PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزارة التعـلـيـم الـعـالــي و الـبـحـث الـعـلـمـي
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
جـامـعـة عمّــار ثـلـيــجــي بالأغــواط
UNIVERSITY OF AMAR TELIDJI LAGHOUAT
كـلـيــــة الـعـلــــوم
FACULTY OF SCIENCES
COMPUTER SCIENCE DEPARTMENT

MASTER THESIS

Domain : Mathematics and Computer Science
Field : Computer Science
Option : Networks, Systems and Distributed Applications

by:
KORIBA Hadjer

THEME

Deepfakes Detection Using Deep Learning

Thesis defended publicly on *20-06-2022*

Jury Members

Dr. CHAIB Noureddine	*Associate Professor*	*President*
Dr. GUELLOUMA Younes	*Associate Professor*	*Examiner*
Dr. BENAROUS Leila	*Associate Professor*	*Supervisor*

N …… University Year 2021/2022

Acknowledgment

First, I thank Allah for giving me the courage and the

will to accomplish this humble job.

Words cannot express my gratitude to my supervisor

Dr. Leila BENAROUS for her valuable comments

her patience and feedback.

A special thanks to Saida Boudouh for her advices

motivatio and guidance.

I couldn’t have achieved this without your help and support.

I also extend my thanks to the jury members who kindly
agreed to judge my work and benefit me with their remarks.

And I would like to express my deep gratitude to all my
Professors at Computer Science Department, Laghouat University
who earnestly and generously contributed in my education and training

Finally, I cannot conclude these thanks without mentioning all
the people, from near or far, who gave me their help or sympathy.

Dedication

I first want to thank - ها - for the strength and patience
that he gave me to overcome all the trials
(Happiness and misfortunes)
experienced during this work.
I dedicate this work from the bottom of my heart to my dear parents
for their support,patience and sacrifices.
And to my lovely sisters Sara, Meriem,my dear brother
Yacine and his lovely wife and son Bouchra and Siradj.
A special thanks to my dearest friend Afaf
for been the best friend i could ever ask for
I'm So grateful for your friendship.
I also want to thank my dear friends:
Saida,Safa,Rahil,Fatna,Sara,Ines,Zhour,Bouchra,Manel,
Nassima,Fatima,Ilhem, Lilia,Kheira,Keltoum.
Bahi,Youssef,Youcef,Mounir,Mehdi,Aziz,Salah,Amine,Ramil.

ملخص

Deepfakes أو التقليد الواقعي للمحتوى السمعي البصري الأصيل، هي تقنيات منتشرة على نطاق واسع خاصة باستخدام شبكة الخصومة التوليدية المدربة مسبقًا (GANs) التي تسهل تبديل وجه الشخص تلقائيًا بآخر في مقطع فيديو. نظرًا لتأثيراتها المدمرة على العالم، أصبح التمييز بين مقاطع الفيديو الحقيقية ومقاطع الفيديو المزيفة مشكلة أساسية. لذلك، فإن الأساليب الآلية لتحديد مقاطع فيديو Deepfake هذه مطلوبة في ضوء الفضائح العامة الأخيرة. تم تكريس الكثير من الأبحاث لتطوير طرق الكشف لتقليل التأثير السلبي المحتمل للتزييف العميق.

في هذه الدراسة، طورنا منصة مفتوحة المصدر تسمى «Deepfake Detection»، وهي تقدم تقنية اكتشاف جديدة تتكون من جزأين: الجزء الأول هو نموذج تصنيف ثنائي يمكنه تصنيف مقاطع الفيديو على أنها مزيفة أو حقيقية. الجزء الثاني هو نموذج مختلف مع مدخلات ومخرجات مختلفة عن الأول. يمكنه تحديد طريقة التوليد من بين هؤلاء الثلاثة: تم استخدام FaceSwap وFace2Face وDeepFakes لإنشاء هذا التزييف العميق (التصنيف حسب الفئات). لقد جمعنا عيناتنا من مجموعة بيانات .FaceForensics++ كانت مرحلة ما قبل المعالجة ضرورية لهذه الدراسة، واستخراج الصور، قص الوجوه (المنطقة ذات الأهمية) وزيادة البيانات. بعد ذلك قمنا بتقسيم بياناتنا إلى مجموعات تدريب واختبار. بعد ذلك، تم استخدام الشبكات العصبية التلافيفية (CNNs) وأساليب نقل التعلم لهذا العمل، وقمنا بتطبيق سبعة نماذج تم تدريبها مسبقًا على CNN في الجزء الأول (التصنيف الثنائي)، مع العديد من التجارب ومعايير ضبط مختلفة لتحديد النموذج الأكثر ملاءمة لتطبيقنا بالإضافة إلى المعايير التي أثرت على نتائجنا. النماذج المختارة التي تم تدريبها مسبقًا على CNN هي: VGG16, Inception-v3, InceptionResNet-v2, Xception, MobileNet-V2, MobileNet-V3 و ResNet50-V2. بناءً على نتائج التقييم، وصلنا إلى أعلى دقة بنسبة 100٪ مع ResNet50-V2 في كلا الجزأين (التصنيف الثنائي والفئوي). أخيرًا، قمنا بتطوير تطبيق الويب الخاص بنا، للمستخدمين للتفاعل مع نماذجنا.

الكلمات الرئيسية: التزييف العميق، التصنيف، GANs، المنطقة ذات الأهمية ، نقل التعلم ، CNNs، ResNet50-V2، FaceForensics++.

Abstract

Deepfakes or the hyper-realistic imitation of authentic audio-visual content, are widely spread techniques specially with the use of pre-trained generative adversarial network (GANs) that makes it easier to automatically swap a person's face with another in a video. Due to their devastating impacts on the world, distinguishing between real and deepfake videos has become a fundamental issue. Therefore, automated methods to identify these deepfake videos are required in light of recent public scandals. Much research has been devoted to developing detection methods to reduce the potential negative impact of deepfakes.

In this study, we developed an open-source platform called « Deepfake Detection », it presents a new detection technique which consists of two parts: the first part is a binary classification model that can classify videos as fake or real. The second part is another model with different input and output from the first one. It can identify which generation method among these three: FaceSwap, Face2Face and DeepFakes were used to create these deepfakes (categorical classification). We collected our samples from FaceForensics++ dataset. The preprocessing phase was necessary for this study, from frame extraction to face cropping (Region of Interest) and data augmentation. After that we split our data into train and test sets. Next, Convolutional Neural Networks (CNNs) and transfer learning approaches were employed for this work, we implemented Seven CNN-pretrained models in the first part (binary classification), with several trials and different fine-tuning parameters to determine which model is the most suitable for our situation as well as the criteria that influenced our results. The selected CNN-pretrained models are: VGG16, Inception-v3, InceptionResNet-v2, Xception, MobileNet-V2, MobileNet-V3 and ResNet50-V2. Based on the evaluation results, we reached the highest accuracy of 100% with ResNet50-V2 in both parts (the binary and categorical classification). Lastly, we developed our web application, for users to interact with our models.

Keywords: Deepfakes, classification, GANs, ROI, transfer learning, CNNs, FaceForensics++, ResNet50-V2.

Résumé

Les deepfakes ou l'imitation hyperréaliste de contenu audio-visuel authentique, sont des techniques largement répandues, spécialement avec l'utilisation d'un réseau génératif

adversaire (GANs) pré-entrainé qui rend plus facile de permuter automatiquement le visage d'une personne avec une autre dans une vidéo. En raison de leurs effets dévastateurs sur le monde, la distinction entre les vidéos réelles et les deepfakes est devenue une question fondamentale. Par conséquent, des méthodes automatisées pour identifier ces vidéos deepfakes sont nécessaires à la lumière des scandales publics récents. De nombreuses recherches ont été consacrées à l'élaboration de méthodes de détection pour réduire l'impact négatif potentiel des deepfakes.

Dans cette étude, nous avons développé une plateforme open-source appelée « Deepfake Detection », qui présente une nouvelle technique de détection composée de deux parties : la première partie est un modèle de classification binaire qui peut classer les vidéos comme fausses ou réelles. La deuxième partie est un modèle différent avec des entrées et sorties différentes de la première. Il peut identifier quelle méthode de génération parmi ces trois : FaceSwap, Face2Face et DeepFakes ont été utilisés pour créer ces deepfakes (classification catégorique). Nous avons recueilli nos échantillons à partir de l'ensemble de données FaceForensics++. La phase de prétraitement était nécessaire pour cette étude, de l'extraction de images, de rognage des visages (région d'intérêt) et à l'augmentation des données. Après cela, nous avons divisé nos données en des ensembles deux ensembles « train », et « test ». Ensuite, des réseaux de neurones convolutifs (CNN) et des approches d'apprentissage par transfert ont été utilisés pour ce travail, nous avons mis en œuvre sept modèles CNN préétablis dans la première partie (classification binaire), avec plusieurs essais et différents réglages des paramètres pour déterminer quel modèle est le plus adapté à notre situation ainsi que les critères qui ont influencé nos résultats. Les modèles CNN pré-entrainé sélectionnés sont : VGG16, Inception-v3, InceptionResNet-v2, Xception, MobileNet-V2, MobileNet-V3 et ResNet50-V2. D'après les résultats de l'evaluation, nous avons atteint la précision la plus élevée, soit 100 %. Nous avons atteint la plus haute précision de 100% avec ResNet50-V2 dans les deux parties (la classification binaire et catégorique). Enfin, nous avons développé notre application web pour que les utilisateurs puissent interagir avec nos modèles.

Mots-clés : Deepfakes, classification, GANs, région d'intérêt, apprentissage par transfert, CNNs, FaceForensics++, ResNet50-V2.

Table of content

List of figures

LIST OF TABLES

LIST OF ACRONYMS

AI: Artificial Intelligence

ML: Machine Learning

DL: Deep Learning.

DNN: Deep Neural Networks

CNN: Convolutional Neural Network.

GAN: Generative Adversarial Networks

VAE: Variational Autoencoders

DFDC: Deepfake Detection Challenge Dataset

LSTM: Long short-term memory

RNN: Recurrent Neural Network

SVM: Support Vector Machine.

LRCN: Long-term Recurrent Convolutional Networks

ROI: Region of Interest

API: Application Programming Interface

GAP: Global Average Pooling

GMP: Global Max Pooling

LR: Learning Rate

AUC: Area Under Curve

ROC: Receiver Operating characteristic

SGD: Sophisticated Gradient Descent

GUI: Graphical User Interface.

HDF5: High-performance data management and storage suite

INTRODUCTION

INTRODUCTION

Over the last decades, the popularity of smartphones and the growth of social networks have made digital images and videos relatively popular. According to several reports, almost two billion pictures are uploaded everyday on the internet. And more than 100 million hours of video content is watched daily on social networks, this widespread use of digital images and videos has led to a rise of techniques to alter image and video contents, such as, employing editing software like Photoshop, video processing and mounting tools and so on. These social networks manipulations leave us wondering whether statements like “believe what you see” or “camera never lies” are still valid!

Although, people used to trust Radio/TV stations news, they no longer do so nowadays. The Internet enabled a non-linear media distribution model but it did not guarantee its news authenticity. Due to the fact that any Internet user can alter and re-distribute news and media contents through social media networks (e.g., YouTube, Facebook, Twitter, etc.).

A new emerging technique has been revealed under the aegis of computer vision and deep learning technology, that allows anyone to generate incredibly convincing fake videos, images, and even manipulate voices. It is widely known as Deepfake Technology. Even if you've never heard of the word deepfake before, chances are you've seen dozens of deepfake videos online and didn't realize it. Deepfake or AI-manipulated videos, are ones in which have been manipulated with the help of artificial intelligence and neural network to either swap one person's face to another, or modify the individual's face to make them appear to say or do something they never said or done (the hyper-realistic imitation of authentic audio-visual content). Even though it seems amusing and fun to create deepfakes, however, there’s another side to the coin, this technology can pose a big threat to the trustworthiness of online media. It can either be used for harmless entertainment purposes or to mislead, deceive and harm.

Deepfakes has brought a growing concern when it comes to global stability, cyber risks, deceptive media content and privacy violation. such as, revenge pornography, hoaxes, financial fraud, fake news, political disputes, blackmail, bullying, insulting and many more.

For example, recent investigations revealed large-scale use of deepfakes in pornography, hence turning deepfakes into a threat to people's reputation. Also online theft cases in which scammers successfully used deepfake imitation to trick company’s employees into wiring money to the scammers.

powerful Artificial Intelligence (AI) technology, in particular, deep neural networks (DNNs), and the unprecedented computing power have made it possible, easier and more

convincing creating such sophisticated fake videos, they're faking it and they're getting better by the day. Thus, there is a pressing need to combat the rise of deepfakes, especially ones that will be used in a malicious manner. One promising countermeasure against them is deepfake detection.

In less than three years, there have been numerous new detection methods in the academic research on deepFakes. However, differences in training datasets, hardware, and learning architectures across research publications make rigorous comparisons of different detection algorithms challenging. Their objective is creating deepfake detection methods able to detect a forensics trace hidden in images: a sort of fingerprint left in the image generation process, that cannot be easily recognized as real or fake by human eye.

Several big companies have decided to take action against this phenomenon such as, Google, Facebook, AWS, Microsoft, and Amazon.Thus, the main purpose of this research is to create deepfake detection method that can classify a video as fake or real. Our innovation in this study is to detect the name of the method (among these three methods: FaceSwap, Face2Face, DeepFakes) that these fake videos was created by. We will choose FaceForensics++ dataset for this work, then we will do some necessary preprocessing for data augmentation and to achieve better performance and accuracy. After that, we will use transfer learning which allows us to implement convolutional neural networks pretrained models that have been already trained on bigger datasets (ImageNet), then, we will move on to the fine-tuning phase. Therefore, we will implement seven pretrained models: VGG16, Inception-v3, Inception-ResNet-v2, Xception, MobileNet-V2, MobileNet-V3 and ResNet50-V2. As a result, we will select the best suitable one using its specific parameters and any additional specific criteria that may have an impact on our results. Finally, we will implement our Front-end (web application) in order for users to interact with our Back-end.

This work is divided into three chapters: the first one discusses the growth of deepfakes and its harm to society, our motivation for this research. Some prevention methods, such as deepfake detection and its existence in literature. The second chapter, illustrates our dataset collection, preprocessing, dataset splitting and pretrained model's implementation. The Last chapter presents our Back-end, Front-end and our experiment results.

Finally, we summarize this work with a general conclusion and future perspectives.

CHAPTER I:
LITERATURE REVIEW OF EXISTING SYSTEMS

1. Introduction

In the past, “fake face” would normally be associated with plastic surgery or makeup but as the people shifted their interest to the cyberworld, and the use of technologies emerged, almost anyone now can create and publish through social media networks stories or images or videos from anywhere, anytime. As the use of filters and with the wide spread of easy-to-use video processing and manipulation, we often question ourselves how real are these posted images and videos? And if anyone can alter and re-distribute news through TV, radio, social media networks (e.g., YouTube, Facebook, Twitter, etc.). Then, we should no longer trust this news or believe everything we see on the internet.

With the evolution of computer vision and deep learning technology, a new emerging technique has appeared known as “Deepfake” which may allow anyone to make fake videos, images and even can manipulate the voices in a realistic way. Although, it seems an interesting technique to make fake videos or images but it could spread misinformation via internet. Deepfake contents could be dangerous for individuals as well as for our communities, organizations, countries, religions, etc. They could cause disputes, spread hatred or destroy lives. As Deepfake content creation involve a high-level expertise with combination of several algorithms of deep learning, it seems almost real and genuine and difficult to differentiate.

In this chapter, we took a look at Deepfake videos. First, we explained the rise of deepfakes and its consequences. Then, we highlighted the motivation of our research and how to prevent its spread. Lastly, we presented Deepfake detection methods existing in literature.

2. Deepfake Videos

Even if you have not heard the term ***deepfake*** before, there are high chances that you have unknowingly came across dozens of online deepfake videos. Deepfake (or "AI faceswap") combines 'deep' and 'fake', which refers to the set of deep learning-based facial forgery techniques where they swap a face of a person with another, using Deep Neural Network-based (DNN) methods, like Autoencoders, Generative Adversarial Networks (GANs), and Variational Autoencoders (VAEs). These methods are capable of producing accurate and persuasive fake videos which cannot be easily distinguished from real videos by human eye. Deepfakes raise some serious ethical issues [1], because they can be misused in many ways, like fake news, elections manipulations, revenge porn, blackmail someone,

hoaxes, financial fraud and automated cyberattacks or even fake terrorism events. the possibilities are limitless [2]. Examples of deepfakes are illustrated in Figures 1.1 and 1.2.

Fig 1.1 Deepfake Video of tom cruise [3]

Fig 1.2 Face-Swap based deepfake [4]

3. The quick rise of deepfakes

Deepfakes are not new, over the past few years, deepfakes have gone viral and a lot of deepfake videos have been crafted and uploaded to the internet [2]. The things that made deepfake possible and more creative and easier and convincing are GANs and autoencoders. However, they have not been seen as a real threat until recently, deepfake development ignited within academic institutions (see Figures 1.3 and 1.4 for illustration), online communities, and in the media and entertainment industries, they are also having a significant impact in pornography film industry, where 96% of deepfake videos have pornographic content. They also found that the top four deepfake pornography websites received over 134 million views. This large audience shows that there is a demand for websites that produce and host deepfake pornography, a trend that will continue to develop unless immediate action is taken [5].

Between December 2018 and October 2019, online deepfake videos number had risen by 84% [6]. Deep Trace found that in 2019 there were 14,678 deepfake videos online, 96% of which are pornographic, it also noted that deepfake creation communities are growing. Forum-based websites like GitHub, 4chan, 8chan, and other are sharing open-source deepfake code [7].

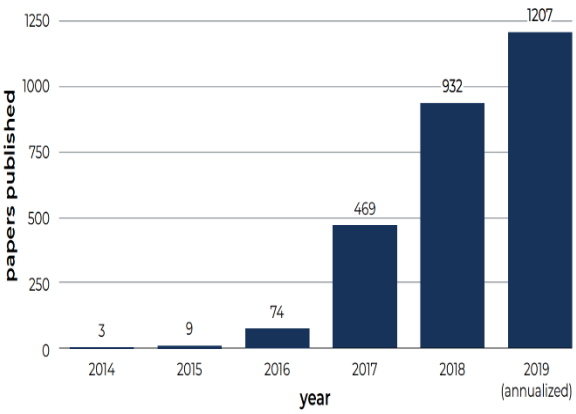


Fig 1.3 Papers mentioning GANs per year [5]

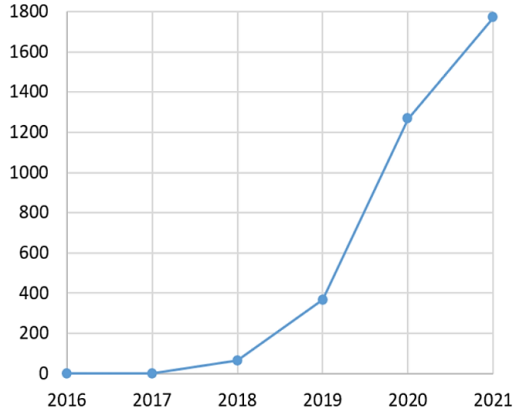


Fig 1.4 Number of papers related to deepfakes in years from 2016 to the end of 2021 [8]

It is surprisingly simple to make realistic deepfakes, similarly to how it is possible to make good videos without requiring the mastery of great skill. Many deepfake applications have been available for few years. They can be used by both novice and professional users. [9].

The technology that makes these falsified videos is continuously improving. While producers once needed hundreds of images to overwrite one face with another. FakeApp is the first method that has been used widely for deepfake creation [8]. Nowadays various easy-to-use deepfake creation applications exist like FaceApp, deepface lab, faceswap, Zao, etc. It is now becoming simple to create a realistic deepfake video with only a single image in a couple of seconds [10].

4. Consequences of the Spread of Deepfakes

Deepfake technology has a huge range of applications which could be used both positively or negatively. However, most of the time it is used for malicious purposes. The unethical uses of deepfake technology have harmful consequences in our society. People regularly using social media are at potential risk of being used to produce Deepfake. However, proper use of this technology could bring many positive results. Below are few examples of both negative and positive deepfake use cases:

4.1. Negative use cases

Deepfakes negative use cases can be risky on the mental safety of users that can be distorted by the blackmails or by the shame they bring to innocent users who get their pictures manipulated in a non-ethical way. It can have disastrous impact on finance if they are applied

on businessmen to fraud them. It may also cause political disputes and spread fear and panic if used by terrorists or criminal organizations. As *Joseph Carson, Chief Security Scientist at Thycotic has said: "Deepfakes are becoming one of the biggest threats to global stability, in terms of fake news as well as serious cyber risks."*.

Deepfakes have elicited growing concern for their usage such as:

- **Pornography:** Most common use of deepfake technology is to make pornography almost 96% of its use cases [7].
- **Politics:** Another malicious use of deepfake is to exploit world leaders and politician by making fake videos of them and sometimes it could cause great risk for world peace. "The idea of deepfakes alone is enough to destabilize political processes" says Patrini [11].
- **Business:** Deepfakes pose numerous financial threats to businesses, they are becoming difficult issue to combat. Hackers and deepfake makers may damage companies' or employees' reputations at risk for fun or retribution.
- **Blackmail:** Deepfakes can be used to generate blackmail materials that falsely incriminate a victim.
- There is also a vast use of Deepfake in art, film industry and in social media.

4.2. Positive use cases of deepfakes

Although most of the time this technology is used for malicious work with bad intention still it has some positive uses also in several sectors. The deepfake creation is no longer limited to experts, it now become much easier and more accessible to anyone. Deepfake technology can be applied to both advertising and business purposes. Technologists now are using the deepfake to make copy of famous artwork such as creating video of famous Monalisa artwork using their images. Deepfake technology can save huge money and time of film industry by using the capabilities of deepfake technology for editing videos rather than re-filming them, one famous example of deepfake use is when David Beckham run a campaign against malaria in 9 different languages [8]. Also, bringing Peter Cushing back to screen after his death in "Rogue One: A Star Wars Story in 2016", using deepfakes and intricate expensive pipelines with face-mounted cameras [12].

5. Motivation and Project Background

Deepfake has become a growing concern among consumers and organizations, as they are exploited by criminals to carry out social engineering attacks, as well as cybercrime trends, besides spreading misinformation and fraud scams. The cost of a deepfake scam was estimated to exceed $250 million in 2020, and this form of technology is still in its early stages [13]. Few of the deep fake use cases that had scandalous and dangerous effects are listed below:

- The first reported deepfake assault occurred in 2019, when fraudsters utilized deepfake audio to imitate a CEO's voice and scammed his company into sending US$243,000 to the parent company in Germany [13].
- A suspicious video of the long-missing Gabon president supposedly delivering a New Year's address accelerated an unsuccessful military coup in an already unstable country [14].
- A chief executive at a UK energy company wired $220,000 to a Hungarian supplier after receiving the instructions from his boss where these instructions were audio deepfake [13].
- In 2020, a Tech firm employee received a strange voicemail from someone who sounded like the organization's CEO in an unsuccessful audio deepfake attempt. The message was for "urgent assistance to complete an urgent business agreement". Luckily, the employee followed his instincts and reported it to the company's legal department [13].
- Deepfakes of North Korean leader Kim Jong-un and Russian president Vladimir Putin these deepfakes were meant to air publicly as commercials to relay the notion that interference by these leaders in US elections would be detrimental to the United States' democracy[15].
- Deepfakes were created for almost every world leader, including Barack Obama, Donald Trump, Nency Pelosi, and Angela Merkel [11].
- In early 2020, Police in the United Arab Emirates are investigating a case where AI was allegedly used to clone a company director's voice and steal $35 million in a massive heist [16].
- A video of Facebook CEO Mark Zuckerberg appearing to talk about how Facebook 'controls the future [17].

- For the first time during a war a deepfake of Ukrainian President Volodymyr Zelensky ordering his people to "surrender" was spread on multiple websites and social media networks such as: Ukraine 24, Facebook, Youtube, Telegram, and VKontakte [5].

Top deepfake artist Hao Li believes that in the next six months, deepfake videos will be completely undetectable [18].

6. Deepfake Prevention Efforts

Researchers and tech giants are focusing on developing tools for exposing fakes, but malicious deepfake makers are getting savvier. Some US states have enacted legislation to criminalize fake media used to deceive, defraud, or destabilize the public. However, as they say “prevention is the best cure”, people need to be cautious when being in the cyberworld, especially when using social networks and when handling financial transactions online, they should:

- Learn about deepfakes and its risks.
- Be cautious when they notice that the video has different skin tones from a frame to another, abnormal movements in the hand gestures and facial muscles, robotic tones or unsynchronized speech. These digital artifacts may indicate that the video is potentially a deep fake [17].
- When receiving business calls or emails, even when reading news always follow the rule of "trust but verify" and not the rule “believe what you see” to avoid falling in the traps of deepfake creators.
- **Social media platforms:** such as Twitter and Facebook are taking active measures to handle synthetic and manipulated media on their platform. In order to prevent disinformation from spreading [17].
- **advocate**: At the moment there is little legal consequence for producing, hosting and sharing deepfakes in various countries such as the US and China [4].
- **Detection:** innumerous academic research is being conducted on deepfake detection techniques from images, videos and audios. In order to speed up the development of new ways to detect deepfake videos, various corporations like AWS, Facebook, Microsoft, and Amazon, as well as academics from the Partnership on AI's Media Integrity Steering Committee, collaborated to create the Deepfake Detection

Challenge (DFDC), offering $1,000,000 prize to the most accurate deepfake detection models on the DFDC [19].

7. Deepfake Detection Using Deep Learning

The majority of deepfake generators leave unique fingerprints in deepfakes. These fingerprints in deepfake videos are either in form of spatial inconsistencies, which occur within individual frames of the video, or temporal inconsistencies, which occur across the sequence of frames of the video. Face area incompatibilities with the background of the video frames, resolution differences, and partially displayed organs and skin texturing are examples of spatial inconsistencies. The majority of deepfake generators fail to render facial features like eye blinking and teeth. The deepfake generation occasionally replaces white strips for teeth that are visible to the human eye (see **Fig**1.5). Temporal inconsistencies include abnormal eye blinking, head poses, facial movements, and variations in luminance in the frame sequence across the video [20].

The common steps in the deepfake detection process are illustrated in (**Fig** 1.6). Normally, deepfake detection is treated as a binary classification (Fake/Real). To train classification models, this type of technique necessitates a large database of real and fake videos. While some datasets are available, the quantity of fake videos is still limited when compared to other datasets used in deep learning. The dataset will then be preprocessed based on the study's goals. This preprocessed data will be divided into three sets: training, validation (optional), and testing. Each of these sets will be used as input to the proposed model, which will train on this dataset and then make predictions on the testing set to detect if these videos are fake or real.

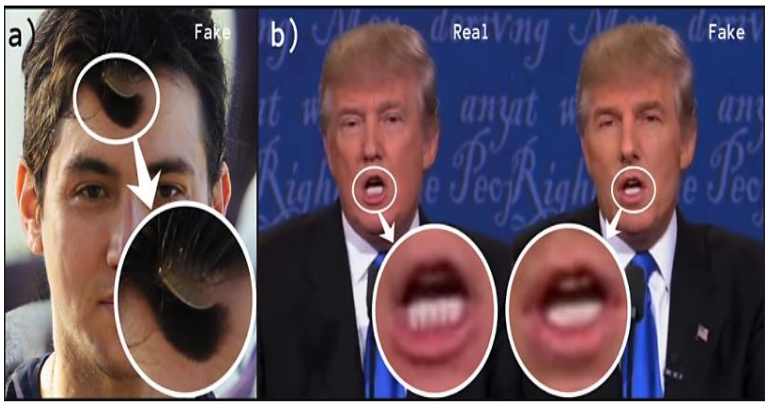


Fig 1.5 Spatial inconsistencies of Deepfakes. **a)**: incomplete rendering of hair. **b):** use of white strip instead of individual teeth [20]

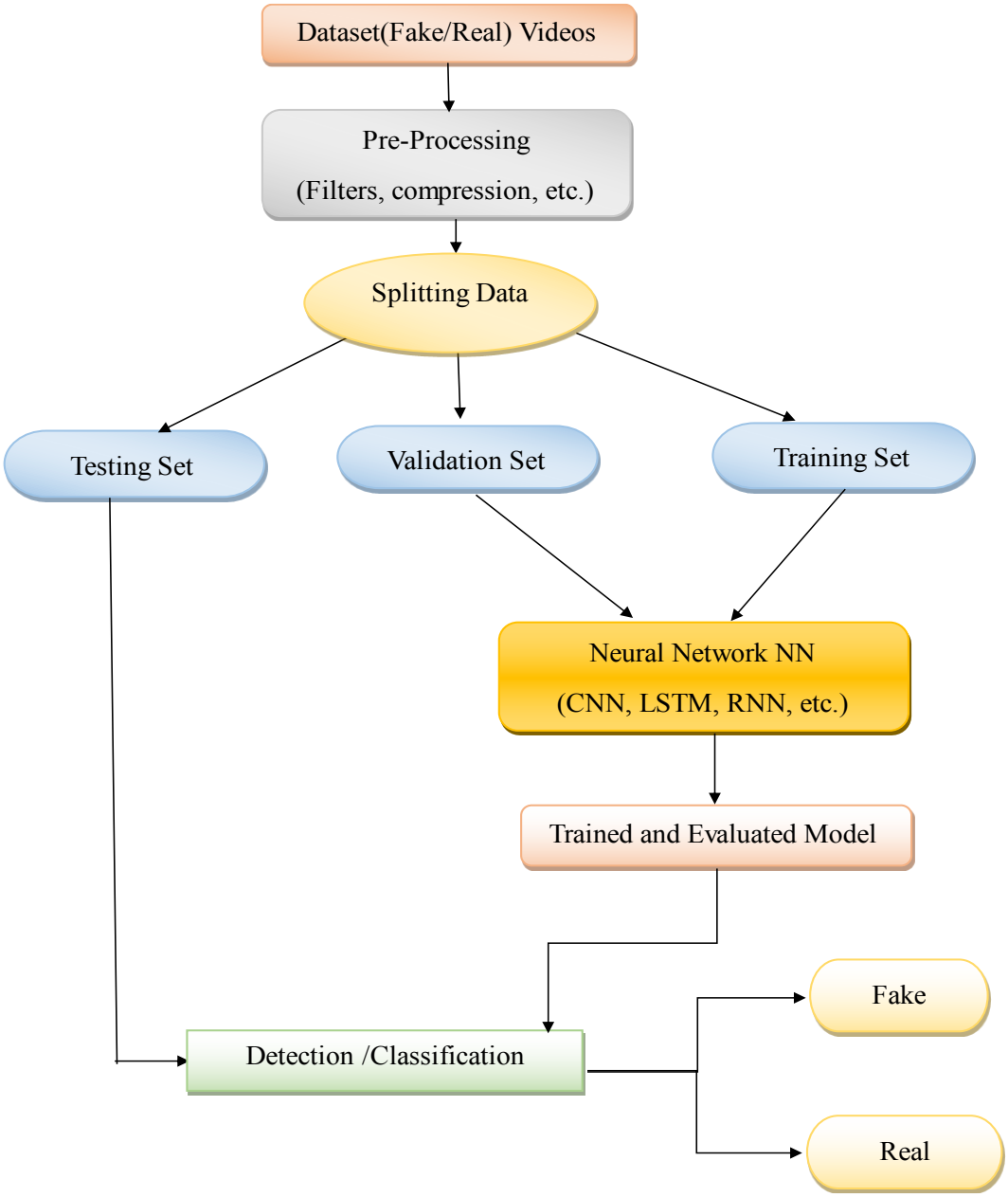


Fig 1.6: Deepfake detection process using deep learning approach

8. Deepfake Detection Models

The mounting concerns over the negative impacts of deepfakes have spawned an increasing interest in deepfake detection. Over these past few years, there have been numerous new detection methods of deepfakes. We mention few of them briefly. In the work presented by Xin Yang et al. [21] they used a new Support Vector Machines (SVM) based method to detect deepfake videos by comparing the face landmarks between the real images and fake images. Darius Afchar, et al. [22] used deep learning algorithms to detect deepfake videos and focusing on the compression artifacts for images in the videos. Their approach is

based on two Convolutional Neural Networks using Meso-4 and MesoInception-4 models to analyze intrinsic characteristics of images. David Guera and Edward J. Delp [23] integrated two deep learning algorithms to detect fake videos. Each video frame is analyzed as it passes through two stages of scrutiny and analysis. The first algorithm (CNN) that is used to retrieve features from frames of fake videos, the second algorithm is (RNN) and (LSTM) which is used to detect fake videos relying on the inconsistencies between the frames and the time discrepancies that appear after the creation of deepfake video they classify fake and real videos. Yuezun Li & Siwei Lyu [24]. presented a model based on comparing the face and the sides of the face in both real and fabricated image frames from the video. They used four models to check their deepfakes which are VGG16, ResNet50, ResNet101 and ResNet152. Mousa Tayseer et al [25]. proposed the DFT-MF model which was built to detect deepfake videos by using the mouth as a biological signal and CNN as the deep learning model. It detects discrepancies between lip movement and voice synchronization in both videos [25]. DeepFake-o-meter platform is another deepfake detection web application that integrates more than 10 state-of-the-art detection methods and it also allows researchers to incorporate their detection method into the platform [26]. Table 1.1 recaps some of the well-known deepfake detection methods by illustrating their used deep learning models, their accuracy as well as their functioning principle (method).

Table 1.1 different techniques used for the Deepfake detection.

Study	Year	Method	Classifiers	Accuracy					Dataset
				AUC	Recall	Acc	EER	Precision	
(Li, Chang, and Lyu) [27]	2018	Physiological Features	LRCN	99.0%	-	-	-	-	UADFV
(Agarwal and Farid) [27]	2019	Physiological Features	SVM	96.3%	-	-	-	-	Own (FaceSwap, HQ)
(Matern, Riess, and Stamminger) [27]	2019	Visual Features	Logistic Regression MLP	85.1%	-	-	-	-	Own AUC
				78.0%	-	-	-	-	FF++ / DFD
				66.2%	-	-	-	-	DFDC Preview
				55.1%	-	-	-	-	Celeb-DF
(Sabir et al.) [27]	2019	Image + Temporal Features	CNN + RNN	96.9%	-	-	-	-	FF++ (DeepFake, LQ)
				96.3%	-	-	-	-	FF++ (FaceSwap, LQ)
(Nguyen, Yamagishi, and Echizen) [27]	2019	Deep Learning Features	Capsule Networks	61.3%	-	-	-	-	UADFV
				96.6%	-	-	-	-	FF++ / DFD
				53.3%	-	-	-	-	DFDC Preview
				57.5%	-	-	-	-	Celeb-DF
(Dolhansky et al.) [27]	2019	Deep Learning Features	CNN		8.4%	-	-	93.0%	DFDC Preview
(Dang et al.) [27]	2020	Deep Learning Features	CNN + Attention Mechanism	99.4%	-	-	3.1%	-	DFFD
(Badhrinarayan Malolan et al.) [28]	2020	Visual Interpretability Methods	Xceptionnet (CNN) + LRP and LIME	-	-	90.17%	-	-	FF++
(Jung, Kim, and Kim) [27]	2020	Physiological Features	Distance	-	-	87.5 %	-	-	Own
(Qi et al.) [27]	2020	Physiological Features	CNN + Attention Mechanism	-	-	100.0%	-	-	FF++ (FaceSwap)
				-	-	100.0%	-	-	FF++ (DeepFake)
				-	-	64.1 %	-	-	DFDC Preview
(Ciftci, Demir, and Yin) [27]	2020	Physiological Features	SVM/CNN	-	-	94.9 %	-	-	FF++ (DeepFakes)
				-	-	91.5 %	-	-	Celeb-DF
(javier.hernandezo et al.) [27]	2020	Physiological Features	CAN	99.9%	-	-	-	-	Celeb-DF v2
				98.2%	-	-	-	-	DFDC Preview
(Hanqing Zhao et al.) [29]	2021	Multi-attentional	LSTM+Xception	99.80%	-	-	-	-	FF++
				67.44%	-	-	-	-	Celeb-DF
Pan Xu et al. [28]	2021	Cross-domain fusion	Meso-net	-	-	96 %	-	-	FF++

9. Conclusion

In this chapter we took a close look at deepfakes and their creation, then we discussed their quick growth and impact on society in these past few years, after that, we mentioned the consequences of their spread in both positive and negative use cases. Along with a few examples that mainly motivated us for this research. Afterwards, we mentioned some prevention efforts proposed by researchers and tech giants for people to take their cautious against deepfakes.

Lastly, we ended this chapter with some deepfake detection models using deep learning existing in literature.

In the next chapter we will discuss our proposed deepfake detection model, and the process we followed developing our deepfake detector, from dataset collection, to preprocessing and data splitting (train/test).

CHAPTER II

DEEP LEARNING BASED DEEPFAKES DETECTION SYSTEM

1. Introduction

In the previous chapter we discussed the growth of deepfakes, its threat to our society, and some potential prevention strategies. In this chapter, we illustrate our deepfake detection approach where we trained and tested our detector on preprocessed images.

We used FaceForensics++ dataset for this work. Our developed deepfake detection model relies on transfer learning. It is trained first to classify the video as fake or real (binary classification), then, it is trained to know the method used to create the fakes among our three selected which are Face2Face, FaceSwap, or DeepFakes (multiclass classification). This chapter illustrates the process of developing the deepfake detector, it will be organized as follows: Section 2 explains the video dataset collection process. Section 3 illustrates the preprocessing of this dataset which consists of frame extraction, then face cropping and dataset organization (train/test). Section 4 depicts the training process.

2. Dataset

Deep neural networks are data hungry and for deepfake detection, we need a huge amount of data. There is plenty of deepfake datasets, but not all of them are publicly and freely available, some of them require fees payment and others are private requiring authors approval. Table 2.1 illustrates some of the deepfake available datasets. The deepfake videos dataset are created by passing original videos to various deepfake generation methods [30].

Table 2.1: Deepfake datasets

Datasets	Type	Size	Generation method	description
DFDC [31].	Videos	10000 fakes 20000 Original(real)	Various deepfake techniques	Dataset, released by Facebook [31].
Wild-Deepfake [30].	Images	1180099	Various deepfake techniques	Dataset consists of images with 7314 face sequences extracted from 707 deepfake and real videos [30].
DeepFake-TIMIT [31].	Videos	640	Faceswap GAN	Includes deepfakes created from Vid-TIMIT [29].
DFD [31].	Videos	3068 fakes 363 original	Various deepfake techniques	Released by Google and Jigsaw. It contains videos of 28 consented persons of diverse genders, ages, and ethnic groupings [31].
Celeb-DF [31].	Videos	5639 fakes 590 original	Various deepfake techniques	Original videos were downloaded from YouTube with subjects ranging in age, ethnicity, and gender [31].

Most of the above-mentioned datasets either do not mention the deepfake generation method or have the videos all mixed together. Since we aim to develop a model that can differentiate between deepfake generation methods, obtaining a well-constructed and classified dataset a

fundamental need. Therefore, we did not use these datasets. Instead, we used **FaceForensics++ dataset.** This dataset consists of 1000 original videos and its manipulated videos created using 4 deep learning face manipulation methods: Face2Face, Deepfakes, Faceswap and NeuralTextures. It is well structured and its videos are well classified and not randomly mixed. Noting that Deepfakes and Faceswap methods belong to the *identity swap* category like illustrated in **Fig 2.1** in section (A), Face2Face and NeuralTextures methods belong to the *facial expression manipulation* category section (B) [32].

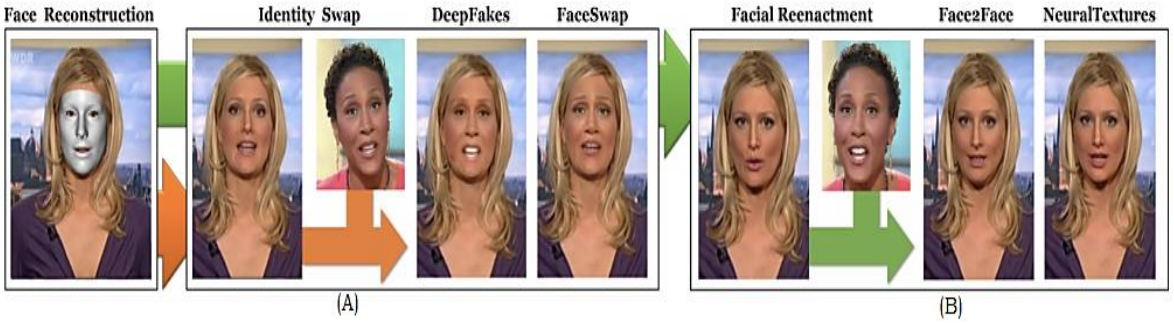


Fig 2.1-different types of FF++ methods [30]

The choice of using FaceForensics++ dataset was not only motivated by its organization but also because it is free and publicly available. Moreover, its use is well documented. Additionally, it provides different video quality options for data downloading depending on time and bandwidth constraints, the varying qualities are important for deepfake detection, i.e., it is easier to distinguish a high-resolution deep fake than a lower resolution one [33]. Among the four existing methods we choose to use deepfake videos created by FaceSwap, Face2Face, DeepFakes which are explained below:

- **FaceSwap:** It is a graphics-based approach, which transfers the face of the source image to the target video. This is done using extraction of facial landmarks from the source image and fits the 3D model using blend shapes and projects on the target image. Image and color correction are applied to blend the morphed face onto the source image to create a realistic looking fake image. This process is repeated for every frame to generate a video out of it [30]. See **Fig 2.2** for illustration of fakes created by FaceSwap method.

Fig 2.2-Example of FaceSwap method

- **Face2Face:** The Face2Face method belongs to the facial reenactment category which aims at transferring facial expressions of the source image to the target image. In this approach, they use the 3D model using blend shape coefficients for transferring facial expressions from one image to another. They use the first frame for temporary face identity and the other frame key points for creating fake expressions [32]. See **Fig 2.3** for illustration of fakes created by Face2Face method.

Fig 2.3-Example of Face2Face method

- **DeepFakes:** This method is based on the training of two autoencoders using a shared encoder to rebuild training images of the source and target faces, respectively. Cropping and aligning the photos is done with the help of a face detector. The trained encoder and decoder of the source face are applied to the target face to create a false image. Using Poisson image editing, the autoencoder output is blended with the remainder of the image [30]. See **Fig 2.4** for illustration of fakes created by DeepFakes method.

Fig 2.4-Example of DeepFakes method

Since we have a limited equipment with moderate computational power at hand, we did not use all the videos provided by FaceForensics++ dataset. Instead, we downloaded 480 videos in total, containing 240 Real videos and 240 Fake videos (80 from Face2Face,80 from DeepFakes and 80 from FaceSwap). To obtain the FaceForensics++ dataset, we fill a google form to request it from its makers, the form is available in their official site [34]. Once they accepted our request, they sent us a link to their download script and provided us with all the information we needed to download our desired data. The following scripts were used to download our dataset:

```
python df_detection.py F:/dataset/original -d original -t videos -c c23 --num_videos 240
python df_detection.py F:/dataset/ofake/face2face -d Face2Face -t videos -c c23 --num_videos 80
python df_detection.py F:/dataset/ofake/faceswap -d FaceSwap -t videos -c c23 --num_videos 80
python df_detection.py F:/dataset/ofake/deepfakes -d DeepFakes -t videos -c c23 --num_videos 80
```

where:

- "**df_detection.py**" is the name of the received script file to download the dataset
- "**F:/dataset/…**" is the output path.
- **--num_videos 80** is the number of videos to download (dataset size).

After downloading the data. We passed to the preprocessing phase which will be explained in the next section.

3. Dataset Preprocessing

Preprocessing the dataset is critical for achieving greater performance and accuracy. Therefore, we preprocessed our dataset by extracting the frames from videos, then, extracting our Region of Interest (ROI) which is the faces (face cropping). The explanation of each phase is giving below:

3.1 Frames Extraction

A video is a sequence of frames. Therefore, the first step is to split the video into frames. The video capture function provided by the OpenCV Python package was used for this purpose. Since the similarity between two adjacent frames is too high, hence, using them will reduce the training efficiency and may cause the overfitting issue which occurs when the model fails to generalize and perform well in the case of unseen data scenarios, defeating the model's purpose. So, for this work, an image is extracted from each video every 1 second, and since most of our videos are not longer than 30-40 seconds, so we extracted 30 frames per each video as illustrated in Table 2.2.

3.2 Face Cropping

A frame in a video does not contain just a face. Indeed, the body parts of the person and the background area of the image comprise most of the video frame. These features can negatively impact the training of the model, and because we are working on deepfake detection, we need to focus on the face area, which is the region of interest for our deep learning model. Thus, we need to capture the face in the image as input and label them. The *cascade classifier* provided by OpenCV was used for this purpose as it crops accurately the face area. There were several complications with non-face selection. For example, it occasionally recognizes earrings instead of the face or some background item with a face form, so we had to manually verify and eliminate the wrongly cropped data. The facial images have to be resized consistently to (224, 224) because this is the image size required for most transfer learning models. In the left part of **Fig 2.5** are illustrations of correctly cropped faces and in the right part are the mis-cropped images.

Fig 2.5-non-face selection issues

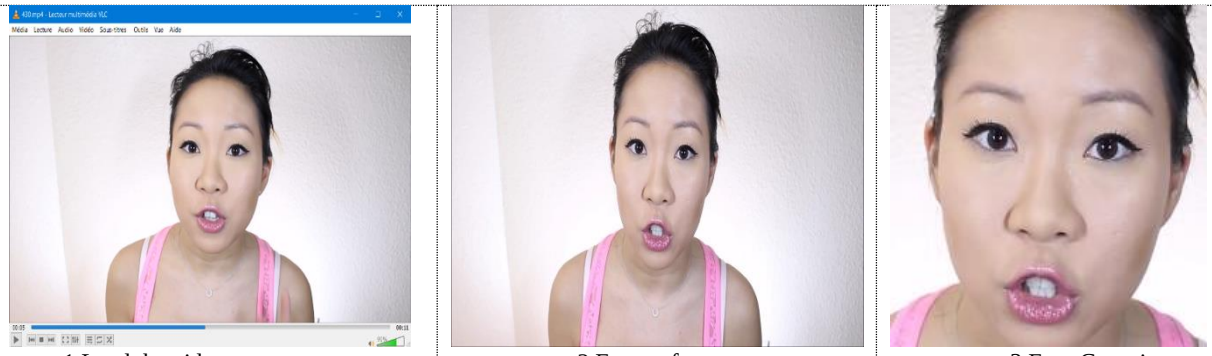
3.3 Data divisions (train/test):

Since we are making two models, we need two different datasets, each organized into training and testing sets for each model. The training set is seeded to the model to help it learn to distinguish between fake and real data, while the testing set is used to evaluate the model's predictions.

- **For our first model:** We divide our data randomly into training and testing sets, and then use the following divisions by 62.5% percent for training and 37.5% percent for testing.

Our training dataset have 9000 preprocessed images half of which are real images. For the testing dataset we have 5400 preprocessed images, 2700 among them are real as shown in Table 2.2.

- **For our second model:** We have also randomly divided our data into training and testing sets, with 62.5% percent for training and 37.5% percent for testing. The training dataset was divided into three different classes named after their deepfake generator method (Face2Face, DeeFakes and FaceSwap). Each class contained 1500 preprocessed image extracted from 50 deepfakes. The training dataset have 4500 images in total. The testing dataset contained 2700 images in total organized on 900 preprocessed images for each class extracted from 30 deepfake video, the dataset details are depicted in Table 2.2.



1.Load the vide 2.Extract frames 3.Face Cropping

Fig 2.6- Preprocessing steps

Table 2.2 total number of our dataset divisions

Method		First Model (480 video)		Second Model (240 video)	
		Train	**Test**	**Train**	**Test**
Real	Videos	150	90	-	-
	Images	4500	2700		
FaceSwap	Videos	50	30	50	30
	Images	1500	900	1500	900
Face2Face	Videos	50	30	50	30
	Images	1500	900	1500	900
DeepFakes	Videos	50	30	50	30
	Images	1500	900	1500	900
Total	Videos	300	180	150	90
	Images	9000	5400	4500	2700

4. Model Construction

After data preprocessing, scaled and resized image data are ready for model training and testing. Due to the limited data size and equipment computational power at hand. The use of pretrained models and relying on transfer learning technique was our ideal choice. Transfer

learning has various advantages like saving training time and presenting better performance of neural networks. Our selected pre-trained model is CNN-based frame feature extraction and classification model. To select the most suitable pretrained model to use, we surveyed the literature about existing models. We resumed the used models in section 4.3 and then we compared between a few of them in section 4.4 to see which one is the best to use for our case. Lastly, after analyzing the results of the models, we selected ResNet50-V2 and started the training, validation and refining loop until we achieved a good accuracy result. We explain the ResNet50-V2 in subsection 4.4 and the obtained results will be illustrated in the next chapter.

4.1 Convolutional Neural Networks (CNN)

CNNs have numerous hidden layers (multi-building blocks), as well as an input and output layer **(Fig 2.7).** The CNN hidden layers are usually made up of convolutional layers where the features are extracted. The Pooling Layer mainly consists of sub-sampling the feature maps. And finally, the fully connected layer, where the classification is done [35].

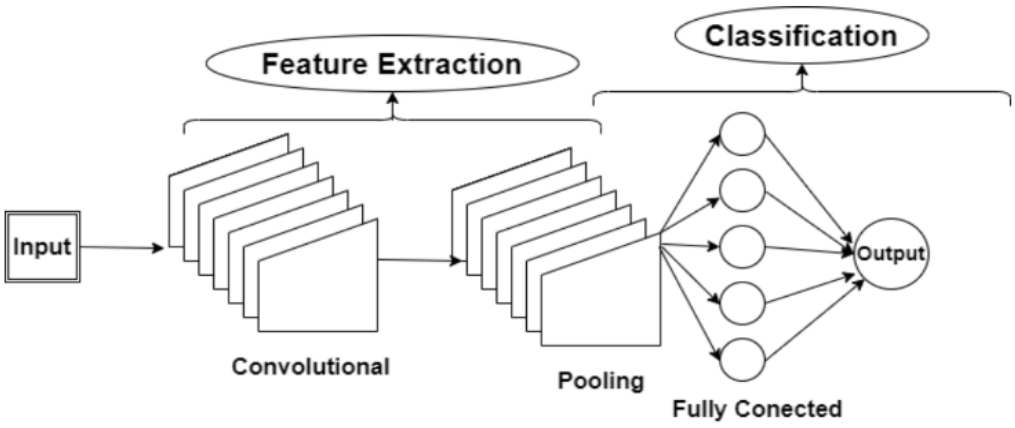


Fig 2.7 CNN Architecture [35]

4.2 Transfer learning and fine-tuning

The reuse of a pre-trained model on a new problem is referred to as transfer learning. it can train deep neural networks with very minimal input data. This is particularly beneficial in data science because most real-world situations do not require millions of labeled data points to train complicated models [36]. A pretrained model needs to have some of its specific layers retrained while leaving the others unmodified in order to be appropriately adapted to a new task. Typically, this modification called fine-tuning [37]. Currently, there is no general rule or recipe to follow in order to determine which layers to fine-tune or which hyper-

parameter settings to use. Most of the decisions are based on previous experiences of dealing with such problem.

4.3 Deep Learning Models

Keras applications are deep learning models with pre-trained weights. These models can be used for feature extraction, prediction, and fine-tuning. Table 2.3 resumes some of Keras pretrained models, a brief description of their functioning as well as their depth and top-1 accuracy. Noting that **top-1 accuracy** refers to the model's performance on the ImageNet validation dataset while the **depth** refers to the topological depth of the network including activation layers, batch normalization layers etc. [38]. ImageNet dataset consists of a about 1.2 million images for training, 50,000 for validation and 100,000 for testing, belonging to 1000 categories including cats, dogs, humans, objects, etc. [39].

Table 2.3 Keras pretrained models [37]

Model	Description	Top-1 accuracy	Depth
Residual Networks (ResNet)	▪ ResNet has hundreds of layers ▪ It achieves excellent results. ▪ It has various architectures available with different numbers of layers, such as: ResNet-18, ResNet-50, ResNet-101, ResNet-151, etc. [40-41]	74.9% - 78.0%	103 - 311
EfficientNet	▪ It offers a variety of networks (B0 to B7) ▪ It outperform state-of-the-art performance with up to 10x greater efficiency (smaller and faster) [42].	77.1% - 85.7%	132 - 438
InceptionV3	▪ It deploys more inception modules than Inception-v2. ▪ It employs factorized 7 x 7 convolutions, label Smoothing, and an extra auxiliary classifier to transmit label information to the network [42].	77.9%	189
Visual Geometry Group (VGG)	▪ It is a multilayer deep CNN architecture. ▪ It has two architectures **VGG-16** and **VGG-19** which have 16 and 19 convolutional layers, respectively [43].	71.3%	16-19
Extreme Inception (Xception)	▪ It applies a 1x1 convolution. ▪ Each data slice is filtered separately using unique filters. ▪ Each slice may be handled independently [44].	79.0%	81
MobileNets	▪ They are efficient neural network topologies. ▪ They are well-suited for mobile and embedded vision-based applications. ▪ They decompose standard convolutions into 1*1 pointwise convolution [31]. ▪ It has three architectures MobileNet, MobileNetV2, and MobileNetV3	71.3%-71.4%	55-105

4.4 Comparison of Network Models

Since we are working with transfer learning in this study, for our first model, we tried seven pretrained CNN models among the above-mentioned models in Table 2.3 which are: VGG16, Inception-v3, Inception-ResNet-v2, Xception, MobileNet-V2, MobileNet-V3 and

ResNet50-V2, all of which were imported from the Keras library and have been applied in our experiment by the first model on the same dataset depicted in Table 2.2, we explained more details about it in the last chapter. The results are shown in Table 2.4.

Table 2.4 Comparison between official accuracy of Keras models and our obtained accuracy

Network	Official documented accuracy	Obtained Accuracy		
		Average	Training	Testing
ResNet50V2	76.0 %	80.0%	100%	60 %
VGG16	71.3 %	72.7%	75.0 %	70.3 %
Xception	79.0 %	67.0%	75.0 %	59.0 %
INceptionV3	77.9 %	47.5%	44.9 %	50.0 %
INceptionResNetV2	80.3 %	65.4%	69.9 %	60.8 %
MobileNetV2	71.3 %	42.5%	34.9 %	50.0 %
MobileNetV3	73.8 %	55.0%	60.0 %	50.0 %

As shown in Table 2.4, the average obtained accuracy of ResNet50-V2 network model has the highest accuracy with 80%, so ResNet50-V2 is the pre-trained CNN model that we decided to continue using in our work. In the next section, we briefly explain the functioning of this model.

4.5 ResNet50-V2 architecture

The ResNet50V2 model accepts 224×224×3pixel input images. It first performs the convolution (7×7) and max-pooling (3×3) operations. There are four stages in this model, as well as some residual and identity blocks. For instance, in Stage I, the network provides three residual blocks (each with three layers) that conduct convolution operations with kernel sizes of 64 and 128. The size (width and height) of the input images are cut in half as it progresses through the phases, and the width of the image channel is doubled. The key difference between the ResNetV2 and ResNetV1 versions is that the former one does the convolution operation later, that is, after performing the batch normalization and ReLU activation. The ResNetV1 performs batch normalization and ReLU activation after convolution [45].

Name of the Layer	Size of the Output	Resnet50V2
Conv 1 (Stage I)	112 × 112	7 × 7 convolution with a stride of 2
	112 × 112	3 × 3 max pooling with a stride of 2
Conv 2 (Stage II)	56 × 56	[1 × 1,64; 3 × 3,64 and 1 × 1,256] × 3
Conv 3 (Stage III)	28 × 28	[1 × 1,128; 3 × 3,128 and 1 × 1,512] × 4
Conv 4 (Stage IV)	14 × 14	[1 × 1,256; 3 × 3,256 and 1 × 1,1024] × 6
Conv 5 (Stage V)	7 × 7	[1 × 1,512; 3 × 3,512 and 1 × 1,2048] × 3
Classification	1 × 1	Global average pooling [7 × 7] with 1000 fully connected Softmax layers

Fig 2.8 Resnet50V2 Layers Architecture [45]

5. Conclusion

In this chapter, we presented our chosen dataset FaceForensics++ and its methods, Then, we preprocessed this data to make it compatible with our models. We used seven pre-trained models to train our dataset, and we chose ResNet50V2 for the rest of our work.

In the next chapter, we will resume the detailed results obtained by our models in term of accuracy. We will illustrate these results through plotting the model loss, confusion matrix and accuracy curve. Then, we will depict the front-end which is a web application allowing users to test whether the uploaded videos are fake or real and if they are fake, with which method were they created.

CHAPTER III
IMPLEMENTATION AND ANALYSIS

1. Introduction

After explaining our aim which is detecting deepfakes along with their generators which were in our case Face2Face, FaceSwap, or DeepFakes. We also demonstrated our conducted experiences in order to select the best classifier suiting our use case.

In this chapter, we present an open platform called DeepFake Detection. This platform is made up of two parts: a front-end and a back-end. The front-end is a user-interactive web application, while the back-end is our ResNet50V2 models trained to detect deepfakes which are able to first distinguish between fake and real videos. Then, classify the method used to create the fake ones.

2. Development environment

The hardware and software tools used in the development of our Web Application are resumed in Table 3.1. We deployed Tensorflow on our GPU because it improvises the performance of this deep learning framework to reach new heights and peaks and it works only with NVIDIA, we had to install CuDNN and CUDA toolkit [46].

Table 3.1 Development environment

Hardware Tools		**Software Tools**	
CPU	Intel(R) Core i7-6700HQ 2.60GHz	**Operating System**	Windows10 64-bit
		Computer Vision	OpenCV library
RAM	16.0 Gb	**Deep Learning**	Tensorflow, Keras
		Implementation Editors	Jupyter Notebook (back-end) Visual Studio Code (front-end)
GPU	NVIDIA GEFORCE-GTX 960	**Implementation Environment**	Miniconda environment
		GUI	**Flask:** Web Server Gateway Interface (WSGI) for our Web Application

3. Deepfake Detection Design

The architecture of our web application is made up of two parts: the back-end and the front-end. Data collection, preprocessing, model creation, and model performance evaluation are all described in the back-end of this project. The front-end is the user interface which is a web application allowing users to interact with our deep learning models. The overall view of our web application architecture is illustrated in **Fig 3.1.**

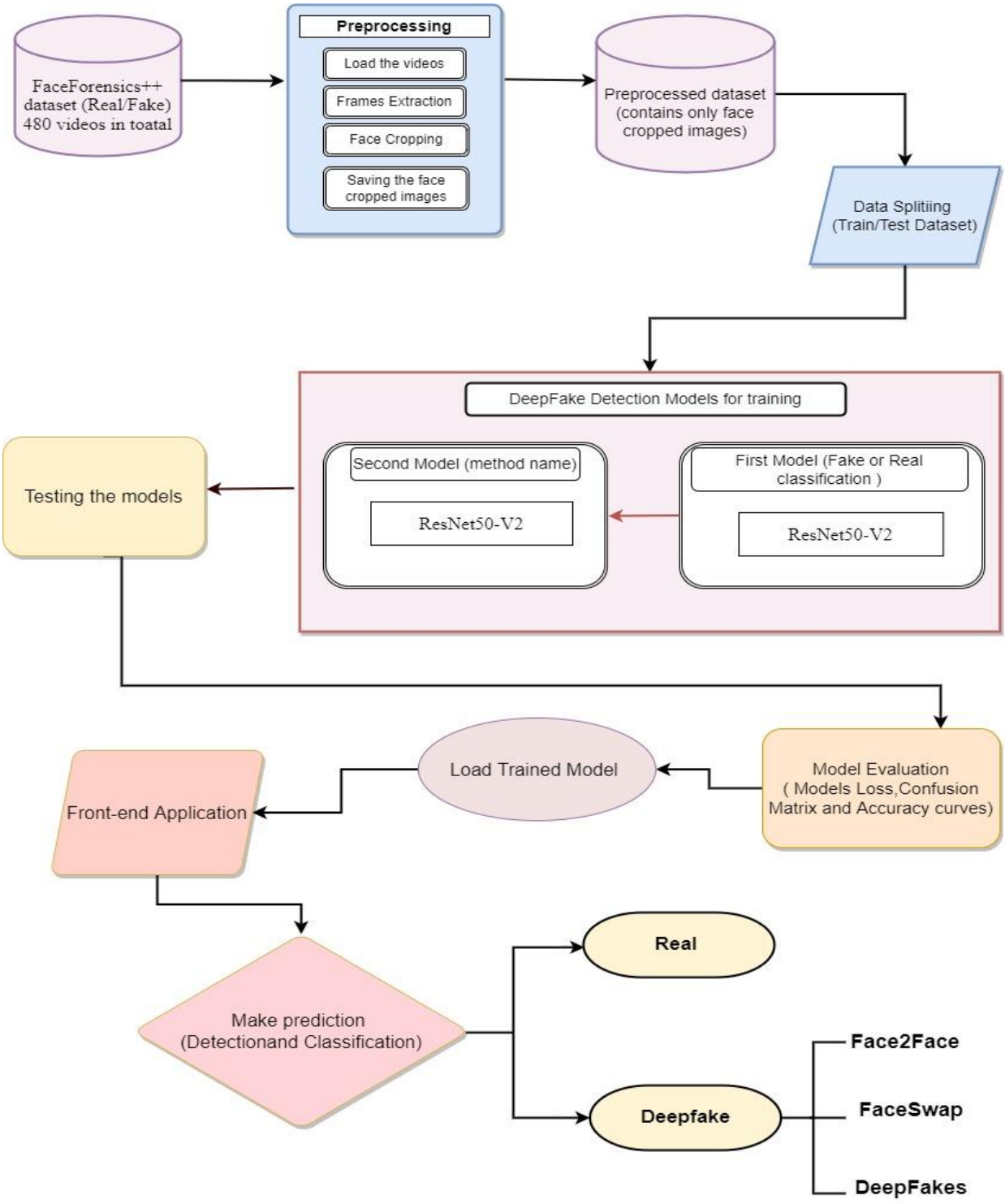


Fig 3.1. Deepfake Detection process (back-end+ front-end)

4. Back-end

When a user submits a video for detection through the web site frontal, the back-end calls our first model to determine whether the video is fake or real. When the video is recognized as fake, the second model is called, and it detects which approach was used to generate this fake.

After the preprocessing we made to our dataset in the first step, we applied Keras' *ImageDataGenerator* class to add data to our images and labels them based on their folders location. It offers a variety of augmentation options, including standardization, rotation, shifts, flips, brightness changes, and more. Using augmented images helps in obtaining more accurate results because the more data we have, the better the model learns.

Then, we used *"keras.applications.resnet v2.preprocess input"* a ResNet50V2 preprocessing function (each model has its own Keras preprocessing function), to convert the images into numerical data (NumPy arrays). Each image is further reshaped into a size of 224 × 224 × 3 for simpler processing as well as standardization purposes. Keras models can be built in two ways, sequential and functional:

- **Sequential API**: For most situations, it allows you to build models layer by layer. It has limitations in that it does not allow you to design models with several layers or inputs and outputs.
- **Functional API:** on the other hand, the functional API enables you to design models in a more flexible way. You can easily define models with layers that connect to more than only the previous and next layers. Layers can be connected to (literally) any other layer.

For this work we tried both ways. We used the following transfer learning workflow:

1. Instantiate a base model and load pre-trained weights into it.
2. Freeze all layers in the base model by setting *trainable = False*.
3. Create a new model (sequential or functional) on top of the output of one (or several) layers from the base model.
4. Train the new model on our new dataset [47].

As illustrated in **Fig 3.2.**

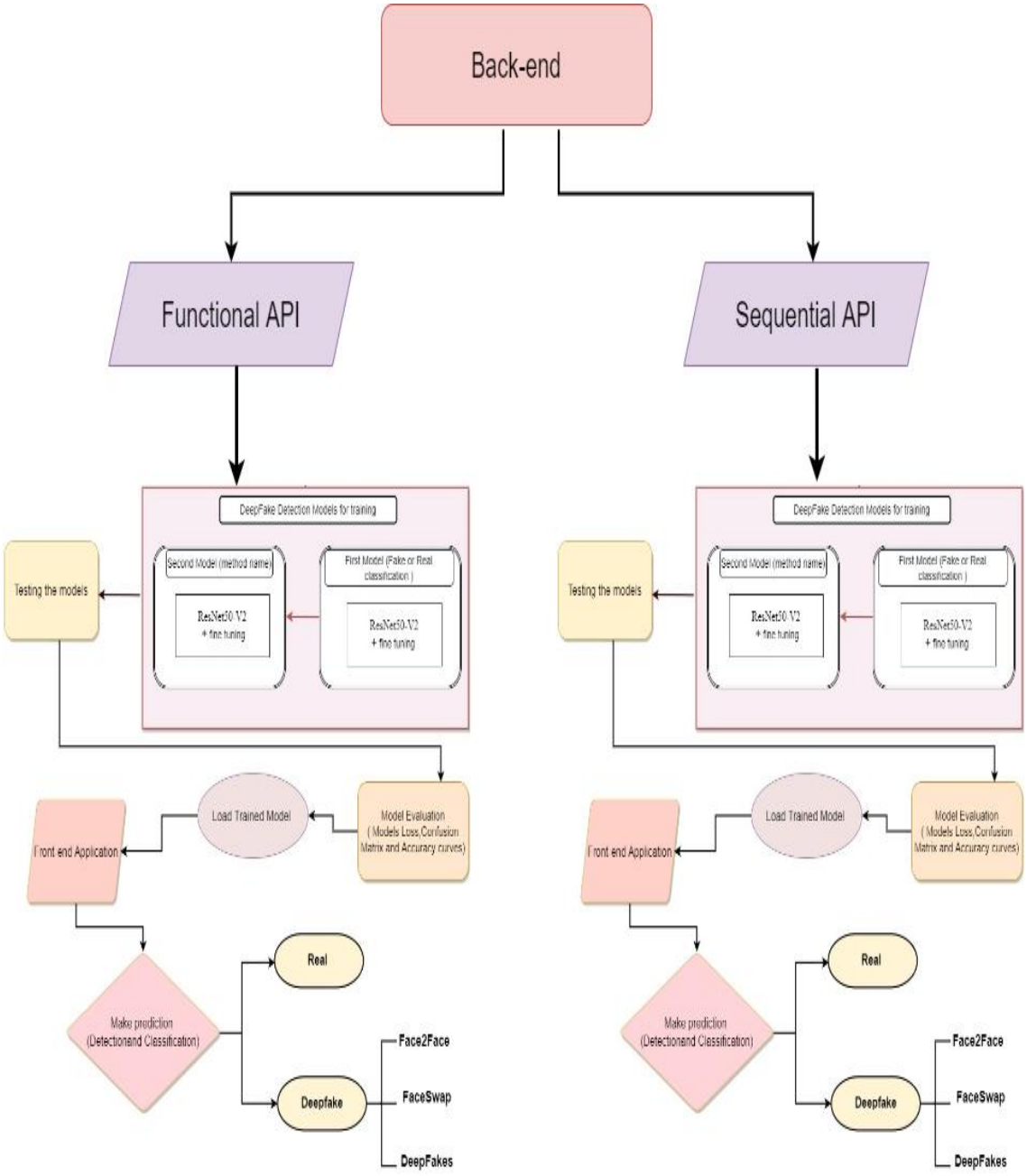


Fig 3.2. Sequential and Functional API process

4.1. Functional API

4.1.1 First Model

In step 3 of our workflow, while creating our model and after trying many layers in order to increase our accuracy we end up by adding a GlobalAveragePooling2D (GAP) layer followed by a dropout function. This was done to make the training process more efficient and to help the model generalize better. Finally, the output of this model was a Dense layer for prediction containing 1 neuron to classify 0 or 1 with sigmoid activation function which is specially for

binary classification as shown in **Fig 3.3.** The sigmoid function always returns a value between 0 and 1. In the compiling and fitting phase, we used a Stochastic gradient descent (SGD) optimizer with learning rate scheduler. The model was trained for 50 epochs. The training time of our model, on a sample of 9000 images for training and 5400 images for testing our model, took around 560s (9.3 min) our local machine with the characteristic described in Table 3.2.

```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0

sequential (Sequential)      (None, 224, 224, 3)       0

resnet50v2 (Functional)      (None, 7, 7, 2048)        23564800

global_average_pooling2d (G  (None, 2048)              0
lobalAveragePooling2D)

dropout (Dropout)            (None, 2048)              0

dense (Dense)                (None, 1)                 2049

=================================================================
Total params: 23,566,849
Trainable params: 23,521,409
Non-trainable params: 45,440
_________________________________________________________________
```

```python
from keras.layers import BatchNormalization
inputs = keras.Input(shape=(224,224,3))
data_augmentation = keras.Sequential([
        layers.experimental.preprocessing.RandomRotation(0.125),
        layers.experimental.preprocessing.RandomWidth(0.1),
        layers.experimental.preprocessing.RandomHeight(0.1),
        layers.experimental.preprocessing.RandomZoom(-0.1),
        layers.experimental.preprocessing.RandomTranslation
        (width_factor=0.1, height_factor=0.1),
        layers.experimental.preprocessing.RandomFlip("horizontal")
])
x = data_augmentation(inputs)

x = base_model(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x)

outputs = keras.layers.Dense(1, activation='sigmoid')(x)

model = keras.Model(inputs,outputs)
```

Fig 3. 3. First Functional Model Architecture

We tried the same process we followed in our first model in functional API, with the seven models, and we got the results resumed in **Table 3.2.** We use global average pooling (GAP) and Global max pooling 2D (GMP) blocks as an alternative to the Flattening block after the last pooling block of our convolutional neural network. They take a tensor of size (input width) x (input height) x (input channels) and computes the average value of all values across the entire (input width) x (input height) matrix for each of the (input channels). The output is thus a 1-dimensional tensor of size (input channels). The learning rate (LR) controls the step size of gradient descent, and different learning rates have great impact on the model's convergence and classification accuracy. The relationship between learning rate and classification results is explored in order to improve the experimental results. The precising the best learning rate depends on selected model and its use case. Thus, experience is the best way to know its value. Area under curve (AUC) is used to see how well our classifier can separate positive and negative examples. The AUC metric computes the area under a discretized curve of true positive versus false positive rates. [48]

Several trials were applied to establish the suitable parameters for fine-tuning, compiling, and fitting the model.

There are several evaluating techniques available to evaluate our models. Among them, the Receiver Operating Characteristic (ROC), accuracy, sensitivity, specificity, and the confusion matrix which we employed in our study and gave us accuracy and AUC results **Table 3.2** shows the final and optimal results.

Table 3.2: Obtained results in Functional API

Model	GMP	GAP	LR	Dropout	Accuracy		AUC
					Train	Test	
ResNet50V2	×	✓	0.001	0.2	69.9%	65%	0.54
	✓	×	0.001	0.3	64.99%	50%	0.50
	×	✓	0.0006	0.1	64,0%	58.5%	0.59
	×	✓	0.0004	0.1	85.0%	58.0%	0.58
	×	✓	0.0002	0.1	80.0%	58.6%	0.59
	×	✓	0.0002	0.3	85.0%	56.2%	0.56
	×	✓	0.0001	0.3	**80.0%**	**69%**	0.58
	×	✓	0.0001	0.2	**89.9%**	**59.62%**	0.6
	✓	×	0.0001	0.2	40.0%	0,49	0.49
	×	✓	0.0001	0.1	80.0%	56.6%	0.57
VGG16	×	✓	0.001	0.3	**75.0%**	**70.3%**	0.59
	✓	×	0.001	0.3	50.0%	59.1%	0.49
	×	✓	0.001	0.2	75.0%	70.1%	0.58
	×	✓	0.0001	0.3	60.0%	57.6%	0.58
Xception	✓	×	0.001	0.3	44.9%	**73%**	0.61
	✓	×	0.001	0.2	60.0%	64%	0.45
	✓	×	0.0001	0.3	**75.0%**	**59.0%**	0.49
	×	✓	0.0001	0.2	55.0%	51%	0.43
	✓	×	0.0001	0.2	69.9%	62%	0.59
MobileNetV2	✓	×	0.001	0.2	34.9%	50%	0.50
MobileNetV3	✓	×	0.1	0.1	40.0%	50%	0.50
	✓	×	0.001	0.2	60.0%	50%	0.50
	✓	×	0.001	0.3	44.9%	50%	0.50
	×	✓	0.0001	0.2	55.0%	50%	0.50
	✓	×	0.0001	0.2	55.0%	68.8%	0.60
INceptionV3	×	✓	0.001	0.2	44.9%	0,49	0.49
Inception-ResNetV2	×	✓	0.001	0.2	60.0%	66.8%	0.56
	×	✓	0.0001	0.2	69.9%	60.8%	0.51
	✓	×	0.0001	0.2	64.9%	63.8%	0.53
	×	✓	0.0001	0.3	64.9%	64.7%	0.54

×: is the none used layer. ✓ : is the used layer.

As a result, ResNet50V2 achieved the highest accuracy of 89.9% in the Functional API when implementing the GAP layer with a learning rate of 0.0001 and a dropout value of 0.2.

4.1.2 Second Model

Since the first experience illustrated that ResNet50V2 is the best, we continue to use it through the rest of this work. The only difference between the first and second models is the last layer, as shown in **Fig 3.4.** Because we have multiclass classification, we have to change the output (last layer) to a dense layer containing three neurons with a Softmax activation function to classify the three methods. We achieved the maximum accuracy with 75% for training with functional API. After we run several experiments where we changed parameter settings in order to improve our model's accuracy, mostly changing the dropout and learning rate values.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 224, 224, 3)]     0

 sequential (Sequential)     (None, 224, 224, 3)       0

 resnet50v2 (Functional)     (None, 7, 7, 2048)        23564800

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 3)                 6147

=================================================================
Total params: 23,570,947
Trainable params: 23,525,507
Non-trainable params: 45,440
```

```
inputs = keras.Input(shape=(224,224,3))
data_augmentation = keras.Sequential([
        layers.experimental.preprocessing.RandomRotation(0.125),
        layers.experimental.preprocessing.RandomWidth(0.1),
        layers.experimental.preprocessing.RandomHeight(0.1),
        layers.experimental.preprocessing.RandomZoom(-0.1),
        layers.experimental.preprocessing.RandomTranslation
        (width_factor=0.1, height_factor=0.1),
        layers.experimental.preprocessing.RandomFlip("horizontal")
])
x = data_augmentation(inputs)
#x = data_standardization(x)

x = base_model(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.1)(x)

outputs = keras.layers.Dense(3, activation='softmax')(x)

model = keras.Model(inputs,outputs)
```

Fig 3. 4: Second Functional Model Architecture

4.2. Sequential API

As we mentioned before A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor**.**

4.2.1 First Model

We started by feeding our images to a 2D convolution layer in a functional way, which detects spatial features in an image with Relu activation function. The output of this layer was given as a sequential input to a flatten layer, which feeds data from convolutional layers into the Dense layer that follows it, and because Dense layers operate on 1D input, the Flatten block reshapes the input, which is a tensor of 2-dimensional vector into a 1-dimensional vector. This dense layer, which has 256 units and uses Relu activation function, is followed by a batch normalization layer, which normalizes its inputs using mini-batch statistics to avoid overfitting and increase learning rate. It is followed by a dropout layer, a dense layer of 128 units with Relu activation function, as well as another dropout layer.

Finally, as illustrated in **Fig 3.5**, the output of this sequential model is a dense layer with one unit and a sigmoid activation function.

We trained our model for 70 epochs with an SGD optimizer.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 3)       30

 model (Functional)          (None, 100352)            23564800

 flatten_1 (Flatten)         (None, 100352)            0

 dense (Dense)               (None, 256)               25690368

 batch_normalization (BatchN  (None, 256)              1024
 ormalization)

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 1)                 129

=================================================================
Total params: 49,289,247
Trainable params: 25,723,935
Non-trainable params: 23,565,312
```

```
output = base_model.layers[-1].output
output = keras.layers.Flatten()(output)
base_model = keras.Model(base_model.input,output)
for layer in base_model.layers:
    layer.trainable = False
x = Conv2D(3,(3,3), activation="relu",input_shape=
          (img_rows, img_cols,1), padding='same')
model = Sequential()
model.add(x)
model.add(base_model)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.1))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

Fig 3. 5: First Sequential Model Architecture

After we run several experiments where we changed parameter settings in order to improve our model's accuracy the obtained results are resumed in Table 3.3.

Table 3.3: obtained results in Sequential API

Drop Out 1	Drop Out 2	LR	Accuracy		AUC
			Train	Test	
0.3	0.3	0.0002	94.9%	49%	0.49
0.1	0.1	0.0001	89.9%	50%	0.50
0.1	0.1	0.001	**100%**	59%	0. 60
0.3	0.3	0.0001	94.9%	51%	0.51
0.25	0.25	0.0004	89.8	49%	0.49

4.2.1.1 Models Evaluation

As shown in Table 3.3 the highest accuracy result was 100% for training and 60% for testing. Henceforth, we conclude that sequential API gave us better results in our case. The confusion matrix, Accuracy and Loss, receiver-operating curve (ROC), area under the ROC curve (AUC), are some of the evaluation methods we used to evaluate our first model.

- **Accuracy and Loss**

The two most well-known and discussed terms in machine learning are accuracy and loss. Accuracy is a metric for evaluating the performance of a classification model. It's the number of predictions in which the predicted value matches the true value [49]. During the training

phase, accuracy evolution is frequently plotted in a graph like shown **Fig 3.6**. The loss function, also known as a cost function, considers the probability or uncertainty of a prediction based on how much it differs from the true value. This provides a more comprehensive view of the model's performance as illustrated in **Fig 3.7**. Decreasing the loss value to closer to null is the aim behind increasing the training epochs. Because, accuracy increases as the loss value decreases.

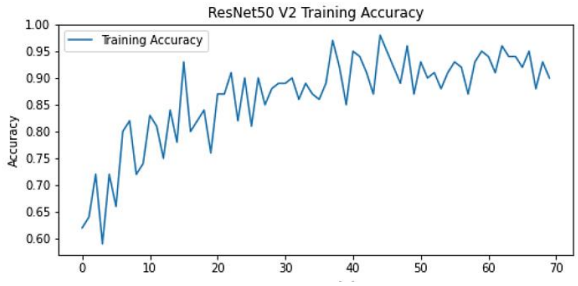


Fig 3. 6 : Training Accuracy plotted graph

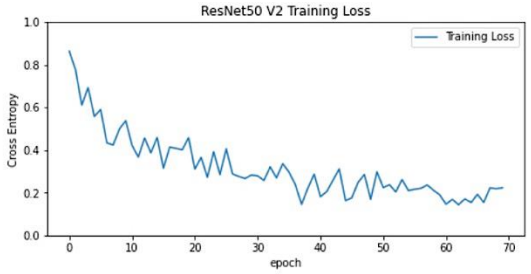


Fig 3. 7: Training Loss

- **The Confusion Matrix**

The confusion matrix is a specialized 2 dimensions array that shows the classifier's performance. It is commonly referred to as the error matrix in the field of machine learning. Depending on the data type, an image region is considered to be positive or negative. A judgment for the observed result can also be correct (true) or incorrect (false). As a result, one of four options will be considered: true positive (TP), true negative (TN), false positive (FP), and false negative (FN) [49]. See **Fig 3.8**

•TP: True Positive is the total number of truly identified fake images (1737).

• TN: True Negative is the total number of truly identified real images (1484).

• FP: False Positive is the total number of images predicted by the model fake and they are actually real (1216)

• FN: False Negative is the total number of images predicted real and they are actually fake (963).

The testing Accuracy is the measure of a correct prediction made by the classifier as shown in the equation:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{TP+TN+FN+FP} \quad (1)$$

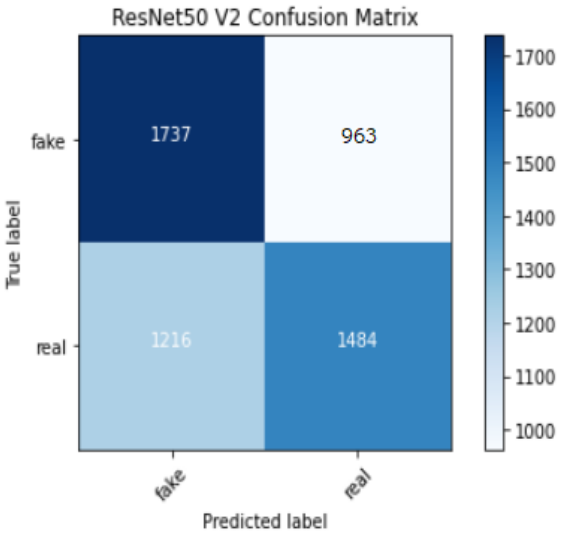


Fig 3. 8: Confusion Matrix

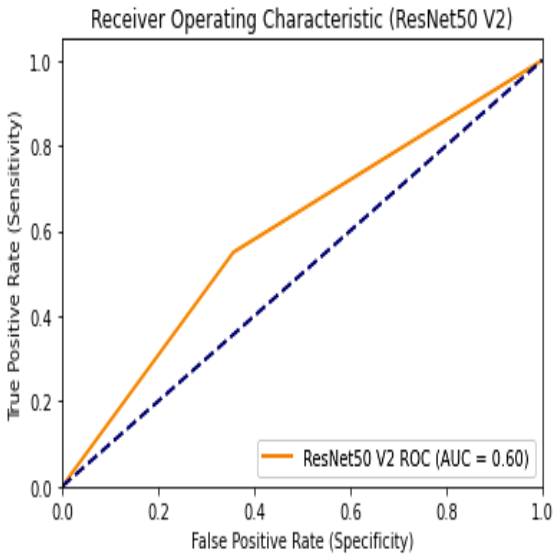


Fig 3. 9: ROC Curve

- **The Receiver Operating Characteristic (ROC)**

When the model's target is a binary classification, the ROC Curve appears as an additional tab to the Confusion matrix. The curves indicate a model's performance by plotting the obtainable recall and specificity as functions of the tolerable false positive rate illustrated in equations (2) and (3) respectively. It can also be used to determine the appropriate threshold value for our model. As illustrated in **Fig 3.9.**

Sensitivity or Recall are known as True Positive Rate.

Specificity is known as True Negative Rate.

$$\text{Sensitivity (recall)} = \frac{\text{TP}}{TP+FN} \quad (2)$$

$$\text{Specificity} = \frac{\text{TN}}{TN+FP} \quad (3)$$

4.2.2 Second Model

In our second model, we used the same process we followed with our first model. The only difference is the last layer, as shown in **Fig 3.10.**

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 3)       30

model (Functional)           (None, 100352)            23564800

flatten_1 (Flatten)          (None, 100352)            0

dense (Dense)                (None, 256)               25690368

batch_normalization (BatchN  (None, 256)               1024
ormalization)

dropout (Dropout)            (None, 256)               0

dense_1 (Dense)              (None, 128)               32896

dropout_1 (Dropout)          (None, 128)               0

dense_2 (Dense)              (None, 3)                 387

=================================================================
Total params: 49,289,505
Trainable params: 25,724,193
Non-trainable params: 23,565,312
```

```
output = base_model.layers[-1].output
output = keras.layers.Flatten()(output)
base_model = keras.Model(base_model.input,output)
for layer in base_model.layers:
    layer.trainable = False
x = Conv2D(3,(3,3),
           input_shape=(img_rows, img_cols,1),
           padding='same')

model = Sequential()
model.add(x)
model.add(base_model)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(3, activation='softmax'))
model.summary()
```

Fig 3. 10: Second Sequential Model Architecture

4.2.2.1 Model's Evaluation

With the same parameter values that gave us 100% percent with the first model, we were able to obtain the maximum accuracy of 100% percent for training and 50% for testing. The evaluation methods we used to evaluate our second model are Accuracy, Loss and the confusion matrix. Their results are illustrated in **Fig 3.11, 3.12** and **3.13**, respectively.

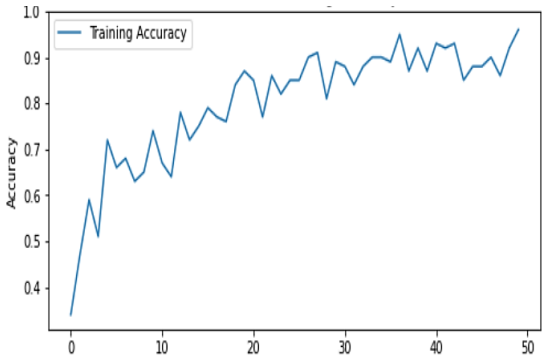


Fig 3. 11 : Obtained Accuracy

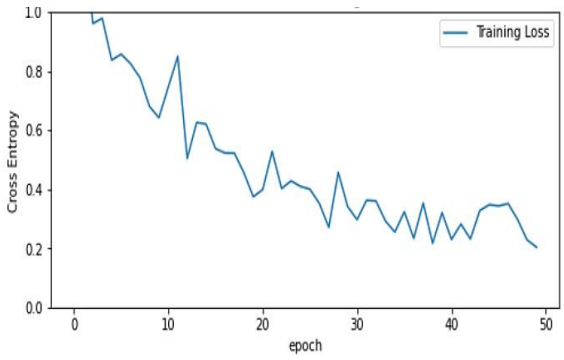


Fig 3. 12 : Obtained Loss

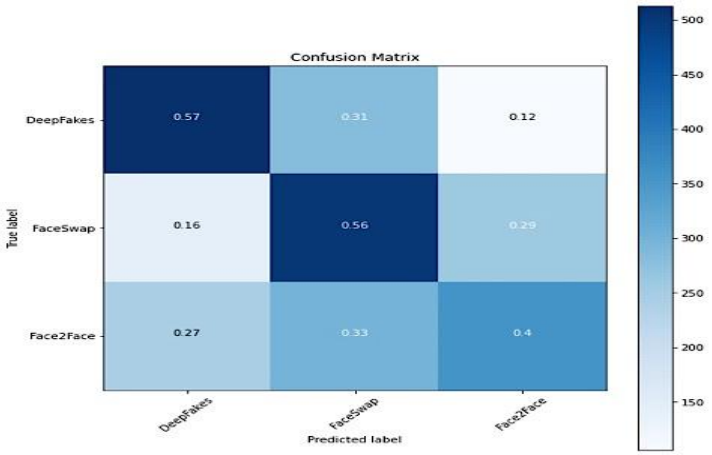


Fig 3. 13 : Confusion Matrix

5. Results Discussion

Our model achieved a good performance translated by its high accuracy during the training phase. However, we noticed a poor performance during the test phase which was conducted using a new dataset that is totally different from the one used in the training. Thus, we presumed that we are in case of overfitting which may be caused by multiple causes, such as:

- The training data has not been cleaned enough and contains noise.
- The model has a lot of variation.
- The training dataset is insufficient
- The model is too complex [50].

We tried several experiments to increase the testing accuracy, such as adding two dense layers, one at the top with 512 units and the other at the bottom before the output layer with 32 units, and each one followed by a dropout layer, but the results remained the same. Also, we attempted to eliminate the first functional layer of conv2D, yet, the accuracy dropped to 75%. Then, we tried changing the split to 80% for training and 20% testing, but it did not make any difference in increasing the testing accuracy.

Due to our limited resources, we believe that the main cause of overfitting in our situation is that the size of our dataset is insufficient. Perhaps, if we chose 60 or 70 frames instead of 30 frames, the problem might be solved. Or if we choose more varied fakes.

During the design phase we had two options, the first is to use one model and train it to distinguish between real videos, fakes created by DeepFakes, fakes generated by FaceSwap and Face2Face fakes at one go. The second was to make a model that distinguishes between fake and rea videos and another model that differentiates between the fakes depending on their creation methods DeepFakes, FaceSwap, and Face2Face. Our decision to use the second proposal came after conducting two experiences one for each proposal and unified the models' parameters and compared the results. We noticed that the second proposal gave better training accuracy results with 100% than the first one with 89%. Thus, we used the second option and reported its results in this manuscript.

In the second model, we noticed that the model can predict better on FaceSwap and Deepfakes videos, which belongs to the identity manipulation category, unlike Face2Face which belongs to to the facial expression manipulation category. Hence, we conclude that the second category is harder to detect even by the human eye.

6. Front-end

Our back-end was implemented in Jupyter Notebook, and our front-end with Flask, so to relate our back-end with the front-end, where the user inputs a video and it will be passed to our models, based on the training given to them. Then, the models make their classification and display the result to the user.

Noting that our models were saved as "hdf5" extension to avoid training it from scratch every time we need the model.

In order to interact with users, our web Application instruct users whether to upload a video from their local machine. Our web Application is mainly composed of three interfaces. **Fig 3.14** shows the illustration of our web application's interface.

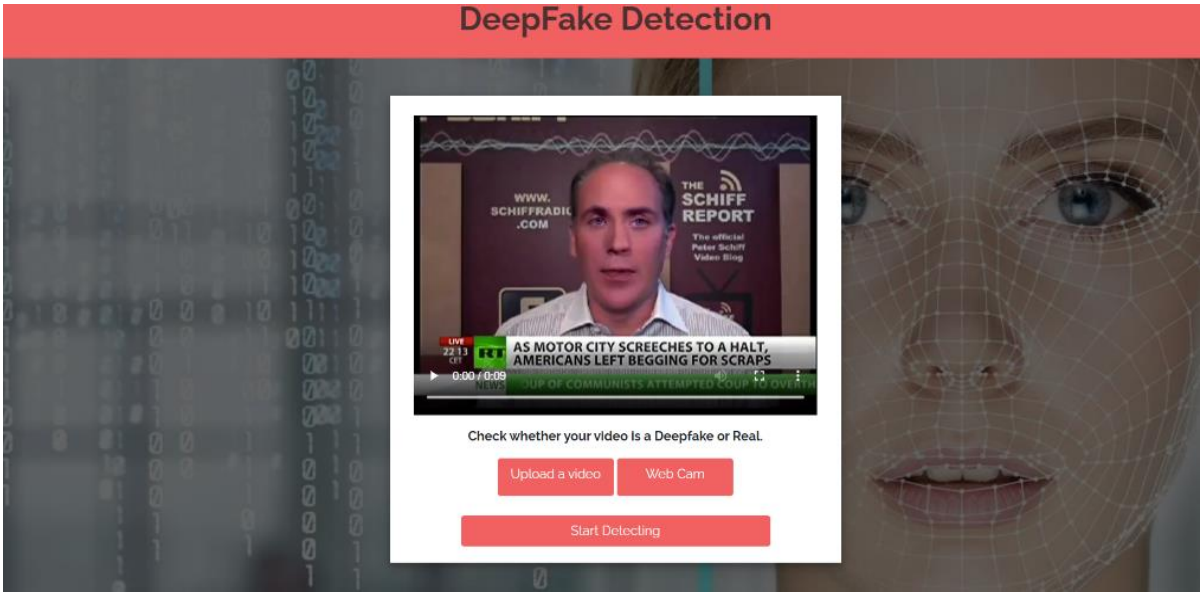


Fig 3. 14: Interface of our Web application

The steps for users to interact with our interface are as following:

- The user has the option of uploading a video from his computer or selecting for real-time detection via his webcam.
- First, if the user chose to upload a video by clicking Upload a video button, s/he will have access to chose a video from his local machine (restricted only for videos).
- The submitted video will be sent to the back-end after clicking start detecting button.
- In order to have high prediction results, we decided to extract 100 frames from the submitted video, and pass those images to our first model to detect whether the video is fake or real by counting the average of predictions, with its percentage result. If the video

is real the second interface will appear (detection interface) as shown in **Fig 3.15**. If the video is fake the detection interface will pop up along with a button to detect the name of the method created by. Once the user clicks the button "Methods Name". It will pass the extracted frames to the second model for classification. The name of the method that generated the fake will be displayed as illustrated in **Fig 3.16**.

- If the user did not select any of the prior options and hits the Start Detecting button, the detection interface will display an error, as depicted in **Fig 3.17**.

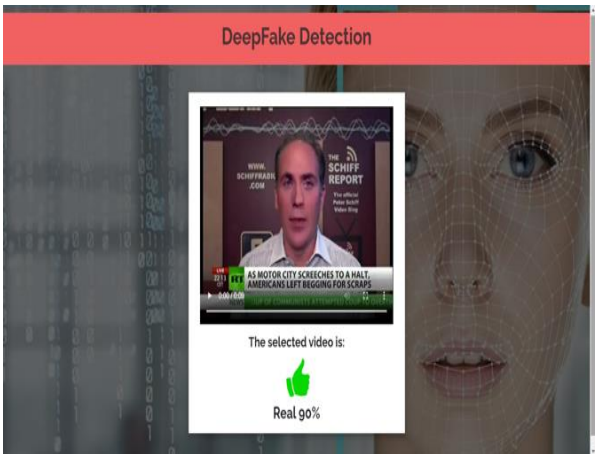


Fig 3. 15 : Detection interface (Real Video case)

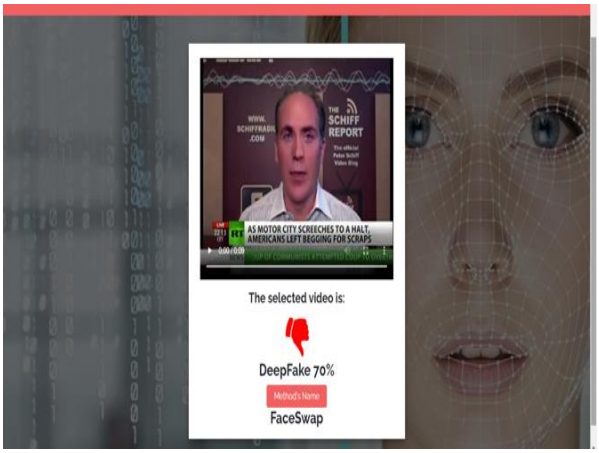


Fig 3. 16: Detection interface (fake Video case)

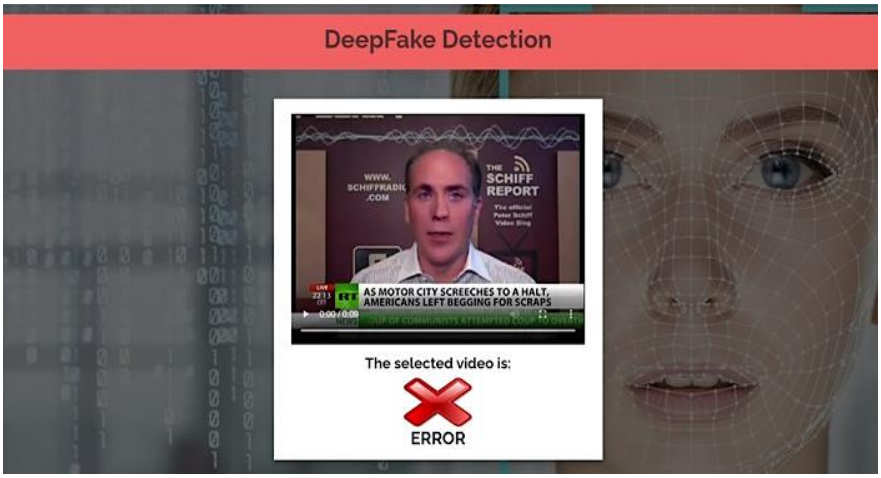


Fig 3. 17: Detection interface (Error case)

- Secondly, if the user selected real-time detection with his web cam, the web cam screen will appear, and the user can capture as many frames as he likes by pressing the space button, which will be saved automatically as seen in **Fig 3.18,** once he presses the start detecting button, those frames will be passed to our models for prediction.

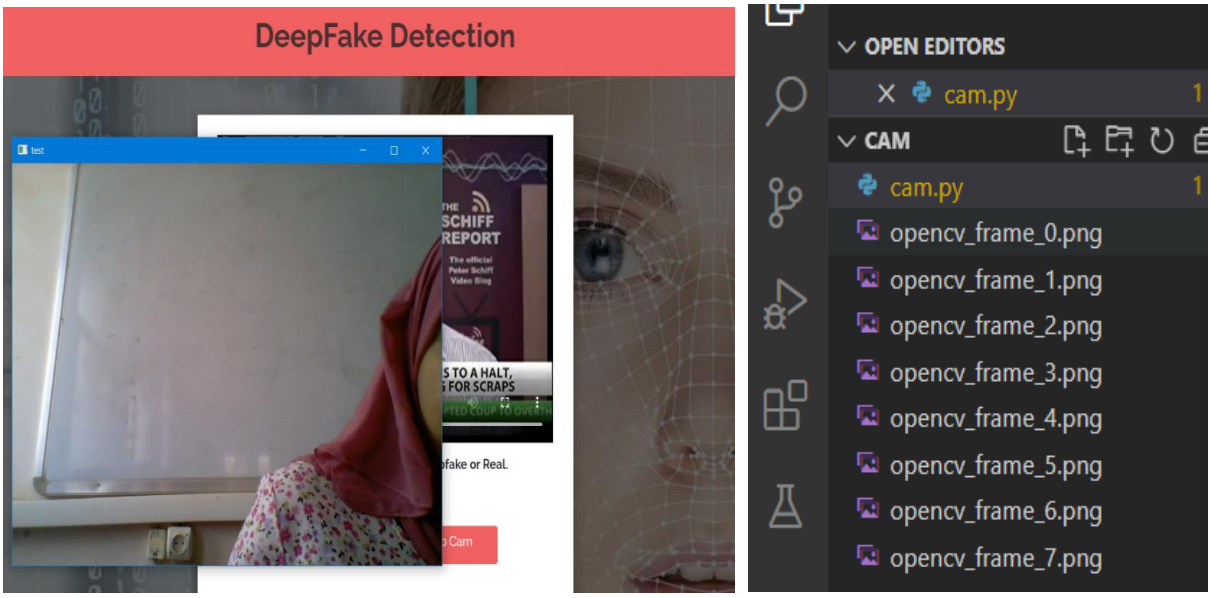


Fig 3. 18: Web cam detection

This approach is designed to test whether our model can recognize facial manipulations like face swapping or filters made by a real-time software tools or desktop applications and see if our model is able to detect them. Or in case if we choose a deepfake content on our mobile or tablets and hand it to the webcam, and see if our model can detect it as fake or not.

7. Conclusion

In this chapter, we presented our open platform called DeepFake Detection, which consists of two main parts: Back-end and Front-end. The Back-end is our ResNet50V2 models trained to detect deepfakes which are able to first distinguish between fake and real videos. Then, classify the method used to create the fake ones. We evaluated our models and discussed the results we got. The Front-end is our implementation interface that allows users to interact with our models.

CONCLUSION

General Conclusion

In an ever-evolving world, we are on the brim of experiencing how technology can become bigger than us - an uncontrollable force for good or bad. Not all digital manipulations are harmful. However, In the last 20 years, technology has drastically altered our social patterns, interactions and media consumption. Thus, deepfake videos or images can cause an unprecedented amount of damage in a political environment, spread disinformation as well as the personal lives of many people to attack individuals and cause social, psychological, religious, mental, and political stress.

As a result, it is essential to create efficient countermeasures to detect and avoid deepfakes. Therefore, in this study we created a deepfake detection models, the first can classify videos as fake or real (binary classification), the second model detects its generation method among these three: FaceSwap, Face2Face, DeepFakes (categorical classification). We used deep learning techniques utilizing transfer learning and CNN for feature extraction.

This work is mainly composed by two parts: Back-end and Front-end. For the Back-end: FaceForensics++ was our chosen dataset in this work to train and test our models, the collected 480 videos were preprocessed using deep learning and data augmentation techniques for frames extraction (images), and to extract the ROI (faces), then we split this preprocessed data into train and test sets. After that we implemented seven CNN-pretrained models: VGG16, Inception-v3, Inception-ResNet-v2, Xception, MobileNet-V2, MobileNet-V3 and ResNet50-V2. According to our evaluation results, we found that ResNet50-V2 is the most suitable model which gave us the highest accuracy with 100% for training and 60% for testing when it comes to our first model, with the second model it achieved 100% for training and 50% for testing. For the Front-end we created a platform called Deepfake Detection in order for the users to interact with our web application.

The purpose of this study is to increase general netizens' awareness of deepfake and their ability to identify spoof media posted on social networks is much more important than debating the fact whether spoof media is true or not.

Future perspectives

As a future perspective, first, we would like to use other generation methods in addition to the three we used.

Further, the system did not take into account videos that had multiple people in frame although this could be easily implemented in the future.

Due to the limited computational capacity, we were unable to implement a large dataset by extending the number of frames (instead of 30 frames, we would like to try 60 or 70 frames). This would have increased our testing accuracy and prevented overfitting.

Another method we would like to attempt in the future is to use K-fold cross-validation and Regularization techniques like Lasso and Ridge to prevent overfitting, but due to time constraints, we decided to delay it as future perspective.

The proposed Real-Time Detection approach could be applied as an extension for applications such as Google Meet, Zoom, Skype, Discord, and others. That would be useful for personal and confidential meetings, such as a conversation between a client and a bank employee regarding money transfers, or an online critical declarations or decisions in government, military, ministry, and so on. It can also be used as a mobile application for enhanced functionality.

We might also add the possibility do detect deepfakes by URLs.

AUTO EVALUATION GRID

AUTO EVALUATION GRID

Task/objective	State	Details and remarks
☒: Achieved. ☐: Not achieved		
Understanding deep learning.	☒	
Historical review of the appearance and evolution of deepfakes.	☒	
Consequences of the spread of deepfakes.	☒	Positive and negative use cases and their impact on society.
Motivation and Project background.	☒	Deepfakes has brought a growing concern when it comes to global stability, cyber risks, deceptive media content and privacy violation
Efforts to prevent deepfake spread.	☒	there is a pressing need to combat the rise of deepfakes, especially ones that will be used in a malicious manner
The general Deepfake detection process using deep learning approach.	☒	One promising countermeasure against deepfakes is deepfake detection
Existing deepfake detection models in literature.	☒	there have been numerous new detection methods of deepfakes. We mentioned few of them briefly
Dataset availability with multiple choices.	☐	Due to the dataset limitations, we did not have many choices we only had one choice which is FaceForensics++ dataset.
fill a google form to request FaceForensics++ dataset.	☒	After that they send us a script to guide us to download their dataset
Preprocessing and data augmentation.	☒	Frames extraction, ROI extraction (face), data split.
Back-end and Front-end construction.	☒	First model to classify real and fake videos ana the second to classify the generation method.
Model construction using CNN and transfer learning, and using seven available pretrained models	☒	keras application provides a set of pretrained models on image net dataset, after implementing them we chose the suitable one
Analyzing the detection results of the seven Pretrained models	☒	Plotting Accuracy and Loss, Confusion Matrix, AUC roc curve.
Avoiding overfitting.	☐	Due to our computational limmitation
Combining CNN with LSTM	☐	
Create a Graphical User Interface	☒	Deepfake detection web application
deepfake detection through the computer camera	☒	
deepfake detection through local machine (own videos)	☒	
Using urls for detection.	☐	Time limitations.
Add more generation methods	☐	Only 4 methods available.
Detecting multi faces in a video	☐	

REFERENCES

REFERENCE

Afchar, D., Nozick, V., Yamagishi, J., & Echizen, I. (2018, December). Mesonet: a compact facial video forgery detection network. In 2018 IEEE international workshop on information forensics and security (WIFS) (pp. 1-7). IEEE.[22]

Agnihotri, A. (2021). *DeepFake Detection using Deep Neural Networks* (Doctoral dissertation, Dublin, National College of Ireland).[42]

Boudouh, S. S., & Bouakkaz, M. (2022, May). Breast Cancer: Using Deep Transfer Learning Techniques AlexNet Convolutional Neural Network For Breast Tumor Detection in Mammography Images. In 2022 7th International Conference on Image and Signal Processing and their Applications (ISPA) (pp. 1-7). IEEE.[35]

Chang, Zi, B., , M., Chen, J., Ma, X., & Jiang, Y. G. (2020, October). Wilddeepfake: A challenging real-world dataset for deepfake detection. In Proceedings of the 28th ACM international conference on multimedia (pp. 2382-2390).[2]

Chesney, B., & Citron, D. (2019). Deep fakes: A looming challenge for privacy, democracy, and national security. Calif. L. Rev., 107, 1753.[7]

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), 861-874.[48]

Ferrer, C. C., Dolhansky, B., Pflaum, B., Bitton, J., Pan, J., & Lu, J. (2020). Deepfake detection challenge results: an open initiative to advance AI. Facebook AI,[online], https://ai. facebook. com/blog/deepfake-detection-challenge-resultsan-open-initiative-to-advance-ai.[19]

Güera, D., & Delp, E. J. (2018, November). Deepfake video detection using recurrent neural networks. In 2018 15th IEEE international conference on advanced video and signal-based surveillance (AVSS) (pp. 1-6). IEEE.[23]

He, Z., Gong, B., & Fan, D. (2019, January). Optimize deep convolutional neural network with ternarized weights and high accuracy. In 2019 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 913-921). IEEE.[38]

Hernandez-Ortega, J., Tolosana, R., Fierrez, J., & Morales, A. (2020). Deepfakeson-phys: Deepfakes detection based on heart rate estimation. arXiv preprint arXiv:2010.00400.[27]

Henry Ajder, Giorgio Patrini, Francesco Cavalli & Laurence Cullen. (September 2019). Deeptrace-the-State-of-Deepfakes-2019.[5]

Johansson, E. (2020). Detecting deepfakes and forged videos using deep learning. Master's Theses in Mathematical Sciences.[44]

Karasavva, V., & Noorbhai, A. (2021). The real threat of deepfake pornography: a review of canadian policy. Cyberpsychology, Behavior, and Social Networking, 24(3), 203-209.[5]

Khalid, H., Tariq, S., Kim, M., & Woo, S. S. (2021). FakeAVCeleb: a novel audio-video multimodal deepfake dataset. arXiv preprint arXiv:2108.05080.[30]

Kolagati, S., Priyadharshini, T., & Rajam, V. M. A. (2022). Exposing deepfakes using a deep multilayer perceptron–convolutional neural network model. International Journal of Information Management Data Insights, 2(1), 100054.[9]

Li, Y., & Lyu, S. (2018). Exposing deepfake videos by detecting face warping artifacts. arXiv preprint arXiv:1811.00656.[24]

Li, Y., Zhang, C., Sun, P., Ke, L., Ju, Y., Qi, H., & Lyu, S. (2021, May). DeepFake-o-meter: an open platform for DeepFake detection. In 2021 IEEE Security and Privacy Workshops (SPW) (pp. 277-281). IEEE.[26]

Li, Y., Yang, X., Sun, P., Qi, H., & Lyu, S. (2020). Celeb-df: A large-scale challenging dataset for deepfake forensics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3207-3216).[31]

Mahmud, B. U., & Sharmin, A. (2021). Deep insights of deepfake technology: A review. arXiv preprint arXiv:2105.00192.[11]

Masood, M., Nawaz, M., Malik, K. M., Javed, A., Irtaza, A., & Malik, H. (2022). Deepfakes Generation and Detection: State-of-the-art, open challenges, countermeasures, and way forward. *Applied Intelligence*, 1-53.[4]

Meskys, E., Kalpokiene, J., Jurcys, P., & Liaudanskas, A. (2020). Regulating deep fakes: legal and ethical considerations. Journal of Intellectual Property Law & Practice, 15(1), 24-31.[1]

Nguyen, T. T., Nguyen, Q. V. H., Nguyen, C. M., Nguyen, D., Nguyen, D. T., & Nahavandi, S. (2019). Deep learning for deepfakes creation and detection: A survey. arXiv preprint arXiv:1909.11573.[8]

Pan, D., Sun, L., Wang, R., Zhang, X., & Sinnott, R. O. (2020, December). Deepfake Detection through Deep Learning. In *2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)* (pp. 134-143). IEEE.[33]

Ragab, D. A., Sharkas, M., Marshall, S., & Ren, J. (2019). Breast cancer detection using deep convolutional neural networks and support vector machines. *PeerJ, 7*, e6201.[49]

Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1-11).[32]

Ruby, U., & Yendapalli, V. (2020). Binary cross entropy with deep learning technique for image classification. Int. J. Adv. Trends Comput. Sci. Eng, 9(10).[39]

Sun, F., Zhang, N., Xu, P., & Song, Z. (2021). Deepfake detection method based on cross-domain fusion. Security and Communication Networks, 2021.[28]

Tayseer, M., Mohammad, J., Ababneh, M., Al-Zoube, A., & Elhassan, A. (2020, April). Digital Forensics and Analysis of Deepfake Videos. In 11th International Conference on Information and Communication Systems (ICICS).[25]

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017, September). Domain randomization for transferring deep neural networks from simulation to the real world. In 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS) (pp. 23-30). IEEE.[36]

Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B., & Zheng, Q. (2020). IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. Computer Networks, 171, 107138.[47]

Verdoliva, L. (2020). Media forensics and deepfakes: an overview. IEEE Journal of Selected Topics in Signal Processing, 14(5), 910-932.[6]

Vinocur, J. (2022). Death by a Thousand Paper Cuts: The Scourge That Is Business Email Compromise. The Brief, 51(2), 10-21.[13]

Vrbančič, G., & Podgorelec, V. (2020). Transfer learning with adaptive fine-tuning. IEEE Access, 8, 196197-196211.[37]

Yang, X., Li, Y., & Lyu, S. (2019, May). Exposing deep fakes using inconsistent head poses. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 8261-8265). IEEE.[21]

Ying, X. (2019, February). An overview of overfitting and its solutions. In Journal of physics: Conference series (Vol. 1168, No. 2, p. 022022). IOP Publishing.[50]

Zaccone, G., Karim, M. R., & Menshawy, A. (2017). Deep learning with TensorFlow. Packt Publishing Ltd.[46]

Zhao, H., Zhou, W., Chen, D., Wei, T., Zhang, W., & Yu, N. (2021). Multi-attentional deepfake detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 2185-2194).[29]

[10]: https://outsourceit.today/real-deepfake-apps-and-websites/

[12]:https://www.creativebloq.com/features/deepfake-examples

[14]: https://www.motherjones.com/politics/2019/03/deepfake-gabon-ali-bongo/

[15]: https://www.technologyreview.com/2020/09/29/1009098/ai-deepfake-putin-kim-jong-un-us-election/

[16]:https://www.forbes.com/sites/thomasbrewster/2021/10/14/huge-bank-fraud-uses-deep-fake-voice-tech-to-steal-millions/?sh=6a225f207559

[17]:https://www.kaspersky.com/resource-center/threats/protect-yourself-from-deep-fake

[18]: https://www.cnbc.com/2019/09/20/hao-li-perfectly-real-deepfakes-will-arrive-in-6-months-to-a-year.html

[20]: https://towardsai.net/p/l/deep-learning-based-deepfake-detection-in-a-nutshell#443a

[3]: https://www.gq.com.mx/estilo-de-vida/articulo/deepfake-tom-cruise-controversia-redes-sociales

[34]-x: https://github.com/ondyari/FaceForensics/tree/master/dataset

[40]: https://iq.opengenus.org/resnet50

[41]: https://viso.ai/deep-learning/resnet-residual-neural-

[42]: file:///C:/Users/ASUS/Downloads/Documents/ambujagnihotri.pdf

[43]: https://viso.ai/deep-learning/vgg-very-deep-convolutional

[44]: file:///C:/Users/ASUS/Downloads/Documents/thesis_final.pdf

[45]:https://www.researchgate.net/publication/357965162_Visual_Sentiment_Analysis_Using_Deep_Learning_Models_with_Social_Media_Data#pf17