and test-time training were self-learned through online research and experimentation, demonstrating high autonomy and technical curiosity.
LO6	Ethical and professional considerations such as misuse of deepfake technology, data privacy, fairness, and security were addressed in the SLEP chapter. In-memory processing and responsible dataset use reinforced these considerations.
LO7	The project produced a sophisticated, three-version deepfake detection system—GenDF—implemented in PyTorch, tested thoroughly, and deployed with a usable Streamlit frontend integrating Grad-CAM explainability.
LO8	The final report critically documents all aspects of the project—research design, implementation decisions, evaluation findings, and reflections—while also highlighting gained skills and lessons learned.

LO9	The author has prepared for the viva by ensuring that each model version is functionally integrated, explainable, and testable. A well-structured frontend and evaluation script were developed to support the demonstration and discussion of technical decisions during the viva.

Table 25: Achievements of Learning Outcomes

APPENDIX 21

Problem	Mitigation
Steep RL/CL learning curve	Tackled unfamiliar concepts like PPO and contrastive loss via self-study, academic papers, and iteration.
TTT integration issues	Fixed adapter dimension mismatch by adjusting input size from 512 to 1280 to match EfficientNet outputs.
Version control complexity	Modularized codebase and isolated reusable components to manage Versions A, B, and C effectively.
Limited compute resources	Upgraded to Colab Pro, enabled GPU usage, and focused tuning on finalized models.
Grad-CAM inconsistencies	Adapted Grad-CAM hooks dynamically to suit each model's architecture and output format.
Lack of expert feedback	Substituted expert interviews with insights from literature and informal peer feedback.
Testing and reporting overload	Automated evaluation across all versions with reusable test scripts, aligning outputs with reporting.

Table 26: Problems and challenges faced

APPENDIX 22

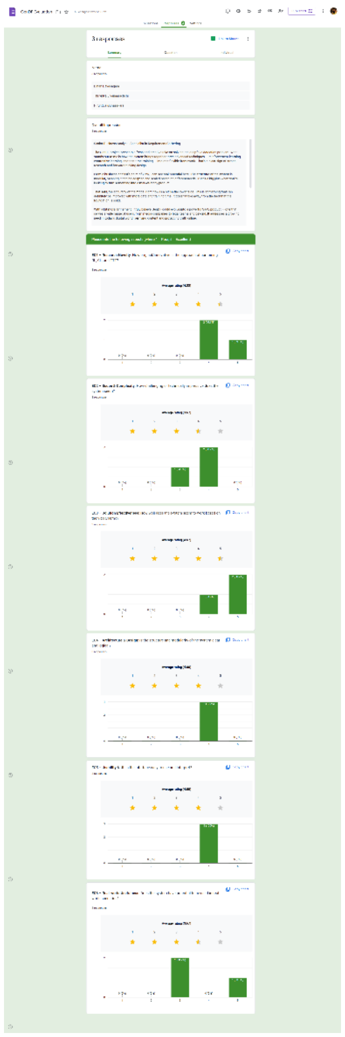


Figure 36: Evaluation evidence

A clear version of this screenshot can be found using this link.