UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

SCHOOL OF ELECTRICAL AND INFORMATION ENGINEERING

Academic Misconduct Declaration

I ___Keon_____Roohani_____ (student name and surname),
_____1869754_____ (student number), hereby declare that:

1. I have read the Student Academic Misconduct Policy [1] and I understand that "*Academic Misconduct includes any action which gains, attempts to gain, or assists others in gaining or attempting to gain an unfair academic benefit. It includes Plagiarism, collusion, cheating, copying, contract cheating, fabrication of data, the use and/or possession of unauthorised materials or devices during an assessment; and falsification or misrepresentation of information including, falsification of a medical certificate, and/or changing a script after it has been marked*" [1].
2. I know that any act of academic misconduct is unacceptable, and carries a penalty as prescribed in the Student Academic Misconduct Policy [1].
3. This assessment/submission for the course_____ELEN4000A/4011A_____is my own work and I have not engaged in any act of academic misconduct in the completion of this work.

*4. In this group submission for the course, _____ I have collaborated with the following students:

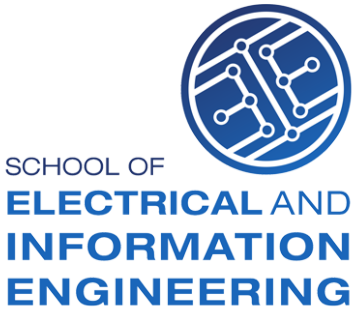
	Surname & Name	Student Number	% Contribution
1			
2			
3			
4			
5			
6			

Signature:  Date: ___09_September_2022_____

Reference:

[1] University of Witwatersrand, Student Academic Misconduct Policy G-C-2022-029, 2022.

*To be filled for group submissions only.



Faculty of Engineering and the Built Environment
University of the Witwatersrand, Johannesburg, South Africa

CLARITY: A CLOUD BASED DEEPFAKE DETECTION PLATFORM

ELEN4000/4011: Information Engineering Design II

Topic 9: "Deepfake hunter: An AI-enabled software platform for detecting deepfakes on the Internet"

Keon M. L. Roohani
1869754

Supervisor: Turgay Celik

9 September 2022

Abstract: The rise of commonplace synthetic media is an emerging threat in the modern world. Rapidly advancing deep learning research is making it easier than ever before to generate hyper-realistic fake images and videos that are undetectable by the human eye. This paper describes the design of Clarity, an automated cloud based platform which uses machine learning to identify deepfakes on the internet. Clarity is designed with sustainability and security at its core to gain user trust and provide long-term support for deepfake detection. Its loosely coupled microservice back-end architecture makes the design scalable and flexible, while still maintaining a high overall performance with low latency. The system utilises industry standard MLOps principles to automate the machine learning pipeline and ensure reproducible, explainable, transparent, and secure detection models that accurately identify the full spectrum of deepfakes in the wild. Clarity's use of the Google Cloud Platform ensures cost reductions due to its support of open-source software and its availability of efficient hardware for model training. Furthermore, it guarantees reduced carbon-footprint over manually deployed hardware by outsourcing computation, storage, and networking to low carbon emission grids. Monetization schemes for the system have been devised to ensure longevity and financial feasibility.

CONTENTS

List of Figures

List of Tables

1. INTRODUCTION

The rise of commonplace synthetic media is an emerging threat in the modern world [8]. Rapidly advancing deep learning research is making it easier than ever before to generate hyper-realistic fake images and videos that are undetectable to the human eye. Fake media of this kind, often referred to as deepfakes, have the potential to undermine social trust, sabotage political discourse, and spread misinformation at unprecedented levels through social media platforms and the internet [8]. Only by leveraging the same deep learning techniques which have placed the world in this quagmire, can we hope to dispel the negative effects which deepfakes are sure to bring in the future. The need has arisen for web services, that are transparent and explainable, which are capable of detecting deepfakes for the ordinary citizen. This paper describes the design of Clarity, an automated cloud based platform which uses machine learning to identify deepfakes on the internet. Clarity is designed with sustainability and security at its core in order to gain user trust and provide long-term support for the *war* on deepfakes. Clarity also aims to be environmentally friendly, with a low carbon footprint and minimal power consumption. Further details of the methodology and operations used to achieve these features is discussed in the remainder of the report. Section 2. provides a synopsis of modern deepfake generation and detection techniques, catalogues emerging deepfake detection platforms, and highlights the need for industry standard systems engineering practices in the production of the Clarity service. Section 3. describes the scope, features, role-players, and specifications of the Clarity system at a high-level. Section 4. provides a detailed breakdown of the overall Clarity system architecture, the deepfake scanner architecture and relevant operational decisions, pertinent hardware and software concerns, potential economic feasibility, and negative environmental impact mitigation techniques. Section 5. briefly describes the expected Clarity deployment life-cycle, from design to real-world potentially profitable cloud-based service. Additionally, attached is appendix A, which presents a newspaper style article that describes the dangers of deepfakes, and the necessity of Clarity, to the layman.

2. BACKGROUND

2.1 Deepfakes in the Modern World:

The term *Deepfake* stems from the concepts of "deep learning" and "fake" media [9]. Deepfakes are synthetically generated textual, image, video, or audio data that is often used to depict people saying things, or performing actions, that have never happened [10]. Naturally, this has resulted in the use of this technology for nefarious purposes such as spreading misinformation on social media platforms, the creation of pornographic material depicting well-known individuals, online bullying, and the misrepresentation of politicians during public discourse [11]. The rapid proliferation of information over social media platforms makes the spread of misleading and socially harmful deepfakes an alarming threat. This is bolstered by the constant improvements in deepfake generation techniques which have resulted in synthetic media that is almost indistinguishable from real media to the ordinary citizen [12]. Section 2.1.1 provides a brief review of the current state of deepfake generation techniques. In order to combat the negative effects that deepfakes could have on individuals and society at large, a number of deepfake detection techniques have been proposed; these are discussed in section 2.1.2. Deepfake detection, in its current form, is often viewed as a 'cat-and-mouse' game [13]; Deepfake generation techniques are improved, which results in the improvement of detection techniques to identify these deepfakes, which results in further generation improvements to 'beat' the newer detection systems, and the cycle continues. In order to categorise newly discovered deepfake generation techniques, academics and industry have compiled deepfake detection datasets to assist in the detection process; these are discussed in section 2.1.3. This emergent *war* on deepfakes has led to the creation of deepfake detection services. These are platforms which continually improve the ability of academics and ordinary citizens to identify deepfakes "in the wild". Existing deepfake detection services are mentioned in more detail in section 2.1.4. The purpose of this paper is the design of one such service which can assist in the identification of deepfakes for ordinary citizens in the coming years.

2.1.1 Deepfake Generation Techniques: Deepfake generation is a continually evolving process that combines the latest in machine learning (ML) methods to develop ever more realistic synthetic media. Generation is most often performed using Deep Learning (DL) techniques due to their ability to

represent high dimensional data, such as those of image or video media [14]. The four major types of deepfake media, that being textual, image, video, and audio data, each have their respective generation techniques. The focus of this paper are the image and video (both with and without audio) deepfake generation methods. It must be noted that stand-alone audio is beyond the scope of this synopsis. With regards to image and video deepfake generation, there are five types of manipulation techniques which are discussed in the literature [12, 14, 15]. These are i) *face/identity swap*, ii) *lip syncing*, iii) *face reenactment* or *puppet mastery*, iv) *entire face synthesis* and finally v) *facial attribute manipulation* [12]. *Face/identity swap* involves swapping the face of a person in the source media with that of a given target media. Traditional approaches utilised manual swapping which often had obvious defects and identifiable occlusions. However, the introduction of deep learning models, such as those used by FakeApp [16] and FaceSwap [17], which utilise encoder-decoder pairs to transfer features from source to target, which involves reconstructing a new face by swapping encoders and decoders, have resulted in higher quality deepfakes than manual methods. *Lip syncing* methods involve ensuring that a given audio matches the visual characteristics, such as lip movement, for a video [12]. An ubiquitous example of this is seen in recreations of US President Barak Obama making false public announcements [18]. These methods often make use of a Recurrent Neural Network (RNN) to map audio to image for each frame [18]. However, the use of Generative Adversarial Network (GAN) methods are also prevalent and take into account the temporal characteristics of audiovisual media while performing lip syncing [19]. *Face reenactment* or *puppet mastery* involves the transfer of facial expressions from one person to another, rather than the entire face as seen in face swapping [12]. Manual methods, such as those used in Face2Face [20], and deep methods, often utilising GANs [21], have shown success in this type of deepfake generation. *Entire face synthesis* involves the generation of images of a human face which do not necessarily exist [15]. This is often achieved using GANs or Variational Auto-Encoders (VAE). Finally, *facial attribute manipulation* describes deepfakes in which a subset of attributes regarding a specific face are altered and the remainder of the face is left unchanged [12]. Each type of deepfake generation method has a plethora of nuanced techniques, each with its own benefits and flaws, that make it impossible to cover the full landscape. Deepfake generation techniques are continually being improved to ensure greater generalisation, better temporal coherence, reduced occlusions, and mitigation of identity leakage [12]. Furthermore, the ease of access, and simplicity of many deepfake generators, permits the layman to generate hyper-realistic deepfakes at a relatively low cost using existing publicly available methods [22]. An understanding of the existing deepfake generation methods is necessary to ensure detection can be performed holistically "in the wild".

2.1.2 Deepfake Detection Techniques: The ability to efficiently and accurately detect deepfakes is vital to inhibit the spread of insidious and defamatory media throughout the world. Traditionally, aspects of deepfake detection have been derived from media forensics [15]. These include in-camera forensics, which identify intrinsic camera fingerprints which are transformed during deepfake generation [23–26]. Additionally, out-camera fingerprints, such as those caused by editing software and manual image splicing, have also been utilised to detect synthetic media [27–30]. While these traditional techniques have shown success in specific cases, they lack the generalisation required to perform effectively against the modern deepfakes which often undergo compression and resizing on social media platforms [31]. The use of DL techniques has shown great promise in deepfake detection [11, 12, 32], with the two major approaches being i) *artifact-specific* and ii) *undirected analysis*. Both approaches make use of standard formations of known DL models such as Convolutional Neural Networks (CNN) [33], RNNs especially with Long Short-Term Memory (LSTM) [34], VAEs [35], and GANs [36]. These approaches train detectors using large datasets of labelled real and fake data. *Artifact-specific* detection observes either spatial or temporal features of a given input [32]. Spacial features include blending artifacts, environmental anomalies, and forensic fingerprints. While temporal features include behavioural and physiological anomalies, synchronization inconsistencies, and an overall temporal incoherence often seen through jitter. *Undirected analysis*, on the other hand, permits the detection algorithm to determine independently the features relevant to deepfake detection [32]. In this regard, deepfake detection can be defined as either a classification problem, in which a given image is in a Boolean sense "fake" or not, or an anomaly problem, in which potentially fake outliers are given an anomaly score representing their deviation from the expected data [32]. For the purpose of this design, an ensemble of DL undirected analysis deepfake detection models will be utilised. This choice is due

to their improved performance on compressed data and overall tendency to be more generalisable than other methods [15]. The use of an ensemble of models has been shown to produce more accurate results with fewer false positives, leading, overall, to a more robust system [37].

Some important factors of any DL model are the chosen dataset, the model architecture, and the model training method. Section 2.1.3 provides further detail on existing datasets which can be used in the deepfake detection training process. The choice of model architecture has implications on the accuracy rate, loss rate, training time, and inference time of a given deepfake detection model [38]. There exists a number of CNN model architectures which have shown success in deepfake detection, these include: MesoNet [39], Residual Network 50 (ResNet-50) [40], Visual Geometry Group Network 19 (VGG-19) [41], XceptionNet [42], and EfficientNet [43] to name a few. There are also also transformer networks, such as the DETR network [44], which are used in state-of-the-art detection. These transformer networks utilise an encoder-decoder architecture as the model baseline. Experiments, such as those seen in [38, 43], show that EfficientNet performs at a higher accuracy than all models on images and videos, except XceptionNet on videos, while also having lower training and inference times. XceptionNet does have longer training and inference times due to its large network size, however, it boasts the highest accuracy of all models, especially on video input data. These trade-offs in accuracy, inference time, and training time are critical points which are evaluated in this design. Finally, the training process used in deepfake detection plays a vital role in the generalisability and accuracy of the model. Many models and experiments [31, 45–48] show that while models exhibit a high accuracy on the dataset used in training, they perform abysmally on other datasets, especially those derived from deepfakes "in the wild". In order to address this weakness, training techniques have been devised to train models on multiple datasets [45, 46]. These methods must minimize catastrophic forgetting, which is a phenomenon that occurs when already trained networks forget previous tasks when trained on new datasets [49]. Both [45] and [46] utilise knowledge distillation (KD) and continual learning techniques to maintain high accuracy while training on multiple deepfake detection datasets. Use of these techniques is employed to lower the required ensemble size, thus increasing the efficiency, and reducing training times, inference times, and power consumption of the system as a whole.

2.1.3 Deepfake Detection Datasets: Training deepfake detection models requires a large amount of labelled images or videos. The scientific community, and industry, have set out to create datasets that fill this requirement. Table 1 shows many of the most notable public datasets available. These datasets contain a wide variety of deepfakes generated using both known and unknown methods. Furthermore, some datasets, such as WildDeepFake [48] and Facebook's Deep Fake Detection Challenge (DFDC) [50] contain a wide variety of actors to reduce statistical bias, such as misrepresentation of gender and race, and provide a high data variance. A robust deepfake detection service should perform at the state-of-the-art on all listed datasets, as well as perform at a high-level on randomly sampled "wild" deepfakes. Additionally, it should be scalable enough to utilise new datasets as they become available.

2.1.4 Deepfake Detection Services: The trend towards widespread deepfake generation has prompted industry and academics to create internet based deepfake detection services. These are platforms that are designed to detect deepfakes over the internet, and serve the purpose of easing the mind of ordinary citizens, while being transparent and explainable.

The MeVer deepfake detection platform [47] is a web based platform that combines an ensemble of five EfficientNet and DETR models of different kinds. The platform places an emphasis on transparency and explainability, with useful *model cards* created to accompany each trained model and describe its features. The platform uses a RESTful API service to communicate with its microservice back-end architecture. It has weaknesses against adversarial attacks, such as Distributed Denial of Service (DDoS) attacks, which make the service unreliable in a real world setting. The service is evaluated using the FaceForensics++ [31], CelebDF-V2 [54], and WildDeepFake [48] datasets, with a non-trivial, but not excellent, balanced accuracy of $\approx 80\%$. A weakness of the system is its lack of temporal detection techniques which would provide a more holistic detection scheme.

The Deepware Deepfake Scanner [64] is an online deepfake detection platform that allows images, videos, and YouTube URLs to be uploaded and scanned. They utilise an EfficientNet-B7 model trained on ImageNet [65] and then adapted to the DFDC dataset [50] using a transfer learning approach.

Table 1: A table showing deepfake detection datasets created by the scientific community and industry for training with images and videos. Datasets are ordered from oldest to newest in terms of release date. The table has been adapted from [1].

Deepfake Detection Datasets					
Dataset Name	Total Images/ Videos	Real Images	Fake Images	Real Videos	Fake Videos
UADFV [51]	98	-	-	49	49
Deepfake-TIMIT [52]	620	-	-	-	620
DFDC-Preview [53]	5,214	-	-	1,131	4,113
FaceForensics++ [31]	5,000	-	-	1,000	4,000
Celeb-DF [54]	6,229	-	-	590	5,639
FakeCatcher [55]	142	-	-	-	142
DFFD [56]	303,039	58,703	240,336	1,000	3,000
Google DFD [57]	3,431	-	-	363	3,068
DFDC [50]	128,154	-	-	23,954	105,500
DeeperForensics [58]	60,000	-	-	50,000	10,000
Vox-Deepfake [59]	2,171,215	-	-	1,125,429	1,045,786
WildDeepFake [48]	7,314	-	-	3,805	3,509
ForgeryNet [60]	3,117,309	1,438,201	1,457,861	99,630	121,617
DF-W [61]	1,869	-	-	-	1,869
FFIW [62]	20,000	-	-	10,000	10,000
OpenForensics [63]	115,325	45,473	70,325	-	-

This service also suffers from a lack of temporal detection capacity and uses only a frame-by-frame analysis. The Deepware platform allows users access to the scanner API itself, which eases its use when comparing it to other services, and using it as a component of larger systems.

DeepMedia [66] is a platform that both generates and detects deepfakes. Its generation methods focus on deepfake translation, this is used when dubbing videos and performing real-time translations. DeepMedia's deepfake detection system is not publicly available, this allows the company to keep its models and pipelines a secret from potential attackers, however, it is known that they are utilising vision-transformer based models for their efficiency and accuracy [67]. DeepMedia is aiming to ensure a 99% accuracy on all existing datasets before their product is monetised [67].

There are other services such as DuckDuckGoose [68] and DeepFake-o-Meter [69] each with their own models, applications, monetisation schemes, and more. The Clarity design aims to build on the strengths of other existing services while maintaining scalability, transparency, explainability, security, ease-of-use, and monetary feasibility.

2.2 Software Systems Engineering:

The design and implementation of real world, distributed software systems is a challenging endeavour. Ostensibly, it may appear that these systems can be designed and implemented rapidly, especially in the case of ML systems which can be trained and deployed within a matter of days; this is not the case. Software systems often have a low cost to entry, however, improperly designed and maintained systems suffer a long-term *technical debt* [70]. This debt can be the cause of code refactoring, system downtime, scalability issues, unwanted dependencies, unsolvable bugs, and testing difficulties later in the system life-cycle [70]. These problems can be combated by using DevOps methodologies [71]. DevOps combines development and operations principles in the implementation of software systems, promoting incremental improvements with constant feedback and monitoring [71]. While DevOps is primarily concerned with system implementation, acknowledgement of its principles in the design phase is crucial to ensure success of the final system. With regards to software systems that utilise machine learning, an outgrowth of DevOps, known as Machine Learning Operations (MLOps) [72–76], has been devised in recent years to support the development and maintenance of such systems. MLOps places

an emphasis on data management, model learning, model verification, model deployment, ethics, law, end-user trust, and security in both the design and deployment of ML systems [76]. MLOps ideologies are supported by a plethora of open-source and enterprise tools which holistically facilitate the design of ML systems [77, 78]. The Clarity deepfake detection system respects DevOps principles in its overall design, while the scanner module itself is designed with MLOps principles at its core.

3. CLARITY: SCOPE AND FEATURES

3.1 System Input Requirements:

The Clarity Deepfake Detection Service is a cloud based system that has been designed to identify synthetic images and videos, both with and without audio. The inputs of the system include client local images and videos, as well as URLs linking to Facebook images and videos, YouTube videos, and Twitter posts. The choice to exclude textual and solely audio deepfakes is due to the comparatively smaller negative societal impact these forms of media demonstrate. Videos inputs cannot be longer than ten minutes; this is to reduce load on the system. Clarity is designed with scalability as a salient principle, meaning that it can be improved flexibly should textual, audio, or other newly emergent deepfakes require detection.

3.2 System Output Specifications:

The outputs of the Clarity system, much like the system's name itself, were chosen with transparency, explainability, and accountability in mind. There are four outputs, namely: A confidence index, a falsity label, a description of the model used, and finally, optional reports by expert analysts. The confidence index is a value between zero and one that indicates the confidence a specific model has that the given input is a deepfake, with zero representing definitely not a deepfake and one representing absolutely a deepfake. The falsity label has three possible values of "Fake", "Real", and "Suspicious". Due to the largely biased nature of the deepfake classification problem, in which the number of real images or videos greatly outweighs the number of deepfakes in the wild, the falsity labels represent the following confidence indices. A "Real" label represents a confidence index of less than 0.5 for all models. A "Suspicious" label represents a confidence index of greater than 0.5 and less than 0.9 for one or more models. Finally, a "Fake" label represents a confidence index of greater than 0.9 for one or more models. This was chosen to minimize false positive identifications of real media being labeled fake, while also providing a buffer in cases where the system is suspicious but cannot conclusively detect a given deepfake. Reduction of false positives is important as mislabeling a real input as fake could have serious implications in political and social discourses. The description of the model used is a useful system output as it provides the user insight into which deepfake detection technique has been used to identify a given deepfake, without providing details of the model hyper-parameters which could lead to adversarial attacks. Furthermore, the confidence index of each model used is shown; this provides further transparency for the final detection decision. Finally, expert reports are textual documents written to provide insight into a Clarity deepfake label decision. These aim to be objective reports with details such as the reason for a decision; spectral, temporal, and spatial inconsistency explanations; the expected generation techniques used; and any other relevant information.

3.3 Actors and Respective Interactions:

There are three expected types of actors who will utilise the Clarity system: Free Users, Customers (Paid Users), and Experts. The actors and their corresponding system interactions are shown in figure 1. These interactions describe the functional requirements of the system that are met as seen in the detailed design in section 4. Free users represent potential customers of the system; these individuals have access to a limited feature set of the overall system and can only make use of a small subset of the available detection models. There is no login requirement for these users, however, it must be noted that their requests are not prioritised by the Clarity scanner so as to prevent potential denial of service for paying customers. Customers are users that have paid for the Clarity system. These users have access to the full functionality of the system, including: All scanner models and corresponding reports, access to previous activity logs, human expert analyst reporting, the ability to dispute scanner results, and login credentials. Customers can purchase varying levels of system support on a per month payment model. A potential service model breakdown is shown in figure 2,

this follows the User Interface (UI) design scheme described in section 4.6. Finally, expert users are individuals that are trusted and vetted by the Clarity system developers. These users must be experts in the field of deepfake generation and detection, or related fields, with accompanying credentials. They are vital to ensure explainability, transparency, and longevity of the Clarity system. Their purpose is to provide detailed reports on media that is suspected of being a deepfake but has not been identified by the Clarity scanner, and to clarify media falsely identified as a deepfake. Furthermore, they provide legal and societal accountability, transparency, and certitude in cases where the correct identification of a deepfake is critical, such as in legal proceedings or political queries.

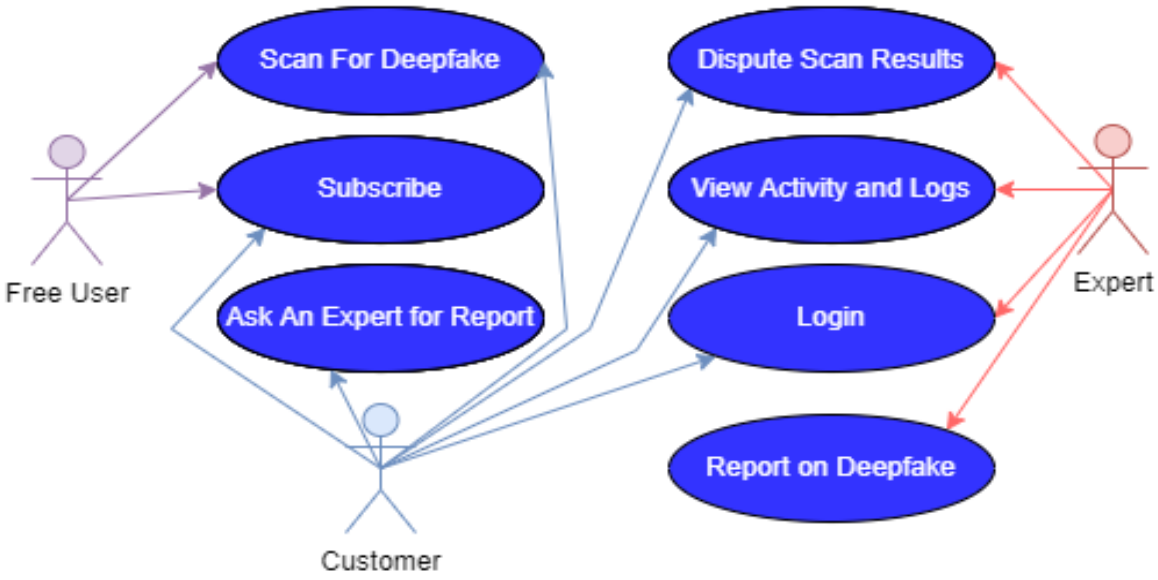


Figure 1: Use case diagram showing the expected Clarity deepfake detection system actors and corresponding system interactions.

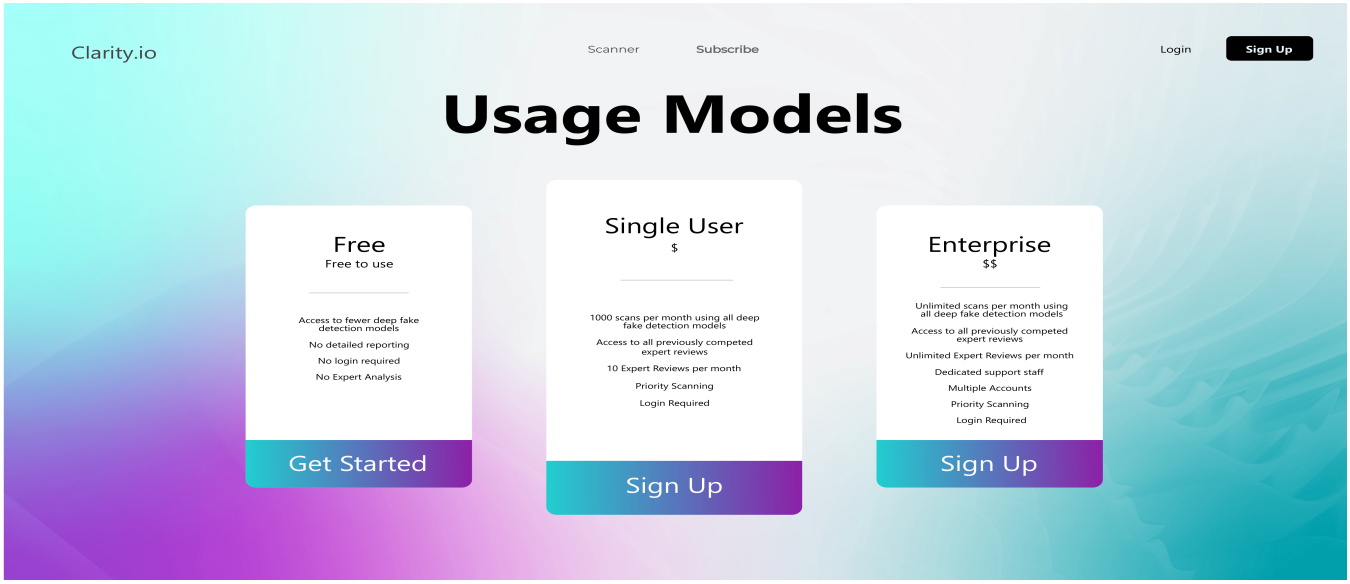


Figure 2: Diagram showing potential customer per month subscription models of the Clarity system following the design scheme of the user interface.

4. CLARITY: A CLOUD BASED DEEPFAKE DETECTION PLATFORM

4.1 Overall System Architecture:

Clarity makes use of a microservice architecture [79] to meet the scalability requirements discussed in section 3. A microservice architecture is a single software system that is comprised of multiple

small, independently developed and deployed services [80]. These services interact using Application Program Interfaces (API) to send and receive necessary data [81]. Each microservice follows the 'single responsibility principle' in its design and should only perform at most one functional requirement of the system. Microservice architectures seek to address scalability issues present within traditional monolithic software systems, that consist of a singular code-base that performs all functionality of the system. Monolithic systems are quicker to develop and deploy, have faster performance due to the centralised code-base, and are easier to debug than microservice systems [82]. However, monolithic systems lack hardware and software scalability, flexibility in terms of new services, reliability in the event of a component failure, and suffer difficulties when deploying changes as the system becomes large [82]. The microservice approach was chosen over the monolithic for this application for a number of reasons. Firstly, the Clarity system may need to be massively scaled after its deployment, especially as the need to identify deepfakes is an ever increasing trend. The use of microservices allows efficient hardware scaling through the use of services such as Google Cloud Platform (GCP) [83], Amazon Web Services (AWS) [84], or Microsoft Azure [85]. This minimizes the expertise, cost, and environmental impact involved in creating and maintaining in-house servers. Secondly, the use of microservices ensures that well defined RESTful APIs are implemented. This enforces modularity with each service representing an independent function of the system. Each service can be developed, deployed, updated, tested, and scaled independently, without worry of influencing other services and inducing unwanted bugs. Thirdly, due to the isolated nature of microservices, multiple instantiations of a single service can be deployed concurrently. This increases performance through massive functional parallelism, which is more complex to implement in monolithic systems.

The overall Clarity system architecture is shown in figure 3. It is comprised of seven back-end microservices, namely the Account, Payment, Billing, Customer Management, Expert Management, Deepfake Reporting, and Deepfake Scanner services. Furthermore, third party payment services are used to increase security and reduce critical user information stored by the system, as per the Protection of Personal Information (POPI) Act [86] of South Africa. The system also makes use of central API Gateway and Load Balancing services. The API Gateway receives client side API requests and distributes them to the correct back-end service; it then returns any necessary responses to the client when they become available [87]. This gateway service decouples the client-side user interface microservices from the back-end system. The load balancing service distributes network traffic among the back-end service deployments as efficiently as possible to reduce latency in the system [87]. As the popularity of the Clarity system increases, the use of a Content Delivery Network (CDN) to distribute the system services geographically could be employed. This would copy content and services to servers around the world and, coupled with the load balancing service, would ensure reduced latency to a worldwide audience although at an added cost. Finally, there are two front-end browser based services that complete the architecture. These are the Customer UI and the Expert UI microservices. The specific functions of each microservice are discussed in sections 4.1.1 - 4.1.8 below, however, the Deepfake Scanner microservice is the heart of the Clarity system and so is discussed in more detail independently in section 4.2. The back-end storage and database concerns for the microservices are discussed in section 4.3.

The Clarity system is designed as cloud based, meaning that there are no in-house servers or databases required to deploy the system [88]. Cloud computing platforms offer access to services in the form of networks, storage devices, servers, and complex applications which perform well at scale on distributed hardware [88]. The decision to use cloud computing is preferable for Clarity, as deep learning systems are computationally expensive for short periods of time, for example during model training, meaning that a pay-as-you-use approach is appropriate. Maintaining state-of-the-art equipment is costly and requires significant expertise, thus, outsourcing this to large cloud platform providers is the most feasible option. Deployment on the cloud can be handled in a few ways, the two most relevant are a 'containerization' and a 'serverless' approach. Both provide minimal overhead, high computing performance, hardware outsourcing, and are supported by the large cloud platform providers, however they differ in a few key ways [89]. Serverless systems follow the Function as a Service (FaaS) framework, meaning that functions are deployed to the cloud on a per-use basis [90]. These functions are lightweight, and easily deployable and testable, however, they create third party dependencies, are not transparent, and can become complicated to keep track of as the system becomes large [89].

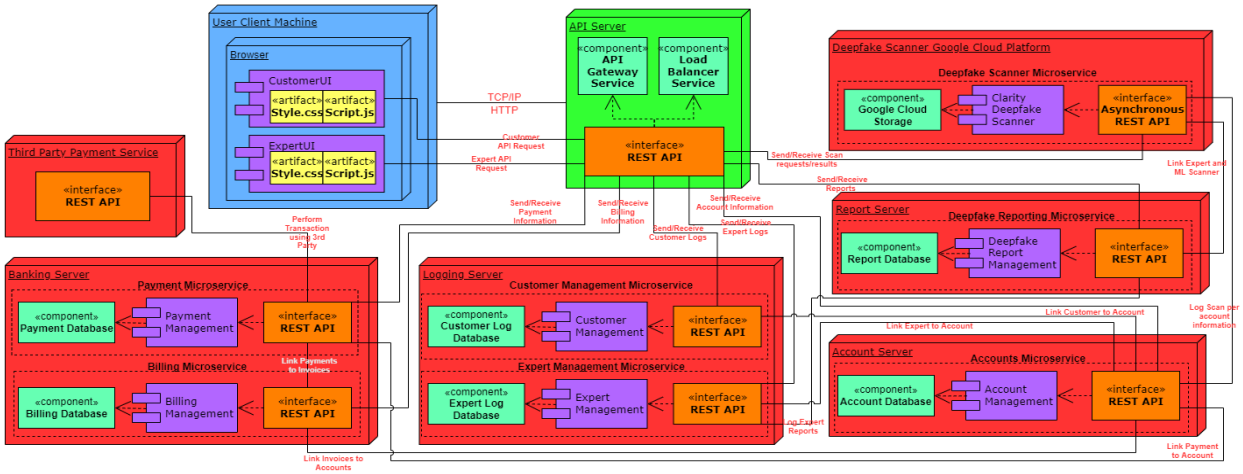


Figure 3: Deployment diagram showing each microservice of the Clarity deepfake detection system with corresponding hardware, internal stack, and relevant inter-service communications.

Containers, on the other hand, are essentially lightweight virtual machines that are code and system agnostic. They package an application and all its dependencies into a single object, and use Platform as a Service (PaaS) methodologies. Containerization has the benefit of added control when managing service polices, resources, and security [89]. It is also highly scalable and easy to deploy. There are some disadvantages such as slightly slower response times when compared to a serverless approach, and potential data losses due to container failures [89]. The Clarity system microservices will be deployed with a widely used, supported, and open-source containerization platform known as Docker [91]. This choice is due to the flexibility and control offered by containerization, as well as its scalability and security benefits. The transparent nature of Docker containers is a requirement for a system such as Clarity that promises explainability.

The choice of cloud computing platform is critical to the deployment and development process of the system. There are three leading cloud platforms, namely, AWS, GCP, and Azure. Each has its respective strengths, levels of support, and applicability to the Clarity system. All systems offer varying support for computing, storage, ML, databases, backups, caching, security, load balancing, and automation. The choice of the GCP for the Clarity system is discussed in section 4.5. It must be noted that in order to scale the Docker based microservices, a container orchestration tool must be utilised. A well-known, industry supported, open-source tool known as Kubernetes [92] can be used to deploy, scale, and manage Docker containers during run-time. Kubernetes was designed by Google, and is well-supported on the GCP with their Google Kubernetes Engine (GKE) [93] which is optimized to run on the cloud.

4.1.1 Account Microservice: The account microservice is responsible for handling all critical user information relating to login credentials, subscription types, and user types. Its connection to other services is mainly for validation purposes to ensure a particular user is permitted to access another service. This is a simple service with a focus on security and efficiency. All user information, especially passwords, should be encrypted using at minimum SHA-512 with salting. The information stored regarding the user must be limited and secured in conformity with the POPI act. Java is a suitable language for this service due to its high speed coupled with wide support, as this service will experience high traffic. However, the choice of language is flexible for this service.

4.1.2 Payment Microservice: The payment microservice is responsible for handling payments through the system, both to and from users. It connects mainly to the billing service to indicate that a specific invoice has received a payment, as well as to the account service to mark payments against a specific user. The service itself does not process any payments or require storage of user payment information, for instance credit card information. This is to ensure compliance with the POPI act. The payments are processed, as indicated in figure 3, by third party services. PayPal [94] is a good choice for the

Clarity system as it is in compliance with South African and International regulation, and is designed for use with online systems.

4.1.3 Billing Microservice: The billing microservice is responsible for handling bills and invoices for the users of the system. All upcoming and historic user bills and invoices are stored and processed in this service. As mentioned previously its main transfer of information occurs with the payment system. It is also linked to the account microservice to ensure invoices correspond to specific users. Python [95] is a suitable language for the development of this service due to its extensive use in, and library support for, finance technology.

4.1.4 Customer Management Microservice: The customer management microservice is responsible for logging and processing all customer activity on the Clarity platform. All payments, scans, queries, and front-end usage is monitored and stored in this service. The service exchanges information with all parts of the system to log which customer is performing a specific customer action. It can be used to create detailed log reports for customers and developers. It must be noted that in compliance with the POPI act, no client-side browser information that is unrelated to critical system actions are logged. This service requires efficiency and low latency to monitor customer activities, thus either C++ or Java are useful languages, however, the choice of language is flexible for this service.

4.1.5 Expert Management Microservice: The expert management microservice is responsible for logging and processing all expert activity on the Clarity platform. All reports, analysis, result disputes, and other front-end usage is monitored and stored in this service. The service exchanges information with the account microservice as well as the deepfake reporting microservice to log an expert's performance metrics. It can be used to create detailed reports for experts and developers. This service requires efficiency and low latency to monitor expert activities, thus either C++ or Java are useful languages, however, the choice of language is flexible for this service.

4.1.6 Deepfake Reporting Microservice: The deepfake reporting microservice is responsible for storing and processing both expert reports and deepfake scanner automated reports. The service uses a stored hashed value for a given input to rapidly display results for previously seen URLs, images, and videos. The service exchanges data with the expert UI microservice, the customer UI microservice, and the deepfake scanner. Customers can obtain reports from this service, while experts can upload custom reports for a specific scanner result. The choice of language is flexible for this service.

4.1.7 Customer UI Microservice: The customer user interface microservice is responsible for front-end communication with customers and free users. It communicates through the API Gateway service to relay information to and from the back-end microservices. The interface is browser based and should be supported by all major browsers, including Google Chrome, Microsoft Edge, Firefox, and Safari. This service consists of JavaScript routines to perform dynamic functionality, as well as HTML and CSS for styling. More details regarding the customer UI are provided in section 4.6.

4.1.8 Expert UI Microservice: The expert user interface microservice is responsible for front-end communication with experts. It communicates through the API Gateway service to relay information to and from the back-end services. The interface is login protected to ensure that only authorised personnel may utilise its features. The main purpose of this service is to provide an interface through which experts may write detailed reports regarding a specific deepfake detection decision. For this reason its design is simplistic and provides only the media to be analysed, an option to download the media, and a text editor box in which to report on the reason for a detection decision. The interface is browser based and should be supported by all major browsers, including Google Chrome, Microsoft Edge, Firefox, and Safari. This service consists of JavaScript routines to perform dynamic functionality, as well as HTML and CSS for styling.

4.2 Deepfake Scanner Architecture:

The Clarity deepfake scanner has been designed with MLOps development principles in mind. The scanner architecture and workflow is shown in figure 4. Each component and tool used in the workflow was chosen based on its support on the GCP, its proprietary or open-source licence, and its capacity to integrate with other components in the system.

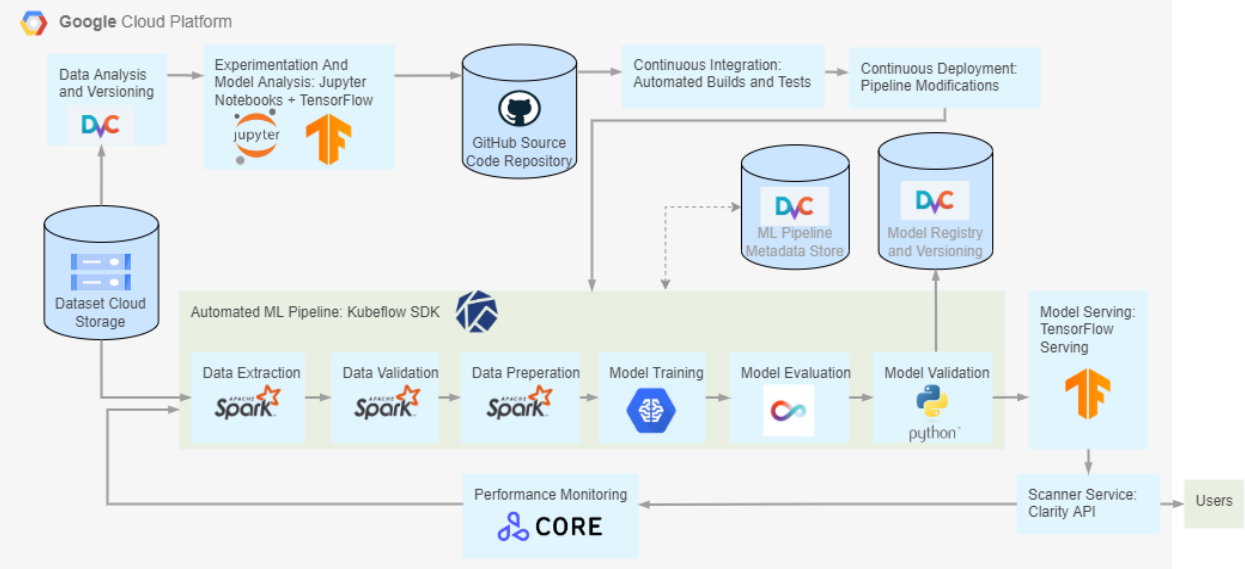


Figure 4: A diagram showing the Clarity Deepfake Scanner architecture and deployment workflow, with all relevant tools, libraries, and platforms used to provide the prediction service.

The system utilises Google Cloud Storage to persistently store training media. This choice is discussed in section 4.3.2 and the incurred costs are mentioned in section 4.8. The scanner makes use of an automated end-to-end ML deployment pipeline to provide a well-tested, robust, and efficient scanner that is always available. Python [95] is used throughout this service as it is an industry standard for machine learning and is supported by all the tools discussed below. Experimentation and model development is performed using Jupyter Notebooks, a web-based computing platform with rich-text and equation support [96]. This standardisation ensures that development is well documented. The models themselves will be implemented using TensorFlow [97] which is an industry standard with many predefined model libraries and support. TensorFlow is open-source and was developed by Google, meaning that it is widely supported on the GCP. Proper versioning of the code-base is to be performed using Git [98] with GitHub [99] serving as a repository. Kubeflow [100], an open-source ML deployment platform, was chosen as the backbone of the workflow due to its support on GCP as well as Kubernetes. This ensures the scanner can integrate seamlessly into Clarity's other microservices and be orchestrated alongside them.

The first part of the pipeline includes data extraction, data validation, and data preparation. This is performed by Apache Spark [101], an open-source, analytics engine built for large scale data processing. This represents the pre-processing phase of the ML system. Spark is designed to operate on distributed systems, such as GCP, with increased data parallelism and fault-tolerance. The data extraction phase of the system involves separating the data into batches of fake and real media for training. The data validation phase ensures that no media is corrupted, and that all media has at least one detectable face that can be scanned. The data preparation phase involves resizing images, adding compression to the data for the purpose of variance, and extracting frames from videos. If the model being trained is detecting temporal characteristics then frames will not be extracted and the entire video will act as an input. An important aspect of ML systems design is reproducibility; this involves ensuring that the data used in training is well documented. Data Version Control (DVC) [102] is an open source versioning system that is similar to Git [98], but for machine learning models and large datasets. Its use is discussed further in section 4.3.2.

The second part of the pipeline involves model training, model evaluation, and model validation. Model training occurs using the Cloud ML Engine service of GCP which utilises distributed GPU and TPU technology to train models in parallel on a pay-per-second scheme. Model Evaluation is performed using Continuous Machine Learning (CML) [103] which is an open-source model analytics platform that provides automatically generated reports for ML models. This service provides the necessary confusion matrices for a given set of model tests as well as accuracy, precision, and recall metrics. Finally, model validation can be performed using in-house python scripts which cross-validate selected hyper-parameters against previous model deployments. DVC is not only useful for its data versioning, but also for its model and ML pipeline metadata versioning. This improves the reproducibility of the system and allows the scanner to be "rolled back" to previous deployments with a practically identical system state. This is a key feature of the Clarity system as updates can never permanently break the system, and exact replicas of the scanner are available in the event of legal inquiries.

Finally, the model is served to the Clarity Scanner Service API via the TensorFlow Serving platform. This, coupled with the Seldon Core [104] open-source serving and monitoring platform, permits the deployment of an ensemble of trained models to be used in the Clarity system. Seldon Core is compatible with Kubernetes allowing it to scale rapidly. Furthermore, its Prometheus component can be used to provide detailed monitoring of the deployed system performance. This is useful to provide feedback to the developers for future versions of the scanner. If particular models perform poorly on real world deepfakes, this can be identified and the models replaced.

4.3 Data and Storage:

4.3.1 Overall Storage and Databases: The Clarity back-end microservices make use of a database per service pattern. This means that each microservice has its own database and corresponding database software [105]. All microservices, with the exception of the Deepfake Scanner, make use of an SQL relational database. These services need to be available and consistent as per the CAP theorem [106] to ensure that information provided to the users, and other back-end services, is always accessible at the latest version. The GCP provides support for a variety of relational Database Management Systems (DBMS). MySQL [107] is an open source, high performance, flexible, and scalable DBMS that is supported by many cloud platforms, including GCP. A logical database schema, showing all microservices, their tables and relations, and their respective interactions, is given in figure 5. The schema is designed in third normal form to ensure minimal record repetitions. The dotted inter-service lines are loosely coupled dependencies, however, they were included to show expected data exchange between service APIs. GCP also provides disaster recovery to prevent data losses and ensures scalability of the MySQL system despite lacking partition tolerance.

4.3.2 Deepfake Scanner Data and Back-end Concerns: Google Cloud Storage was chosen due to its large capacity at low cost. The choice of a schemaless storage option to house the deepfake detection datasets, shown in table 1, is twofold. Firstly, these datasets are not standardised meaning that unnecessary processing would be required to convert them into a relational state. Secondly, relational databases are expensive to access when large BLOB media files are involved, and often have restrictions on single object size. This problem is circumvented by coupling the cloud storage directly to the machine learning pipeline using GCP functionality. Storage in the cloud is also flexible and allows more datasets to be added as required without needing to upgrade and maintain in-house storage facilities. Furthermore, this allows all media inputs by users to be stored and assessed, permitting the creation of a new Clarity deepfake detection dataset over the system's life cycle. This feature ensures that Clarity is consistently monitoring and defending against improvements in the deepfake generation state-of-the-art.

As mentioned previously, data versioning is another important aspect of the Clarity system. The use of DVC ensures that the data used to train a model, or a given set of models, is versioned appropriately by replacing all the data for a given instance with a set of metafiles. These point to the relevant data, rather than reproducing it entirely. It must be noted that similar metafiles are used in the model and deployment pipeline versioning discussed in section 4.2.

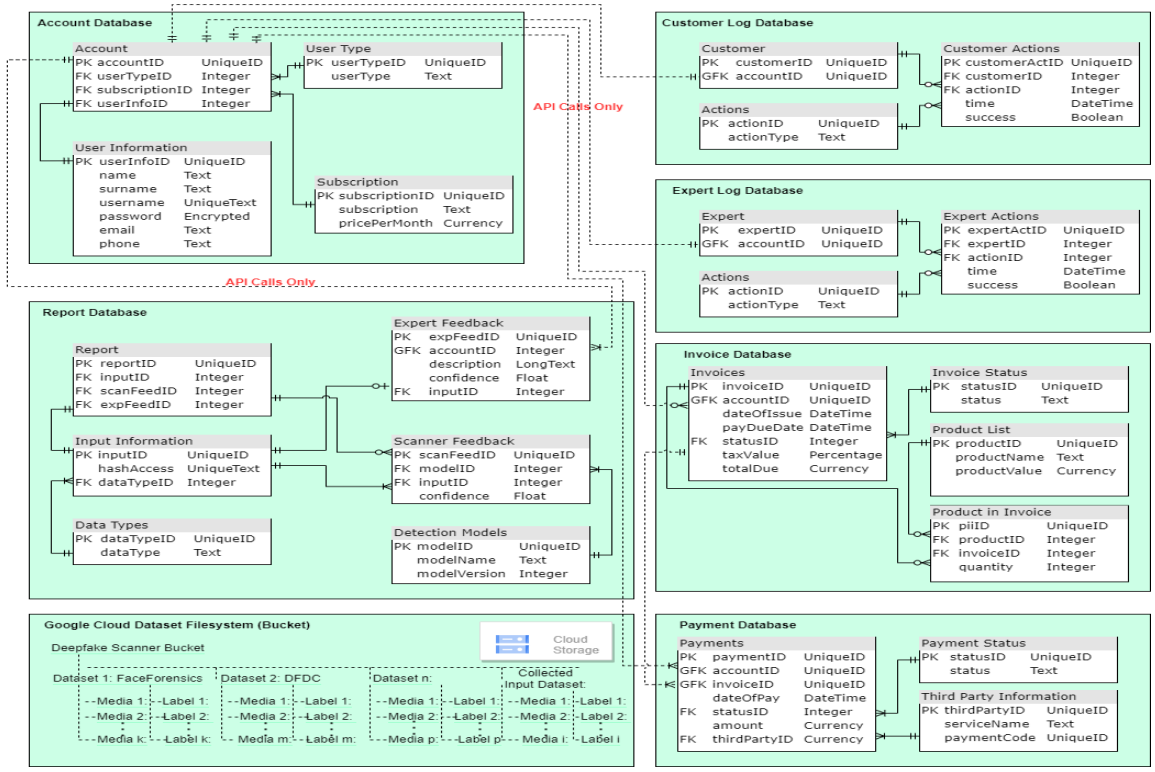


Figure 5: A logical database schema showing the tables, relations, and columns of each microservice of the Clarity Deepfake Detection Platform. The schemaless Deepfake Scanner storage system is provided for completeness.

4.4 Detection Model Concerns:

The model selection process is crucial to the Clarity system. The overall prediction accuracy, detection inference time, environmental impact, training time, and cost of the system are affected by the models chosen. The system is designed to run initially using an ensemble of six models which provide classification scores as described in section 3.2. The models are chosen such that each of the five deepfake generation techniques discussed in section 2.1.1 are adequately detectable by the system. Five of the six models are trained using a combination of datasets from table 1 to ensure high accuracy on a unique generation technique. The final model is trained using knowledge distillation and incremental learning to detect deepfakes of all generation types. This ensures the robustness of the Clarity system. Each model has two relevant features: Architecture and Training Methodology. The EfficientNet-B4, XceptionNet, and VGG-16 architectures have seen success in state-of-the-art detection and are used in the ensemble. The final models which form the ensemble are as follows:

- VGG-16 trained to identify face swaps using a sub-set of datasets.
- EfficientNet-B4 trained to identify entire face synthesis using a sub-set of datasets.
- EfficientNet-B4 trained to identify lip-syncing using a sub-set of datasets.
- EfficientNet-B4 trained to identify face reenactment using a sub-set of datasets.
- VGG-16 trained to identify facial attribute manipulation using a sub-set of datasets.
- XceptionNet trained using knowledge distillation and incremental learning on all available datasets.

It must be noted that while the described models have shown state-of-the-art performance in deepfake detection, the landscape is constantly changing and evolving. The focus of the Clarity system is not the exact models or methods used to detect specific deepfakes. Rather, the system provides a platform on top of which new and improved techniques can continually be deployed in the wild. Furthermore, the media collected from real world inputs over time can be used to train new models. This mitigates potential concept drift in the detection system by requiring occasional full system retraining with the latest possible datasets.

4.5 System Hardware:

Developing and maintaining the hardware to run a software system is costly and requires significant expertise. Furthermore, it is generally not a scalable approach to deploy web based systems using in-house infrastructure. The Infrastructure as a Service (IaaS) model provided by the cloud companies is more cost effective and outsources expertise requirements. Cloud services are also exactly scalable which makes them more environmentally friendly than in-house servers which waste the unused computational power while active.

The choice between AWS, Azure, and GCP was an important factor in the overall design of the system. AWS is the largest and most establish cloud computing service [108]. They offer the greatest functionality of all three services. However, they are more costly when it comes to general purpose computing. Their ML services, including Amazon SageMaker, are state-of-the-art for MLOps with distributed GPU capabilities [108]. AWS has the greatest global coverage for CDN systems. Azure is the second largest cloud provider in terms of market share [108]. They provide seamless integration with Microsoft tools, which is often useful for enterprise systems, and boast a wide feature set. GCP is the smallest of the three in terms of market share. However, it is the most cost effective for general purpose computing which suits the microservice back-end structure. GCP is committed to supporting open-source software [108]; this is extremely beneficial as it allows the use of many free open-source third party tools to be utilised whenever necessary. This adds to the cost effectiveness of GCP. Furthermore, GCP, similar to other providers, includes load balancing, caching, and CDN services to reduce system latency. It also has full support for Docker and Kubernetes which are necessary for the microservice system design. The most valuable hardware support offered by GCP, however, are its Tensor Processing Units (TPUs). These are processors tailored to machine learning purposes which can offer a speed-up of up to an order of magnitude for certain ML computations. TPUs appear to outperform GPUs in terms of both efficiency and power consumption [109]. This is especially true when batched training occurs, which is directly applicable to the image classification problem the deepfake scanner is designed to solve. A more detailed cost breakdown is discussed in section 4.8. These factors made GCP the best choice for the Clarity system.

4.6 User Interface and Customer Experience:

The customer user interface is designed to meet the input/output requirements discussed in section 3. This is shown in figure 6, and provides a clear report displaying the overall detection decision and confidence index. Additionally, the "ask an expert" button is included to provide the aforementioned expert reports. The "disagree with assessment" functionality allows users to bring obviously incorrect detection decisions to the attention of experts and developers. This is the report functionality available to free users as seen by the use of the XceptionNet model only. A modern colour scheme is used throughout the front-end. More detail on other user interface components is provided in Appendix B.

4.7 Security and Robustness:

Security of the Clarity system is of the utmost importance. Any compromise in information or inability to provide detection services will induce a lack of user trust and undermine the purpose of the system. A pertinent vulnerability is a potential Distributed Denial of Service (DDoS) attack. This is an attack in which the Clarity service is flooded with requests leading to extreme latency and potentially large system costs. The login functionality of the system ensures that free-users can never impede the services available to paying customers. Additionally, if a paid user attempts a DDoS attack, their account can be quickly suspended with relevant legal action taken against their known information. DDoS attacks affecting free-users can be mitigated through relevant monitoring software and metrics provided by the GCP. Banning of specific IP addresses which have performed DDoS attacks is another way to slow the progress of attackers. Machine Learning platforms suffer other types of adversarial attacks, including evasion, poisoning, inference, and extraction attacks. Evasion attacks occur during model testing and are designed to cause errors in the system [110]. Poisoning attacks involve the inclusion of specific data into training to cause detection weaknesses [111]. Inference attacks involve the users obtaining private system information through basic requests using the user interface [112]. Finally, extraction attacks occur when an attacker uses numerous requests to steal the entire model [113]. These attacks are performed by both external and internal adversaries. The Python machine

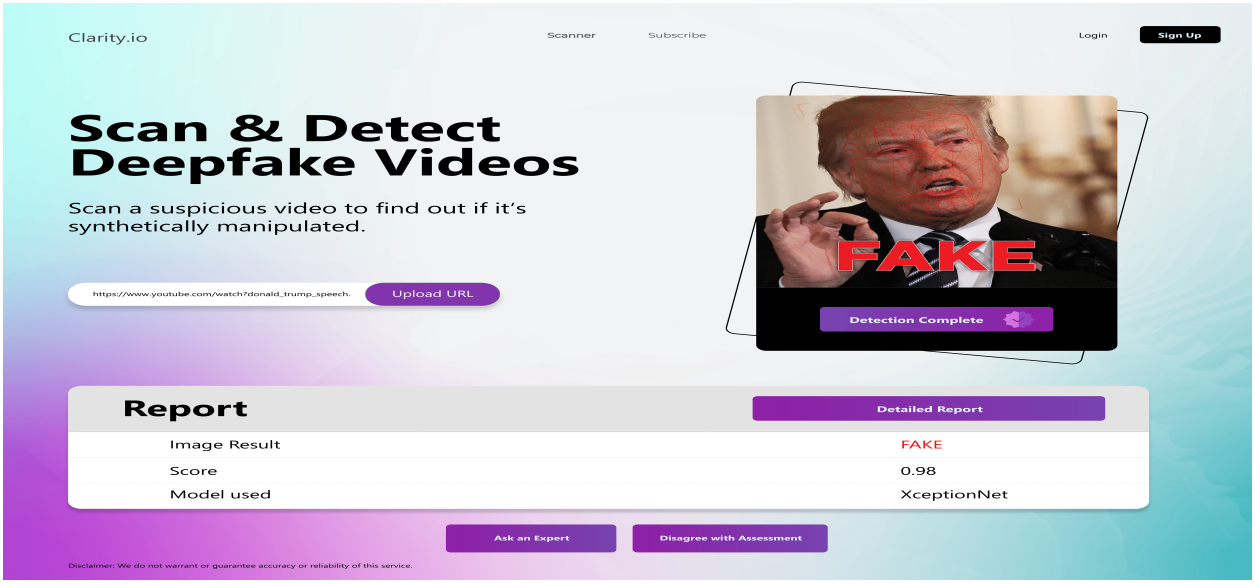


Figure 6: A diagram showing a potential customer user interface layout which shows relevant Clarity functionality after a completed scan. The scanner report is shown with a URL input. Furthermore, the "Ask an Expert" and "Disagree with Assessment" functionality is displayed.

learning library, known as Adversarial Robustness Toolbox (ART) [114], provides a series of defense mechanisms against these attacks that will be utilised in Clarity for increased safety.

4.8 Costs and Environmental Concerns:

4.8.1 Total System Costs: The prudent use of open-source platforms reduces software costs to essentially zero. The IaaS, however, incurs a monthly cost based on the choice of components and services used. A cost breakdown for GCP and necessary IaaS components is shown in table 2. At first glance, it may appear that GPUs are more cost effective than TPUs on a per chip basis, however, TPUs provide speed-up which lessens required training time. This could be used to improve model accuracy at a fixed cost, or to save cost by training within a fixed training time. Either way, TPUs offer greater flexibility and performance for this ML problem than GPUs.

Table 2: A table detailing the IaaS costs of the Google Cloud Platform (GCP) services. This includes general computing (GC) [2], storage [3], Database Management Services (DBMS) [4], load-balancing (LB) [5], caching [6], and ML [7] costs. All of these services are to be used in the Clarity system with long-term discounting applied.

Cloud Service Provider Costs						
	GC (\vCPU hour)	Storage (\GB \month)	DBMS (\vCPU \hour)	LB (First 5 Rules\hour)	Caching Egress (\GB)	ML TPU v3-32 (\month)
GCP	$0.009815	$0.020	$0.01982	$0.025	≈ $0.10	$10,512
	GC Memory (\GB hour)	Storage (\10,000 operations)	DBMS Memory (\GB\hour)	LB (Per Rule \hour after 5)	Caching Fill (\GB)	ML: 8 × NVIDIA A100 80GB (\month)
GCP	$0.001316	$0.05	$0.00336	$0.01	≈ $0.025	$8117.80

4.8.2 Carbon-footprint discussion: No exact carbon footprint calculation can be provided for a scalable system that is in its design phase. However, certain design decisions will drastically lower the systems negative environmental impact and carbon-footprint. Firstly, the choice of neural network

training hardware is vital as these are computationally expensive tasks. Secondly, the overall carbon emission, power usage model, and power source of the cloud platform indicates the negative environmental impact of a cloud-based system. There are two metrics which are relevant to a distributed hardware system's carbon-footprint. Firstly, the power consumption as seen in 1 [109], and the carbon-emission as seen in 2 [109].

$$kWh = \frac{HtT \times Proc_n \times PPP_{Ave} \times PUE}{1000} \tag{1}$$

Where kWh are the total Kilowatt hours of the system, HtT are the hours required to train a given model, $Proc_n$ are the number of GPUs or TPUs, PPP_{Ave} is the average power per processor unit, and PUE is the power usage effectiveness as a percentage [109].

$$tCO_2e = \frac{kWh \times kgCO_2}{1000} \tag{2}$$

Where tCO_2e are the metric tons of carbon dioxide emitted and $kgCO_2$ are the kilograms of carbon-dioxide per kWh of the carbon emitting power source [109]. These metrics can be used to determine the exact carbon footprint of the GCP TPUs measured against their GPUs. It has been shown that TPUs, on average, utilise less power than GPUs when performing similar tasks [109]. Furthermore, GCP offers varying levels of support for "going green" based on the region chosen to host the Clarity system [115]. For instance, the Zurich region boasts an 85% carbon free energy usage with a remarkably low local grid emission intensity of 86 gCO_2e/kWh [115]. This coupled with its centralised location, and comparable closeness in region to South Africa, makes it an efficient low carbon choice for initial deployment of Clarity. Beyond this, GCP offers monitoring support for carbon emissions [116], meaning that the entire Clarity system carbon-footprint can be automatically monitored throughout its life-cycle. Finally, GCP aims to be "carbon-free" by 2030, which means the carbon-footprint of Clarity should decrease over time.

5. DEPLOYMENT LIFE-CYCLE

Clarity, as it has been shown, is a large and complex system. It is not feasible to suggest that a single engineer implement, maintain, and continually improve this system. Rather the development should begin with the implementation of a simple Minimum Viable Product (MVP) which consists of a basic, free-to-use, detection platform, with a single detection model, on the web. This version of Clarity can then be gradually scaled, ported to the cloud, and containerized. As funding is secured, either through industry, government, or gradual increases in sales, greater functionality on the GCP can be utilised. It is vital that the monthly IaaS costs never outweigh the cash inflow of the system. However, with the growing interest in deepfake generation, and subsequent widespread need for its detection, it is likely that a few thousand paid users, at $10 per month, could be obtained to cover the costs of the Clarity system described in this paper. From there, the system can be maintained, improved, and stand at the forefront of deepfake detection services as the customer base increases.

6. CONCLUSION

The design of the cloud based Clarity platform for detecting deepfakes on the internet has been described. Its loosely coupled microservice back-end architecture makes the design scalable and flexible, while still maintaining a high overall performance with low latency. The system makes use of industry standard MLOps principles to automate the machine learning pipeline and ensure reproducible, explainable, transparent, and secure detection models that accurately identify the full spectrum of deepfakes in the wild. Clarity's use of the Google Cloud Platform ensures cost reductions due to its support of open-source software and its availability of efficient TPU hardware for model training. Furthermore, it guarantees reduced carbon-footprint over manually deployed hardware by outsourcing computation, storage, and networking to low carbon emission grids. Potential monetization schemes for the Clarity system have been devised to ensure longevity and financial feasibility. With this design, another step has been taken towards limiting the negative effect of deepfakes on the modern world.

REFERENCES

[1] F. Juefei-Xu, R. Wang, Y. Huang, Q. Guo, L. Ma, and Y. Liu, "Countering malicious deepfakes: Survey, battleground, and horizon," 2021. Available: https://arxiv.org/abs/2103.00218

[2] "General computing pricing," Google Cloud, 2022. Available: https://cloud.google.com/compute/all-pricing (Accessed 2022-09-08).

[3] "Cloud storage pricing," Google Cloud, 2022. Available: https://cloud.google.com/storage/pricing (Accessed 2022-09-08).

[4] "Cloudsql for mysql pricing," Google Cloud, 2022. Available: https://cloud.google.com/sql/docs/mysql/pricing (Accessed 2022-09-08).

[5] "All networking pricing," Google Cloud, 2022. Available: https://cloud.google.com/vpc/network-pricing (Accessed 2022-09-08).

[6] "Cloud cdn pricing," Google Cloud, 2022. Available: https://cloud.google.com/cdn/pricing (Accessed 2022-09-08).

[7] "Cloud tpu pricing," Google Cloud, 2022. Available: https://cloud.google.com/tpu/pricing (Accessed 2022-09-08).

[8] D. Fallis, "The epistemic threat of deepfakes," *Philosophy & Technology*, vol. 34, no. 4, pp. 623–643, Dec 2021. Available: https://doi.org/10.1007/s13347-020-00419-2

[9] M. Westerlund, "The emergence of deepfake technology: A review," *Technology Innovation Management Review*, vol. 9, pp. 39–52, 11 2019.

[10] S. Adee, "What are deepfakes and how are they created?" *IEEE Spectrum*, 2020. Available: https://spectrum.ieee.org/what-is-deepfake (Accessed 2022-09-02).

[11] M. Albahar and J. Almalki, "Deepfakes: Threats and countermeasures systematic review," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 22, pp. 3242–3250, 2019.

[12] M. Masood, M. Nawaz, K. M. Malik, A. Javed, A. Irtaza, and H. Malik, "Deepfakes generation and detection: State-of-the-art, open challenges, countermeasures, and way forward," *Applied Intelligence*, pp. 1–53, 2022.

[13] K. M. Sayler and L. A. Harris, "Deep fakes and national security," Congressional Research SVC Washington United States, Tech. Rep., 2020.

[14] T. Nguyen, C. M. Nguyen, T. Nguyen, T. Duc, and S. Nahavandi, "Deep learning for deepfakes creation and detection: A survey," 09 2019.

[15] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia, "Deepfakes and beyond: A survey of face manipulation and fake detection," 2020. Available: https://arxiv.org/abs/2001.00179

[16] "Fakeapp," 2018. Available: https://www.malavida.com/en/soft/fakeapp/ (Accessed 2022-09-04).

[17] "Faceswap: Deepfakes software for all," 2018. Available: https://github.com/deepfakes/faceswap (Accessed 2022-09-04).

[18] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing obama: Learning lip sync from audio," *ACM Trans. Graph.*, vol. 36, no. 4, jul 2017. Available: https://doi.org/10.1145/3072959.3073640

[19] K. Vougioukas, S. Petridis, and M. Pantic, "End-to-end speech-driven facial animation with temporal gans," 2018. Available: https://arxiv.org/abs/1805.09313

[20] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of rgb videos," 2020. Available: https://arxiv.org/abs/2007.14808

[21] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8798–8807.

[22] J. Kietzmann, L. Lee, I. McCarthy, and T. Kietzmann, "Deepfakes: Trick or treat?" *Business Horizons*, vol. 63, 12 2019.

[23] I. Yerushalmy and H. Hel-Or, "Digital image forgery detection based on lens and sensor aberration," *International Journal of Computer Vision*, vol. 92, pp. 71–91, 03 2011.

[24] A. Popescu and H. Farid, "Exposing digital forgeries in color filter array interpolated images," *Signal Processing, IEEE Transactions on*, vol. 53, pp. 3948 – 3959, 11 2005.

[25] H. Cao and A. Kot, "Accurate detection of demosaicing regularity for digital image forensics,"

Information Forensics and Security, IEEE Transactions on, vol. 4, pp. 899 – 910, 01 2010.

[26] Z. Lin, J. He, X. Tang, and C.-K. Tang, "Fast, automatic and fine-grained tampered jpeg image detection via dct coefficient analysis," *Pattern Recognition*, vol. 42, no. 11, pp. 2492–2501, 2009. Available: https://www.sciencedirect.com/science/article/pii/S0031320309001198

[27] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "A sift-based forensic method for copy–move attack detection and transformation recovery," *Information Forensics and Security, IEEE Transactions on*, vol. 6, pp. 1099 – 1110, 10 2011.

[28] D. Cozzolino, G. Poggi, and L. Verdoliva, "Splicebuster: A new blind image splicing detector," in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2015, pp. 1–6.

[29] A. Gironi, M. Fontani, T. Bianchi, A. Piva, and M. Barni, "A video forensic technique for detecting frame deletion and insertion," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6226–6230.

[30] Y. Wu, X. Jiang, T. Sun, and W. Wang, "Exposing video inter-frame forgery based on velocity field consistency," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 2674–2678.

[31] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics++: Learning to detect manipulated facial images," 2019. Available: https://arxiv.org/abs/1901.08971

[32] Y. Mirsky and W. Lee, "The creation and detection of deepfakes," *ACM Computing Surveys*, vol. 54, no. 1, pp. 1–41, jan 2022. Available: https://doi.org/10.1145%2F3425780

[33] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015. Available: https://arxiv.org/abs/1511.08458

[34] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, mar 2020. Available: https://doi.org/10.1016%2Fj.physd.2019.132306

[35] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013. Available: https://arxiv.org/abs/1312.6114

[36] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. Available: https://arxiv.org/abs/1406.2661

[37] M. S. Rana and A. H. Sung, "Deepfakestack: A deep ensemble-based learning technique for deepfake detection," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2020, pp. 70–75.

[38] S. Pashine, S. Mandiya, P. Gupta, and R. Sheikh, "Deep fake detection: Survey of facial manipulation detection solutions," 2021. Available: https://arxiv.org/abs/2106.12605

[39] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "MesoNet: a compact facial video forgery detection network," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, dec 2018. Available: https://doi.org/10.1109%2Fwifs.2018.8630761

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. Available: https://arxiv.org/abs/1512.03385

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. Available: https://arxiv.org/abs/1409.1556

[42] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2016. Available: https://arxiv.org/abs/1610.02357

[43] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019. Available: https://arxiv.org/abs/1905.11946

[44] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020. Available: https://arxiv.org/abs/2005.12872

[45] M. Kim, S. Tariq, and S. S. Woo, "Fretal: Generalizing deepfake detection using knowledge distillation and representation learning," 2021. Available: https://arxiv.org/abs/2105.13617

[46] M. Kim, S. Tariq, and S. S. Woo, "Cored: Generalizing fake media detection with continual representation using distillation," in *Proceedings of the 29th ACM International Conference on Multimedia*, ser. MM '21. New York, NY, USA: Association for Computing Machinery, 2021,

p. 337–346. Available: https://doi.org/10.1145/3474085.3475535
[47] S. Baxevanakis, G. Kordopatis-Zilos, P. Galopoulos, L. Apostolidis, K. Levacher, I. B. Schlicht, D. Teyssou, I. Kompatsiaris, and S. Papadopoulos, "The mever deepfake detection service: Lessons learnt from developing and deploying in the wild," 2022. Available: https://arxiv.org/abs/2204.12816
[48] B. Zi, M. Chang, J. Chen, X. Ma, and Y.-G. Jiang, "Wilddeepfake: A challenging real-world dataset for deepfake detection," 2021. Available: https://arxiv.org/abs/2101.01456
[49] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11651
[50] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, and C. C. Ferrer, "The deepfake detection challenge (dfdc) dataset," 2020. Available: https://arxiv.org/abs/2006.07397
[51] Y. Li, M.-C. Chang, and S. Lyu, "In ictu oculi: Exposing ai generated fake face videos by detecting eye blinking," 2018. Available: https://arxiv.org/abs/1806.02877
[52] P. Korshunov and S. Marcel, "Deepfakes: a new threat to face recognition? assessment and detection," 2018. Available: https://arxiv.org/abs/1812.08685
[53] B. Dolhansky, R. Howes, B. Pflaum, N. Baram, and C. C. Ferrer, "The deepfake detection challenge (dfdc) preview dataset," 2019. Available: https://arxiv.org/abs/1910.08854
[54] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, "Celeb-df: A large-scale challenging dataset for deepfake forensics," 2019. Available: https://arxiv.org/abs/1909.12962
[55] U. A. Ciftci, I. Demir, and L. Yin, "Fakecatcher: Detection of synthetic portrait videos using biological signals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.
[56] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. Jain, "On the detection of digital face manipulation," 2019. Available: https://arxiv.org/abs/1910.01717
[57] N. Dufour and A. Gully, "On the detection of digital face manipulation," *Google AI Blog*, 2019. Available: https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html (Accessed 2022-09-04).
[58] L. Jiang, R. Li, W. Wu, C. Qian, and C. C. Loy, "Deeperforensics-1.0: A large-scale dataset for real-world face forgery detection," 2020. Available: https://arxiv.org/abs/2001.03024
[59] X. Dong, J. Bao, D. Chen, W. Zhang, N. Yu, D. Chen, F. Wen, and B. Guo, "Identity-driven deepfake detection," 2020. Available: https://arxiv.org/abs/2012.03930
[60] Y. He, B. Gan, S. Chen, Y. Zhou, G. Yin, L. Song, L. Sheng, J. Shao, and Z. Liu, "Forgerynet: A versatile benchmark for comprehensive forgery analysis," 2021. Available: https://arxiv.org/abs/2103.05630
[61] J. Pu, N. Mangaokar, L. Kelly, P. Bhattacharya, K. Sundaram, M. Javed, B. Wang, and B. Viswanath, "Deepfake videos in the wild: Analysis and detection," 2021. Available: https://arxiv.org/abs/2103.04263
[62] T. Zhou, W. Wang, Z. Liang, and J. Shen, "Face forensics in the wild," 2021. Available: https://arxiv.org/abs/2103.16076
[63] T.-N. Le, H. H. Nguyen, J. Yamagishi, and I. Echizen, "Openforensics: Large-scale challenging dataset for multi-face forgery detection and segmentation in-the-wild," 2021. Available: https://arxiv.org/abs/2107.14480
[64] "Deepfake scanner," Deepware, 2021. Available: https://deepware.ai/
[65] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
[66] "Deepmedia," DeepMedia, 2022. Available: https://www.deepmedia.ai/ (Accessed 2022-09-05).
[67] H. Field, "Meet the company working with the air force to detect deepfakes," Emerging Tech Brew, 2022. Available: https://www.emergingtechbrew.com/stories/2022/08/22/meet-the-company-working-with-the-air-force-to-detect-deepfakes?utm_campaign=etb&utm_medium=newsletter&utm_source=morning_brew&mid=f37fb5df9fea7eb468dd205b3590e1bb (Accessed 2022-09-05).
[68] "Detect deepfakes using our software," DuckDuckGoose, 2022. Available: https://www.duckduckgoose.ai/ (Accessed 2022-09-05).

[69] "Deepfake-o-meter," Buffalo, 2020. Available: http://zinc.cse.buffalo.edu/ubmdfl/deep-o-meter/ (Accessed 2022-09-05).
[70] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, and D. Dennison, "Hidden technical debt in machine learning systems," *NIPS*, pp. 2494–2502, 01 2015.
[71] P. Jha and R. Khan, "A review paper on devops: Beginning and more to know," *International Journal of Computer Applications*, vol. 180, pp. 16–20, 06 2018.
[72] "Machine learning operations," MLOps, 2022. Available: https://ml-ops.org/ (Accessed 2022-09-05).
[73] P. Canuma, "Mlops: What it is, why it matters, and how to implement it," neptune.ai, 2022. Available: https://neptune.ai/blog/mlops (Accessed 2022-09-05).
[74] G. Symeonidis, E. Nerantzis, A. Kazakis, and G. A. Papakostas, "Mlops – definitions, tools and challenges," 2022. Available: https://arxiv.org/abs/2201.00162
[75] D. A. Tamburri, "Sustainable mlops: Trends and challenges," in *2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2020, pp. 17–23.
[76] A. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in deploying machine learning: a survey of case studies," *ACM Computing Surveys*, apr 2022. Available: https://doi.org/10.1145%2F3533378
[77] "Build your open-source mlops stack," MyMLOps, 2022. Available: https://mymlops.com/ (Accessed 2022-09-05).
[78] "A curated list of awesome mlops tools," GitHub, 2022. Available: https://github.com/kelvins/awesome-mlops (Accessed 2022-09-05).
[79] O. Al-Debagy and P. Martinek, "A comparative review of microservices and monolithic architectures," in *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, 2018, pp. 000 149–000 154.
[80] N. Dragoni, S. Giallorenzo, A. Lluch-Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," 06 2016.
[81] R. Chen, S. Li, and Z. E. Li, "From monolith to microservices: A dataflow-driven approach," 12 2017, pp. 466–475.
[82] C. Harris, "Microservices vs. monolithic architecture," Atlassian, 2022. Available: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith. (Accessed 2022-09-06).
[83] "Cloud computing services," Google Cloud, 2022. Available: https://cloud.google.com/ (Accessed 2022-09-06).
[84] "Cloud computing services - amazon web services (aws)," AWS, 2022. Available: https://aws.amazon.com/ (Accessed 2022-09-06).
[85] "Cloud computing services: Microsoft azure," Microsoft Azure, 2022. Available: https://azure.microsoft.com/en-us/ (Accessed 2022-09-06).
[86] "Protection of personal information act," 2013. Available: https://www.gov.za/sites/default/files/gcis_document/201409/3706726-11act4of2013protectionofpersonalinforcorrect.pdf
[87] J. Zhao, S. Jing, and L. Jiang, "Management of api gateway based on micro-service architecture," *Journal of Physics: Conference Series*, vol. 1087, p. 032032, 09 2018.
[88] P. K. Paul and M. K. Ghose, "Cloud computing: Possibilities, challenges and opportunities with special reference to its emerging need in the academic and working area of information science," *Procedia Engineering*, vol. 38, pp. 2222–2227, 2012, international Conference On Modelling Optimization And Computing. Available: https://www.sciencedirect.com/science/article/pii/S1877705812021807
[89] Y. Kuznetsova, A. Kolomytsev, M. Syomochkin, and O. Vdovitchenko, *Serverless and Containerization Models and Methods in Challenger Banks Software*, 01 2021, pp. 169–185.
[90] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, J. E. Gonzalez, R. A. Popa, I. Stoica, and D. A. Patterson, "Cloud programming simplified: A berkeley view on serverless computing," 2019. Available: https://arxiv.org/abs/1902.03383
[91] "Docker," Docker, 2022. Available: https://www.docker.com/ (Accessed 2022-09-06).

[92] "Production-grade container orchestration," Kubernetes, 2014. Available: https://kubernetes.io/ (Accessed 2022-09-07).
[93] "Google kubernetes engine (gke)," Google Cloud, 2022. Available: https://cloud.google.com/kubernetes-engine (Accessed 2022-09-07).
[94] "Digital wallets, money management, and more: Paypal us," PayPal, 1998. Available: https://www.paypal.com/ (Accessed 2022-09-07).
[95] "Welcome to python.org," Python.org, 1991. Available: https://www.python.org/ (Accessed 2022-09-07).
[96] "Project jupyter," Project Jupyter, 2018. Available: https://jupyter.org/ (Accessed 2022-09-08).
[97] "Tensorflow," TensorFlow, 2015. Available: https://www.tensorflow.org/ (Accessed 2022-09-08).
[98] "Git," Git, 2005. Available: https://git-scm.com/ (Accessed 2022-09-08).
[99] "Github," GitHub.Inc, 2008. Available: https://github.com/ (Accessed 2022-09-08).
[100] "Kubeflow," Kubeflow, 2018. Available: https://www.kubeflow.org/ (Accessed 2022-09-08).
[101] "Apache spark™ - unified engine for large-scale data analytics," Apache Spark, 2014. Available: https://spark.apache.org/ (Accessed 2022-09-08).
[102] "Data version control," DVC, 2017. Available: https://dvc.org/ (Accessed 2022-09-08).
[103] "Continuous machine learning (cml)," Iterative.ai, 2022. Available: https://cml.dev/ (Accessed 2022-09-08).
[104] "Seldon core," Seldon, 2021. Available: https://www.seldon.io/solutions/open-source-projects/core (Accessed 2022-09-08).
[105] A. Messina, R. Rizzo, P. Storniolo, M. Tripiciano, and A. Urso, "The database-is-the-service pattern for microservice architectures," vol. 9832, 09 2016, pp. 223–233.
[106] A. Fox and E. Brewer, "Harvest, yield, and scalable tolerant systems," 03 1999, pp. 174 – 178.
[107] "Mysql," MySQL, 1995. Available: https://www.mysql.com/ (Accessed 2022-09-07).
[108] "Aws vs azure vs gcp – the cloud platform of your choice?" Veritis, 2021. Available: https://www.veritis.com/blog/aws-vs-azure-vs-gcp-the-cloud-platform-of-your-choice/ (Accessed 2022-09-07).
[109] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," 2021. Available: https://arxiv.org/abs/2104.10350
[110] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," 2017. Available: https://arxiv.org/abs/1709.05583
[111] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," 2018. Available: https://arxiv.org/abs/1804.00792
[112] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, "Membership inference attacks on machine learning: A survey," 2021. Available: https://arxiv.org/abs/2103.07853
[113] T. Takemura, N. Yanai, and T. Fujiwara, "Model extraction attacks against recurrent neural networks," 2020. Available: https://arxiv.org/abs/2002.00123
[114] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. M. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.0.0," 2018. Available: https://arxiv.org/abs/1807.01069
[115] "Carbon free energy for google cloud regions," Google Cloud, 2022. Available: https://cloud.google.com/sustainability/region-carbon (Accessed 2022-09-08).
[116] "Carbon footprint," Google Cloud, 2022. Available: https://cloud.google.com/carbon-footprint (Accessed 2022-09-08).

APPENDIX A: Non-Technical Report

Shedding Some Clarity on the Deepfake Epidemic

What is a deepfake?

Deepfakes, the latest in synthetically manufactured fake media, are images or videos, generally of people, that have been created using cutting-edge deep learning technology. Deep learning is a type of machine learning that uses extensive computing power to learn complicated tasks, such as playing chess, or in this case, how to manipulate someones face to make them appear to be doing something they have never done before. The algorithms "learn" by training on thousands of images and videos of people. They then recreate these people doing whatever the person producing the deepfake desires. Surely you have seen videos of celebrities making strange comments in robotic voices with warped lips and jittering eyes? That is a deepfake.

Why are deepfakes dangerous?

The aforementioned celebrity deepfake can be spotted from a mile away. So why should we worry if a few, obviously fake, images and videos are floating around the internet? This question will be answered with another question. Is figure 1 real or fake?

Figure 1: A potentially fake image of a person.[1]

If you answered real, then congratulations! You have fallen victim to the latest in deepfake generation technology. If you answered fake, then you are correct. This person does not exist. This image is the result of one of many new techniques for generating deepfakes. These methods are publicly available, permitting almost anyone to create a deepfake of their own, although sometimes at the cost of having to leave their computer running over night. Now, while *we* may be content generating images of a non-existent person smiling, some people have more nefarious uses in mind. These techniques can be used to create realistic images and videos of anyone saying and doing anything. This permits individuals to create pornographic content of unwilling individuals, or videos of politicians and celebrities spreading misinformation, or even to blackmail and bully friends and family. Without any way to tell what is real and what is fake we are at risk of a complete breakdown of social trust.

How do we stop the deepfake spread?

Unfortunately, the truth is, we cannot. The ease-of-access to powerful computers in the cloud and the necessary deep learning software makes it simpler than ever before to create hyper-realistic, potentially harmful and slanderous media. This coupled with the advent of social media and the rapid proliferation of information around the world via the internet means that deepfakes are here to stay. However, all is not lost. While we humans may not be able to identify deepfakes, we know someone, or rather something, that can. By fighting fire with fire and turning the deepfake's own creators against them, we can utilise deep learning to identify deepfakes with a very high accuracy. All that is missing are dedicated platforms, which are easily accessible, that can be used to scan an image or video with these deep learning identification techniques. This would provide the needed clarity on whether given media is real or fake. This is where Clarity: A Cloud Based Deepfake Detection Platform, finds its purpose. Clarity is a web service that is designed to be supported on your favourite browser. It provides an easy-to-use interface that lets anyone upload an image, video, or URL to scan and check if it is a deepfake. The URLs need to point to Twitter or Facebook posts, or YouTube videos for the scanner to provide meaningful results. These results come in the form of reports that describe how confident Clarity is that a given input is a deepfake. If it thinks its real; it will tell you. If it thinks its fake; it will tell you. If its suspicious but not sure; it will tell you that as well! Just as a bonus, Clarity will also tell you what deep learning model it used to identify the deepfake. And if, after all that, you disagree with the prediction, there are feedback features with which you can let the Clarity system know. Addition-

[1] `https://thispersondoesnotexist.com/`

ally, for some added cost, experts in the field of deep learning and deepfakes can provide a more thorough explanation regarding a given deepfake detection result. So this is how to stop the spread of deepfakes through social, political, economic life, not by destroying them or leaving them be, but rather by providing some Clarity.

How does Clarity work?

Clarity uses a combination of different state-of-the-art deep learning techniques in parallel to ensure that it can detect a wide range of deepfake types. It is cloud based, meaning that it is run on external hardware built by third party providers. Google Cloud Platform was chosen to host the Clarity system. As mentioned previously, training these deep learning models to generate deepfakes can take a long time on a personal computer. The same can be said for the deep learning models used to detect deepfakes. This makes Google Cloud Platform an excellent resource when preparing the detection system, as they provide many specialised computers which are all linked together and can train the model very quickly.

What does all this cost?

Clarity is free-to-use should you want to try out the scanning functionality. The free version has only one detection model. However, paid users receive login credentials and the ability to use all six detection models. This increases Clarity's accuracy, as well as speeds up its performance. The fee for such services is small and is offered on a pay-per-month scheme. A monthly $10 should cover the costs required to maintain the system and provide the user unlimited access to advice and reports from experts should they require it. Smaller monthly payments could also be made, however this would incur fewer Clarity benefits.

Can we trust Clarity?

Clarity makes use of Google Cloud Platform's built-in, industry standard, security to ensure that user information is safe and encrypted. The system is in compliance with the Protection of Public Information (POPI) Act with regards to data collection. This means that no user data is obtained that is not absolutely necessary to system functioning. Furthermore, the detection models used are trained on the latest and most modern deepfakes to ensure the highest levels of performance at all times. Also, models and predictions are saved. This means that if, in the future, a prediction is needed again for whatever reason, it can be provided. This includes situations in which predictions are required for legal reasons. In such cases the entire model itself can be brought as evidence for the detection. These factors, coupled with the advice provided by the deepfake experts to paying customers makes Clarity a trustworthy and reliable system.

Are there any downsides to Clarity?

The most notable downside to the Clarity system is the power consumption, and corresponding carbon-footprint, caused by training the detection models, and running the system in general. Deep learning systems are notorious for using lots of energy, thus generating higher carbon emissions than other software systems. This in turn creates high-cost for development and maintenance of such systems.

How are these issues being addressed?

Google Cloud Platform provides a few ways in which the overall carbon emissions of the system can be reduced. Firstly, it provides impressive monitoring services that allow developers to observe which changes cause the highest increases in energy consumption and which changes cause the system to be more efficient. Secondly, it provides access to global servers that run on low carbon emission grids. These are grids powered by renewable energy such as wind, solar, or hydroelectric systems. Finally, Google Cloud Platform itself has promised to "go green" by 2030. This means that by using their systems, Clarity's carbon-footprint will reduce automatically over time.

What does the future look like?

The future appears bright, despite the impending prevalence of deepfakes in the world. Clarity, and an army of similar platforms, will rise up against each new deepfake generation method. We will never again know a world without deepfakes, but with adequate and continually maintained detection services, the world will at least have some semblance of Clarity as it moves into this new era of media.

APPENDIX B: Additional User Interface Designs

This section provides further detail on possible user interface designs for the Clarity Deepfake Scanner customer interface. Figure 7 shows the main deepfake scanner page before an input is provided by the user. Figure 8 shows the login screen for the system which can be used by both customer and expert user types.

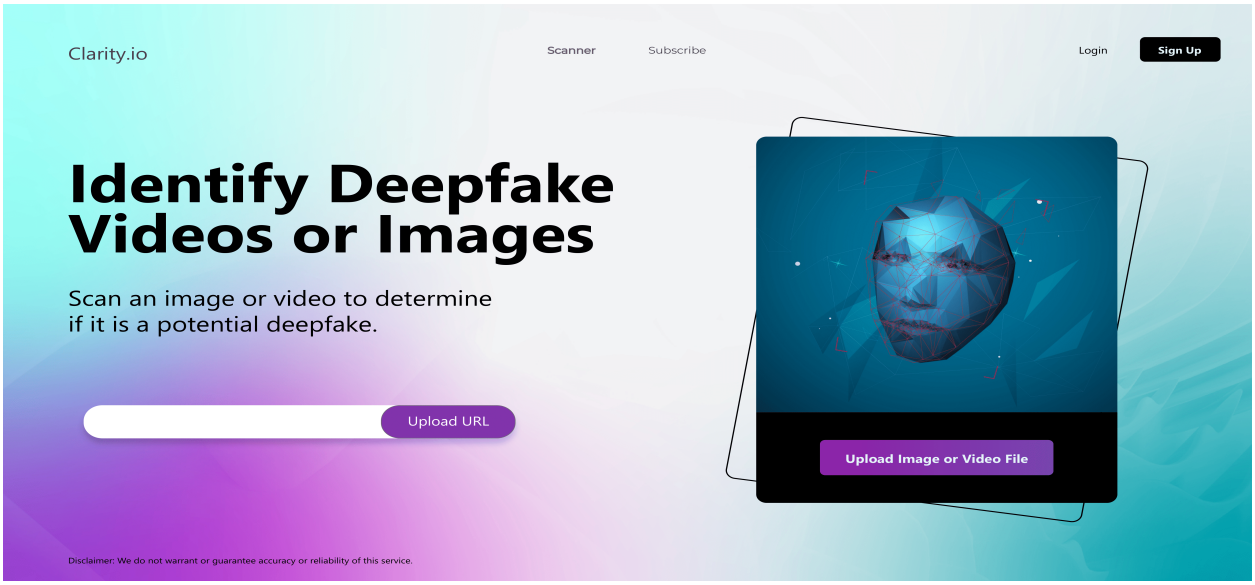


Figure 7: A diagram showing a potential customer user interface layout which shows relevant Clarity functionality before completion of a scan. The scanner input options of URLs, images, and videos are made clear to the user.

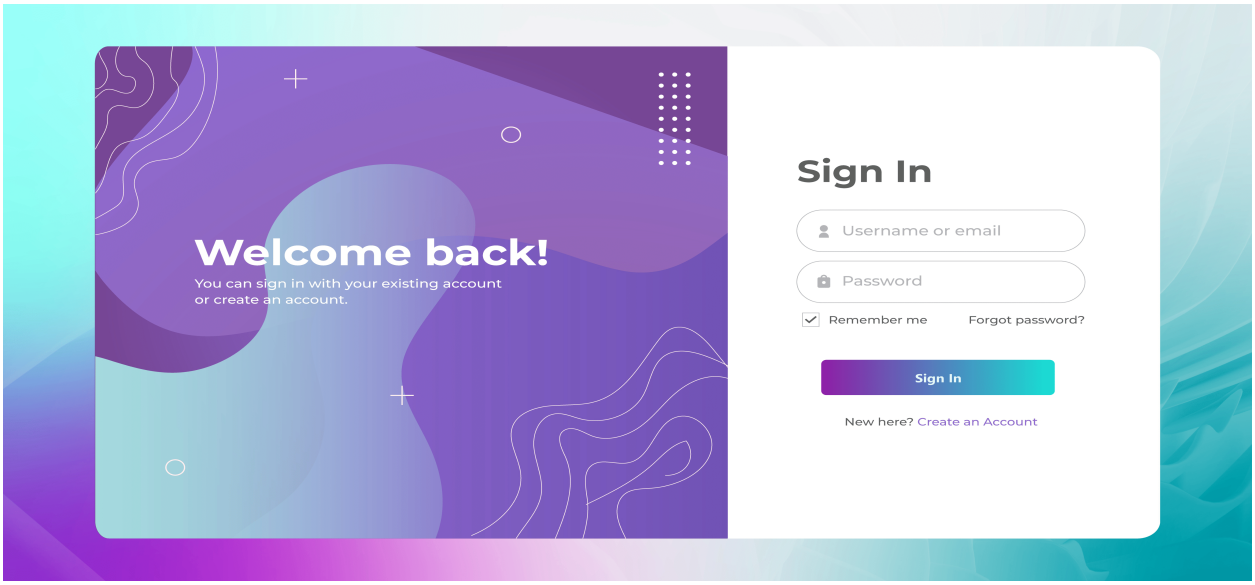


Figure 8: A diagram showing a login screen design for the system following a similar colour scheme and layout to the other UI components.