الجمهورية الجزائرية الديمقراطية الشعبية

People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University 8 May 45 -Guelma-

Faculty of Mathematics, Computer Science and Material Science

Department of Computer Science

Master's dissertation

Field: *Computer science*

Option: Science and technology of information and communication

Theme :

Fake Faces Detection Based on an Auto-Encoder Network (Application Developed on Jetson Nano)

Supervisor :

Dr.Bencheriet Chemesse Ennahar

Presented By:

TABA Zahra

June 2022

Thanking

First, I thank ALLAH, our one, and merciful GOD, for giving me the health and courage to complete this modest work. I implore him to guide me on the right path and wrap me in His goodness and blessing.

The work presented in this thesis was carried out in the LAIG laboratory (Laboratoire d'Automatique et Informatique de Guelma), supervised by the Director Kechida Sihem and engineer Fisli Soufiane whom I would like to thank for welcoming me to this laboratory.

I want to express my deep gratitude and sincere thanks to my supervisor Mme Chemesse Ennehar Bencheriet, a teacher in the Computer Science department 8 mai 1945 Guelma University, for her keen interest, her kindness, her inspiring advice, and her constant encouragement with my work at all stages, to carry out this thesis. It was a great pleasure for me to have the opportunity to work under her supervision.

I would also like to thank madame Brahmia Souhaira wishing her and her daughter all the success.

I would also like to thank the jury members who gave me the privilege of agreeing to review my work and for their time in this regard.

A big thanks to all teachers of the Computer Science department for their patience and dedication to the students.

I thank anyone who has given me a moment to help, advise, or encourage me.

Finally, I thank all my family and especially my mother for her patience and support throughout my journey may Allah bless her and lend her health and long life.

Dedication

I dedicate this work to my family. A special feeling of gratitude to my dear loving parents who gave me everything without expecting anything in return and for supporting and encouraging me in my life. May ALLAH, our GOD, the Almighty, One and Merciful, bless you with good health and long life.

To my dear sisters Somia, Yasmine, Mawada, and my dear loving brother Salah. source of joy and happiness to me, source of hope and motivation.

To all my friends who have always supported and encouraged me during these years of study, thank you.

ملخص

قد تضاعف جيل الوجوه الاصطناعية على مدى السنوات القليلة الماضية ، مما أدى إلى إنشاء صور ومقاطع فيديو مزورة واقعية للغاية تعتمد على تقنيات ” Deepfake“.

في هذه الأطروحة ، نقترح نظامًا يهدف إلى اكتشاف الوجوه المزيفة في الصور استنادًا إلى نوع معين من الشبكات العصبية يسمى الترميز التلقائي التلافيفي (CAE). يتيح لنا نهج التعلم العميق غير الخاضع للإشراف هذا الحصول على تمثيل بسيط لبياناتنا التي نستخدمها بعد ذلك لتقدير كثافة النواة وخطأ إعادة البناء للتمييز بين الصور "الخاطئة" و "الحقيقية". ".

لقد اخترنا استخدام Auto-Encoder Convolutionnel المعروف بأدائه في اكتشاف الحالات الشاذة. قمنا بتدريب AE على 50000 عينة من قاعدة بيانات .Flickr أسفرت الاختبارات التي أجريت على مجموعة بيانات الوجوه الزائفة StyleGan عن نتائج إيجابية ولكن يمكن صقلها عن طريق إطالة التدريب.

التطبيق مضمن في مجموعة مطوري ” NVIDIA Jetson Nano 2GB“ ، والتي تعتبر جهاز كمبيوتر صغير قوي للغاية لأنه يحتوي على وحدة معالجة الرسومات (GPU) مع 128 مركزًا ، تم تصميمه خصيصًا لتطبيقات الذكاء الاصطناعي والروبوتات.

كلمات مفتاحية : الوجوه المزيفة , الوجوه الحقيقية, الترميز التلقائي التلافيفي convolutional NVIDIA Jetson Nano 2GB , Autoencoder

RÉSUMÉ

Au cours des dernières années, la génération de visage synthétique s'est rapidement développée en créant des images et des vidéos forgées hyper-réalistes basées sur les techniques de Deepfake.
Dans ce mémoire de master, nous proposons un système qui vise à détecter les faux visages (fake faces) en utilisant un type spécifique de réseau neuronal appelé Auto-Encodeur Convolutionnel (CAE). Cette approche d'apprentissage en profondeur non supervisé nous permet d'obtenir une représentation simple de nos données sur laquelle nous utilisons ensuite l'estimation de la densité du noyeau et de l'erreur de reconstruction pour distinguer les images « fausses » et « réelles ».
Nous avons opté pour l'utilisation de l'auto-encodeur convolutif connu pour ses performances dans la détection des anomalies. Il a été entrainé sur 50 000 échantillons issues de la base Flickr. Les tests sur la base des faux visages StyleGan ont abouti a des résultats favorables mais qui peuvent être affinés en prolongeant l'apprentissage.
L'application a été réalisée sur la carte « kit de développement NVIDIA Jetson Nano 2GB » considérée comme un mini-ordinateur très performant car elle dispose d'un GPU avec 128 cœurs. Spécialement conçue pour les applications de L'IA et la robotique.

Mots clés: faux visages, vrais visages, Autoencoder, Auto-Encoder Convolutionnel, Deepfake, kit de devellopement NVIDIA Jetson Nano.

ABSTRACT

Synthetic face generation has overgrown over the past few years, creating hyper-realistic forged images and videos based on Deepfake techniques.
In this master's thesis, we propose a system that aims to detect fake faces in images based on a specific type of neural network called Convolutional Auto-Encoder (CAE). This unsupervised deep learning approach allows us to obtain a simple representation of our data on which we then use the estimate of kernel density and reconstruction error to distinguish between "false" and "real" images. ".
We have opted for the use of the Convolutional Auto-Encoder known for its performance in detecting anomalies. We trained the AE on 50,000 samples from the Flickr database. The tests performed on the false faces StyleGan dataset yielded favorable results but can be refined by prolonging the training.
The application is embedded in the NVIDIA Jetson Nano 2GB Developer Kit, considered a very powerful mini-computer because it has a GPU with 128 cores, specially designed for AI and robotics applications.

Key words: fake faces, real faces, Autoencoder, Convolutional Autoencoder, Deepfake, Jetson Nano developer kit.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

GENARAL INTRODUCTION

Creating incredibly realistic synthetic human face photos has become considerably more accessible than before due to the significant advancements made in deep-learning technologies named deepfakes.
Deepfake is a program that belongs to the GAN family, the Generative Adversarial Networks that relies on an auto-encoder, which consists of an encoder and a decoder network.
Face photos include rich and intuitive personal identity information, making them useful for biometric verification and identification. However, on the other hand, face images are vulnerable to forgery. They have a low level of privacy. They can be forged using deepfake technology that allows the transfer of facial expressions on image or video from a targeted person to another.
Several types of research have been carried out in recent years, and many machine learning applications have been developed to detect deepfakes.
For this main objective, we have therefore developed a system that aims to detect fake faces in images based on a specific type of neural network called Convolutional Auto-Encoder (CAE). This unsupervised deep learning approach allows us to obtain a simple representation of our data on which we then use the estimate of kernel density and reconstruction error to distinguish between "false" and "real" faces.

We have chosen to structure our study around four main chapters.

- **Chapter 1** : **Generalities on Deepfake technology**: This chapter provides a general overview of deepfake technologies. The last part of this chapter has been reserved for a state-of-the-art of recent research on detecting fake faces.

- **Chapter 2** : **Convolutional Neural Networks** : This chapter detailed the typical architecture of convolutional neural networks, which is a fundamental part of deep networks, and the different types of autoencoder networks.

- **Chapter 3** : **Conception**:In this chapter, we present the basic architecture of our system as well as the development and operation of the different modules.

- **Chapter 4** :**Implementation**: We reserved this chapter for the overall implementation of our system and the training and testing of autoencoder on faces and fake faces. Then we present some experimental results obtained by our model.

We end our study with a general conclusion and prospectives for future work that may be developed by other students.

CHAPTER 1 GENERALITIES ON DEEPFAKE TECHNOLOGIES

1 Introduction

In the last years, deep learning has become more powerful; fake images and videos, including facial information generation, have been rapid, particularly with deep learning-based Deepfake generation methods that have become a significant public concern. The deepfake technique can create fake images and videos by swapping a person's face with another person, which leads to misinformation and inspiring misunderstanding; nowadays, deepfakes are adopted by a wider audience, which is why deepfake detection is becoming a necessity.

2 Deepfake

Deepfakes are a combination of "deep learning" and "fake." They are hyper-realistic videos digitally manipulated to depict people saying and doing things that never actually happened, using face swaps that leave little trace of manipulation. Deepfakes are the product of artificial intelligence (AI) applications. That merges, combines, replaces, and superimposes images and video clips and also analyzes big datasets to learn to mimic a person's facial expressions, mannerisms, voice, and inflections to create fake videos that appear authentic [Wes19].

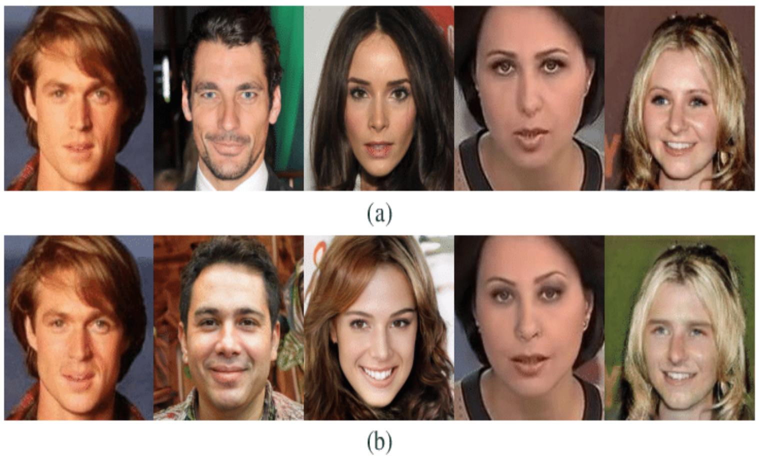


FIGURE 1.1: Deepfake example. (a) real face images, (b) fake face images

3 Importance of face in security systems and communication

The face represents the front part of the head that in humans extends from the forehead to the chin and includes the mouth, nose, cheeks, and eyes. This part presents the importance of face in two domains: security systems and communication.

a) In communication

Communication is often defined as relaying a message to another person or persons and exchanging information by speaking, writing, or using some other medium.
The importance of face in communication represents the distinction of being able to see the other party or parties in a conversation. It allows for a better exchange of information since both speaker and listener can see and interpret body language and facial expressions.[Sou+18]

b) In Security systems

Recent events, such as terrorist attacks, exposed a severe weakness in most sophisticated security systems. Various government agencies are now more motivated to improve security data systems based on body or behavioral characteristics, often called biometrics. A biometric system processes raw data to extract a template that is easier to process and store but carries most of the information needed. Face recognition seems to be a suitable compromise between reliability and social acceptance and balances security and privacy.[Aba+07]
Face recognition systems fall into two categories: verification and identification. Face verification compares a face image against a template face image whose identity is being claimed. On the contrary, face identification compares a query face image against all image templates in a face database to determine the identity of the query face.
But still, exist some factors that can significantly affect system faces recognition performances, such as Illumination, Pose, time delay (the face changes over time), and occlusions. [Aba+07]

4 History

- In 1997 [BCS97], a project called "Video Rewrite" altered existing video footage of people to make them seem like they were mouthing the words that appeared on a new audio track: essentially, putting words in their mouths..

- The early 2000s were reasonably silent as computer vision moved deeper into the facial recognition world. Developments in this field made drastic improvements to motion tracking that make today's deepfakes more convincing.

- In 2001 [CET01], it debuted the active appearance models algorithm. Using a thorough statistical model to match a shape to an image proved a big step forward. They made face matching and tracking significantly more efficient.

- In 2016 and 2017, two papers [Thi+16][SSKS17] established deepfakes as achievable with consumer-grade hardware, the Face2Face project out of the Technical University of Munich and the Synthesizing Obama project out of the University of Washington. They improved computing and rendering times while updating graphical fidelity to look photo-realistic.

Therefore, we can say that the first steps towards this technology were made in the 90s by academic institutions and were later adopted by a wider audience that made it later abused by criminals and by falling into the wrong hands, deepfakes can lead to chaos and uncertainty.
Deepfake is also considered one of the most dangerous uses of AI (Artificial Intelligence). So most of its real-world applications have either discrediting or fraudulent intentions. The victims of such actions are often famous people, celebrities, and politicians [w1].

5 Advantages and disadvantages of deepfake technologies

There are advantages of deepfake technology, given its popularity world wide. [DW21] These advantages are:

- Making digital voices for actors who lost theirs because of disease.
- Making digital voices for actors who lost theirs because of disease.
- Recreate classic scenes in movies; create new movies starring long-dead actors.
- Allows for automatic and realistic voice dubbing for movies in any language.

Disadvantages of deepfakes:

- Threat to world security when deepfake procedures can make videos of world leaders with forged speeches for fraudulent purposes.
- Abused to cause political or religious misunderstanding between countries, to fool the public, and affect results in election campaigns.

- Create confusion in financial markets by creating fake news.
- Create forged satellite broadcasting images of the Earth to hold items that do not exist to create chaos in the military..
- Share doctored footage of people.

6 Dataset

We present below the most used databases for detecting real and fake faces.

6.1 Flickr-Faces-HQ,FFHQ

The dataset FFHQ contains a collection of 70,000 face images with a high-quality resolution generated by generative adversarial networks (GAN). The images were collected from the Flicker platform and contain images with various accessories such as eyeglasses, sunglasses, hats, etc. According to the dataset author, a pre-processing step was done to prune and remove noises from images.[Alm21]

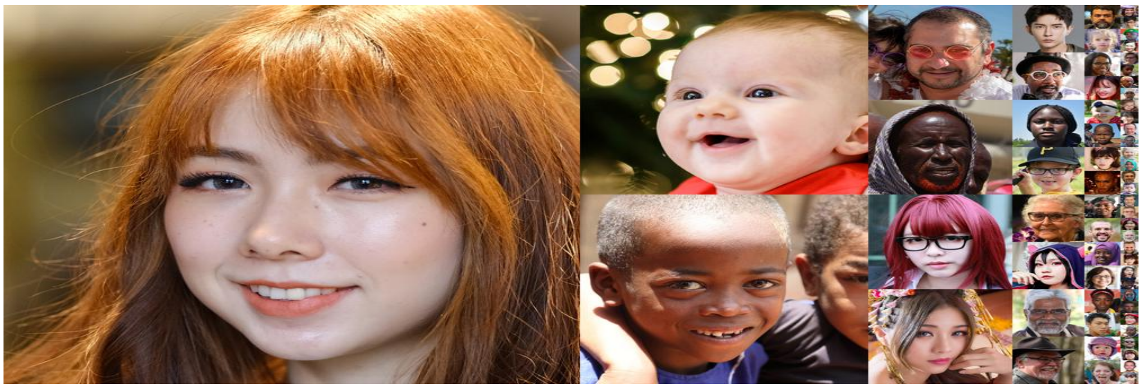


FIGURE 1.2: Faces-HQ, FFHQ dataset examples.

6.2 100K-Faces

100K-Faces is a well-known publicly available dataset that includes 100,000 unique human images generated using StyleGAN. StyleGAN was applied on a large dataset consisting of over 29,000 images gathered from 69 different models, generating photos with a flat background. [Alm21]

FIGURE 1.3: 100K-Faces example.

6.3 Diverse Fake Face Dataset (DFFD)

DFFD contains 100,000 and 200,000 fake images generated by adopting respective state-of-the-art methods (ProGAN and StyleGAN models). The dataset includes approximately 47.7% male photographs, 52.3% female images, and most of the samples range in age from 21 to 50 years old. [Alm21]

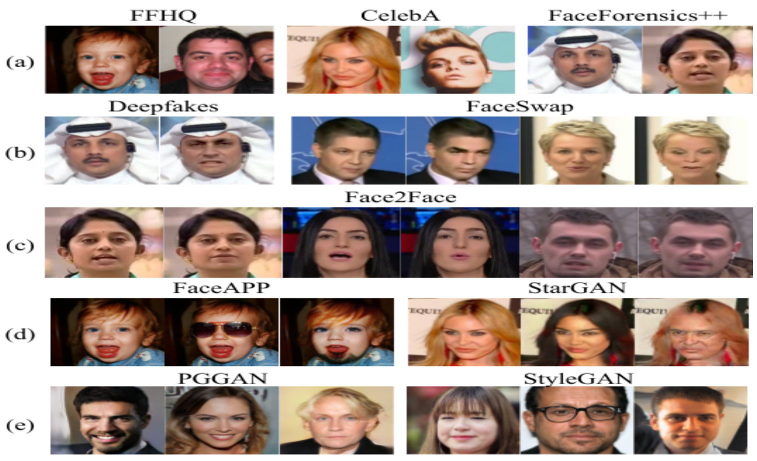


FIGURE 1.4: Diverse Fake Face Dataset (DFFD) example.

6.4 CASIA-WebFace

A database called CASIA-WebFace includes about10,000 subjects and 500,000 images. This dataset was first crawled from the IMDB website, containing 10,575 well-known actors and actresses of IMDB. Then the photos of those celebrities are extracted using clustering methods. [Alm21]

FIGURE 1.5: CASIA-WebFace dataset example.

6.5 VGGFace2

This database contains over three million face photographs from over nine thousand different subjects, with an average of over 300 images per subject. Images were gathered from the Google engine, which has a wide range of information, such as ethnicity, illumination, age, and occupation (e.g., actors, athletes, and politicians). [Alm21]

FIGURE 1.6: VGGFaces dataset example.

6.6 The Eye-Blinking Dataset

The eye-blinking dataset is specially designed to deal with eye blinking detection. This dataset consists of 50 interviews and videos for each person, lasting approximately thirty seconds with one eye blinking happening at least once. The author then tags the left and right eye states for each video clip using their tools. [Alm21]

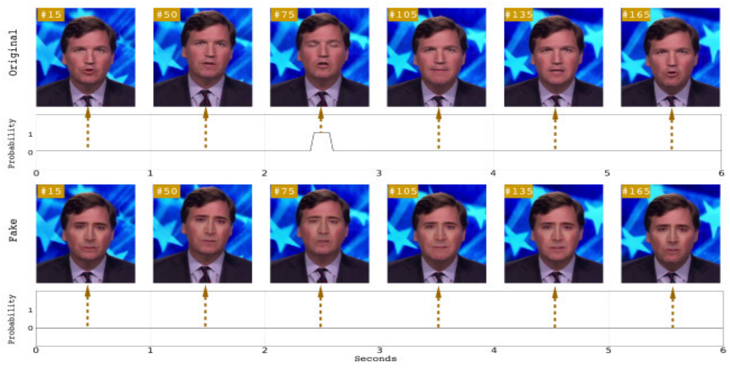


FIGURE 1.7: The Eye-Blinking dataset example.

6.7 DeepfakeTIMIT

DeepfakeTIMIT is a dataset of videos released using the database containing a collection of swapped faces videos generated using the GAN-based approach. The dataset was produced with a lower quality model with 64 × 64 sizes and a higher quality model with 128 × 128 input/output size. Each non-real video collection contains 32 subjects. The author created ten fictitious videos for each subject .[Alm21]

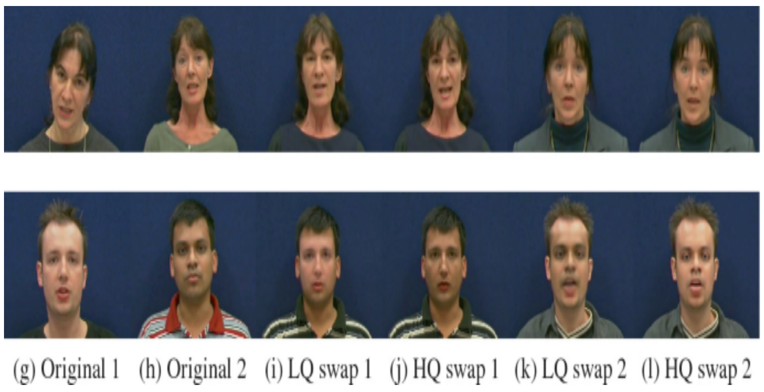


FIGURE 1.8: Deepfake TIMIT dataset example.

7 Related work on fake face detection

We presented in this part the summary of 10 articles that represent recent works of fake face detection.

Article 1: Swapped face detection using deep learning and subjective assessment [Din+20]

In this paper, researchers proposed the creation of the largest face-swapping detection dataset that uses face-swapping methods based on auto-encoding using Nirkin's method [Nir+18] with the Auto-Encoder-GAN method.
The module proposed in this study provides high accuracy prediction coupled with an analysis of uncertainties.
Researchers in this paper have also built a website that collects pairwise comparisons for 400 images from human subjects, which will later be used in comparing

the rank of the module.
The test result of the module proposed in this study, which has been evaluated over diverse databases and compared the results with the ranking from their website, found that it gives an excellent correspondence to human ranking, which proves the module's effectiveness for detecting swapped faces.
The tests were carried out to show the effectiveness of the proposed method in practice, with true positive rates greater than 96% with very few false alarms.

Article 2: Learning Spatio-temporal features to detect manipulated facial videos created by the Deepfake techniques [Ngu+21]

In this article, researchers have proposed a deep 3D convolutional neural network that can simultaneously learn spatial and temporal features, to detect deepfake videos in a consecutive image sequence using the two datasets FaceForensics++ and VIidTIMIT.
The method proposed by the authors and manipulates 1.3 million parameters is used to extract the Spatio-temporal characteristics of the 3D images by making combinations between CNN (for extraction of the features of the frames) and the LSTMs to capture the inconsistencies between frames.
The input of this model takes the following dimensionality for the 3D image size (128.128.16.3). After the tests, they obtained better results using an input as 16 straight faces.
Because of the test done on the two datasets mentioned above, they obtained superior results with a binary detection precision of 99.4 for high-quality datasets and 94.5 for low-quality datasets.

Article 3: Deepfakes Detection Techniques Using Deep Learning: A Survey [Alm21]

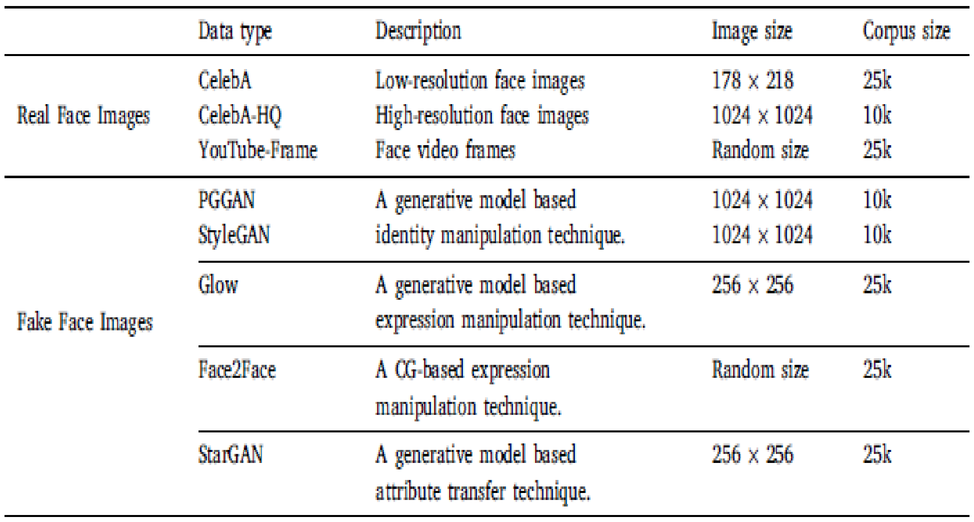
In this paper, the authors focus on providing a comprehensive study for deepfake detection using various deep-learning approaches such as long short-term memory (LSTM), recurrent neural network (RNN), Convolutional Neural Network (CNN), and even the hybrid approaches have been proposed.
To detect deepfakes, images, and videos, the authors divided these approaches into two major techniques: image detection techniques and video detection techniques. Which has well been divided into two main categories: biological singles analysis (such as (GAN) based model that can detect the deepfake video source by analyzing the "heartbeat" of deep fakes, which active an accuracy of 97.3%.), and spatial and temporal features analysis. Then they gave a detailed description of the architecture tool and performance for each method used in this study.
For deepfake image detection, a Hybrid approach was introduced, which uses a pairwise-learning. The approach first uses GANs to create and generate a fake image.
Then, on the popular fake feature network (CFFN) generated by GANs, a pairwise learning model captures the discriminated information between the fake and authentic images. Results show that this approach can overcome the shortcomings of the existing state-of-the-art fake image detectors.
For datasets, they mentioned the various publicly accessible datasets (like DFDC, DF-TIMIT, FaceForensics++) used in this domain, categorizing them by dataset sort, source, and method.

Article 4: Deepfake Detection Approaches Using Deep Learning: A Systematic Review [DW21]

This article offers a study of the tools and algorithms used to create and detect deepfakes and the strategies associated with deepfakes.
Presenting the ideologies of deepfake algorithms and how deep learning has been used to enable such technologies.
The most used tools, according to this article, are: DFaker, FaceSwap-GAN, Faceswap, deepfakerLab, DeepFaker-tk. Deepfake detection methods are divided into two main classes: false image detection approaches; they are based on deep learning like CNN, SVM, random forest, multilayer perceptron, and GAN, which produce images that are more difficult to notice.
The false Video Detection Approach is also divided into two groups: approaches use chronological features and approaches that explore visual artifacts.
Researchers have proposed the pipeline technique that uses CNN and LSTM to detect deepfake video from blinking eyes.
The proposed techniques are verified on the Face Forensics++data set. the UADFV and DeepfakeTIMIT datasets.
In the test's result, they found that general facial recognition systems like VGG and Facenet are unable to successfully detect deepfakes, and the use of SVM to measure image quality and lip synchronization generates a very high error rate.

Article 5: Fake face detection via adaptive manipulation traces extraction network [Guo+21]

In this article, the authors proposed a pre-processing module named AMTEN based on the convolution layers and designed to predict manipulation traces.
By integrating AMTEN and CNN, they constructed a robust fake face detector named AMTENnet which can learn discriminative features from manipulation traces.
They conducted a series of experiments to simulate the practical forensics under the complex scenario to evaluate the proposed module, using their hybrid fake face (HFF) dataset (Table 1). The result proved that the proposed AMTEN achieved desirable pre-processing as well the detector AMTENnet achieved accuracy up to 98.52%.



	Data type	Description	Image size	Corpus size
Real Face Images	CelebA	Low-resolution face images	178 x 218	25k
	CelebA-HQ	High-resolution face images	1024 x 1024	10k
	YouTube-Frame	Face video frames	Random size	25k
Fake Face Images	PGGAN	A generative model based	1024 x 1024	10k
	StyleGAN	identity manipulation technique.	1024 x 1024	10k
	Glow	A generative model based expression manipulation technique.	256 x 256	25k
	Face2Face	A CG-based expression manipulation technique.	Random size	25k
	StarGAN	A generative model based attribute transfer technique.	256 x 256	25k

TABLE 1.1: Details of HHF dataset.

Article 6: Perception matters: Exploring imperceptible and transferable anti-forensics for GAN-generated fake face imagery detection [Wan+21]

The researchers in this paper presented a study of the imperceptible and transferable anti-forensics on GAN-generated fake face imagery detection by proposing a novel adversarial attack method better suited to fake face imagery anti-forensics. The proposed method achieves higher adversarial transferability and improves the visual quality performance with 9.7% and 7.3% higher attack success rates on average on fake face imagery antiforensics.
For the Datasets, authors have created face image datasets for the fake face imagery detection using StyleGAN and StyleGAN2 datasets, respectively. With images resized to 128 ×128.
After various tests and experiments, they found that the proposed method raises additional security concerns to fake face imagery detection by fooling both deep learning and non-deep learning-based forensic detectors.

Article 7: Deepfakes and beyond: A Survey of face manipulation and fake detection [Tol+20]

The authors in this article have shown several methods of face manipulation and fake detection. Four techniques have been mentioned:

- **Face synthesis:** represents a manipulation that creates entire non-existent face images, usually through powerful GAN. The authors here proposed a fake detection system based on the analysis of the convolutional traces. They use classifiers such as k-Nearest Neighbors (k-NN), SVM, and Linear Discriminate Analysis (LDA). Their proposed approach was tested using fake images generated through several datasets (AttGAN, GDWCT, StarGAN, StyleGAN, and StyleGAN2), achieving a final 99.81% Accuracy for the best performance.

- **Identity Swap:** this manipulation consists of replacing one person's face in a video with another person's face. The authors proposed a temporal-aware pipeline to automatically detect fake videos considering a combination of CNNs and RNNs. Their proposed approach was evaluated using a proprietary database and achieved a final accuracy of 97.1%.

- **Attribute Manipulation:** represents a manipulation consisting of modifying some attributes of the face, also known as face editing or face retouching. Researchers proposed a detection system based on pixel co-occurrence matrices and CNN. Using the CelebA database for training, they achieved a final 99.4% accuracy for the best result.

- **Expression Swap:** this manipulation consists of modifying the person's facial expression, also known as face reenactment. The authors proposed an approach that is motivated as fake videos should have unnatural optical flow due to the unusual movement of lips, eyes, etc. Using the FaceForensics++ database, they obtained an Acc = 81.6% for the best performance in manipulation detection.

Article 8: Low-complexity fake face detection based on forensic similarity [PRZ21]

Researchers in this paper proposed a fake face detection framework based on the difference in similarity between the face and background area. By creating a new face forgery detection method, «the forensic similarity method,» to speed up training and inference speed, the authors proposed a method using a face forgery detection framework based on VGG19.
In the performance evaluation of the proposed method, they used two public face tampering video datasets: FaceForensics++ and Celeb-DF. Choosing the FaceForensics++ dataset for training on the C23 (Compression level 23) lightly compressed version and testing the proposed method on the four face tampering subsets (FF++/DF, F2F, FS, NT), While using the Celeb-DF dataset to evaluate the generalization capability of the method.
The evaluation results found that the proposed model has better or comparable performance with reduced parameters and lower complexity. In particular, it reduces the parameters by approximately 72% compared to Xception and achieved accuracy gains of 8-12% under the CELEB-DF dataset.

Article 9: Face image manipulation detection based on a convolutional neural network [Dan+19]

This study proposed an expert system that could identify whether an image is original or has been altered using a deep learning approach, including the Manipulated Face (MANFA)model. This is a customized convolutional neural network model, especially XGB-MANFA (eXtreme Gradient Boosting- MANFA), effectively detecting manipulated images.
Also, a hybrid framework (HF-MANFA) model is a robust framework for dealing with manipulated face detection in imbalanced dataset scenarios.
The experimental tests were carried out on two datasets: the "MANFA dataset, "which evaluates the computational time of different models, and the "SwapMe and FaceSwap" dataset to compare the proposed model with the state-of-the-art models.

These experiments were divided into different parts. They found that the primary purpose of the first experiment is to check the proposed model's performance in a balanced dataset scenario. In contrast, the second experiment validates different models' performance in the imbalanced dataset scenario.
The authors found that the proposed model can detect images edited manually by a human or automatically by a computer. Therefore, it also plays a significant role in digital image security, which surpassed the best-known result by approximately 6%, with an area under the curve (AUC) that surpassed 93.4% in classification results.

Article 10: DeepFakes: a New Threat to Face Recognition? Assessment and Detection [KM18]

In this paper, the authors presented the first publicly available database containing 620 deepfake videos split on training and evaluating subsets with high-quality (128x128) size and low quality (64 x 64 input/output) Using software on GAN approach.
The deepfake that's been generated in this paper can effectively imitate the facial expressions as well as the mouth movement and the blinking of the eyes. So far, in detecting deepfake they used an audiovisual approach that detects the inconsistency between visual lip movements and speech in audio and then applied several baseline methods.
By evaluating face VGG and Facenet-based recognition systems, they found they are vulnerable to deepfake videos, with 85.62% for VGG and a 95% false acceptance rate for Facenet.
As a result, they found that the technique based on image quality measures with an SVM classifier is the best performing method for the detection of deepfake with an 8.97% equal error rate.

8 Conclusion

Deep learning has been rapidly developing, creating new technologies, Including Deepfakes, that have become popular because of the massive availability of images and videos in social content. The creation of deepfake videos can be harmful and have malicious purposes for public and private people as it is challenging to be detected by naked eyes. Also, it becomes more accessible for amateur use; that is why deepfake detection systems are becoming more and more obligatory to reduce the spreading of misinformation and misunderstanding caused by these fake videos and images.

CHAPTER 2 CONVOLUTIONAL NEURAL NETWORKS

1 Introduction :

The lifestyle of modern society has changed significantly in recent years with the rapid expansion of artificial intelligence (AI), machine learning (ML), and deep learning (DL) technologies that impact nearly every technological aspect of society.
Neural Network is a machine learning (ML) technique that is inspired by the sophisticated functionality of human brains and resembles the human nervous system and the structure of the brain. It can be used in a variety of problems. This chapter presents a group of deep learning techniques that are based on Neural Network technology.

2 Machine Learning

Machine learning is the technology of developing computer algorithms that can emulate human intelligence. It draws on ideas from different disciplines, such as artificial intelligence, probability, statistics, computer science, information theory, etc. This technology has been applied in such diverse fields as pattern recognition, computer vision, spacecraft engineering, etc.[ENM15]
A machine learning algorithm is a computational process that uses input data to achieve the desired task without being programmed to produce a particular outcome. The most important property of these algorithms is their distinctive ability to learn the surrounding environment from input data with or without a teacher. [ENM15]

3 Machine Learning Approaches

Machine learning can be divided according to the data labeling into four types of algorithms supervised, unsupervised, semi-supervised, and reinforcement learning. [ENM15]

3.1 Supervised learning

is used to estimate an unknown (input, output) mapping from known (input, output) samples, where the output is labelled. Supervised learning has two parts. One is called classification, and the other is called regression.

a) Regression

Used to predict continuous values like stock price and home price in a specific city. Standard algorithms are linear regression, Support Vector Machines, Multivariate Regression algorithms, etc.[Vas+19]

b) Classification

Used to predict boolean values like True/False or Male/Female. Standard algorithms are Naive Bayes, Decision Trees, Support Vector Machines (SVM), Decision Tree, Random Forest, etc.[Vas+19]

3.2 Unsupervised learning

Only input samples are given to the learning system (e.g., clustering and estimation of probability density function).[Vas+19]

3.3 Semi-supervised Learning

learning is a combination of both supervised and unsupervised, where part of the data is partially labeled. The labeled part is used to infer the unlabeled portion (e.g., text/image retrieval systems).[w2]

3.4 Reinforcement learning

In artificial intelligence, reinforcement learning is a type of dynamic programming that trains algorithms using a system of reward and punishment. A reinforcement learning algorithm, or agent, learns by interacting with its environment. The agent receives rewards for performing correctly and penalties for performing incorrectly. The agent learns without intervention from a human by maximizing its reward and minimizing its penalty. (e.g. playing games and learning from every move that it was correct or not).[Vas+19]

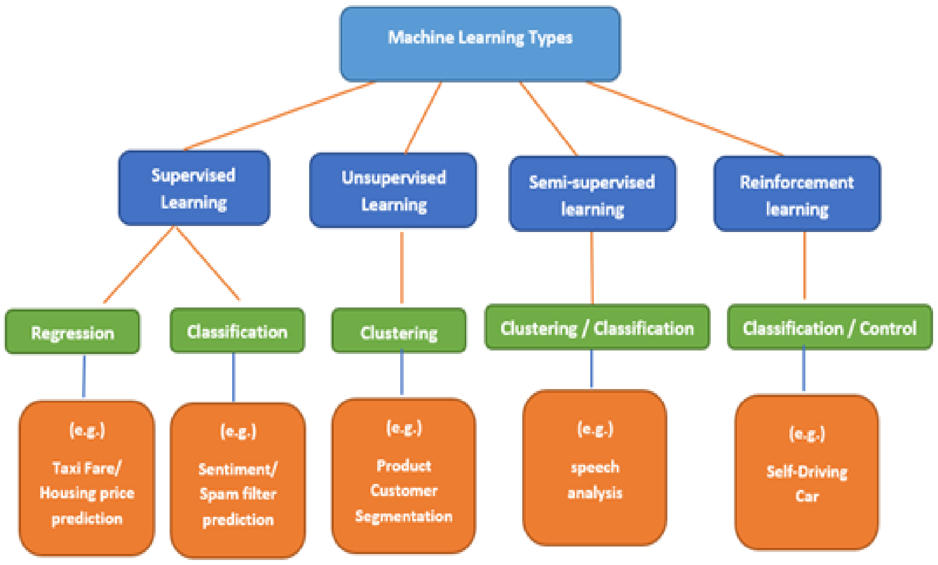


FIGURE 2.1: The four different Machine Learning algorithms

4 Basics of Artificial Neural Networks (ANNs)

The basic concept of Artificial Neural Networks (ANNs) is partially inspired by how the human brain functions. Figure 1 shows artificial neural network architecture. Neural networks are multilayer networks that consist of a single input layer, one or multi hidden layers, and one output layer. The input to neural networks is a set of input values. The goal of neural networks is to predict and classify those values into pre-defined categories.[Alm21]

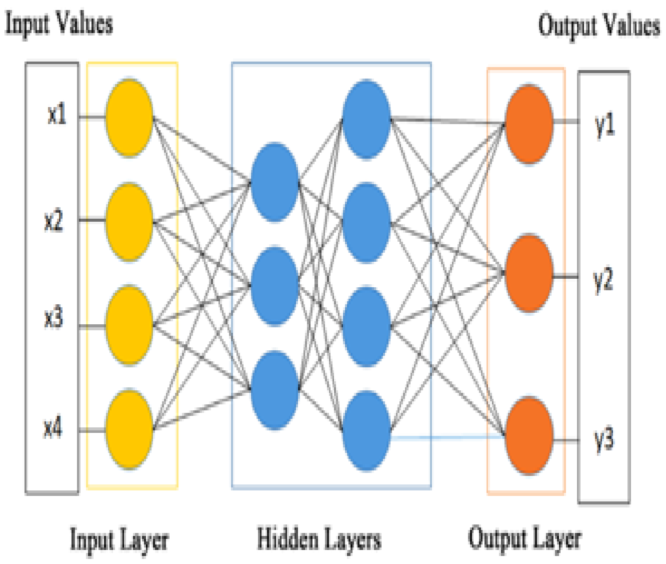


FIGURE 2.2: Artificial neural networks architecture.

5 Deep Learning

Deep learning is a sub-field of machine learning to deal with algorithms that are mostly based on specific types of artificial neural networks, sometimes with a high number of layers and nodes. It mirrors the functioning of human brains. Deep learning algorithms are similar to how the nervous system is structured where each neuron is connected with the other and passes information. Deep learning models work in layers, and a typical model at least has three layers. Each layer accepts the information from the previous and passes it on to the next one.
The Difference between machine learning and deep learning model is in the feature extraction area. Feature extraction is done by a human in machine learning, whereas deep learning models figure it out by themselves.
Deep learning models perform well with the amount of data, whereas old machine learning models stop improving after a saturation point.Thus, deep learning is a specific type of machine learning, part of AI.[Vas+19]

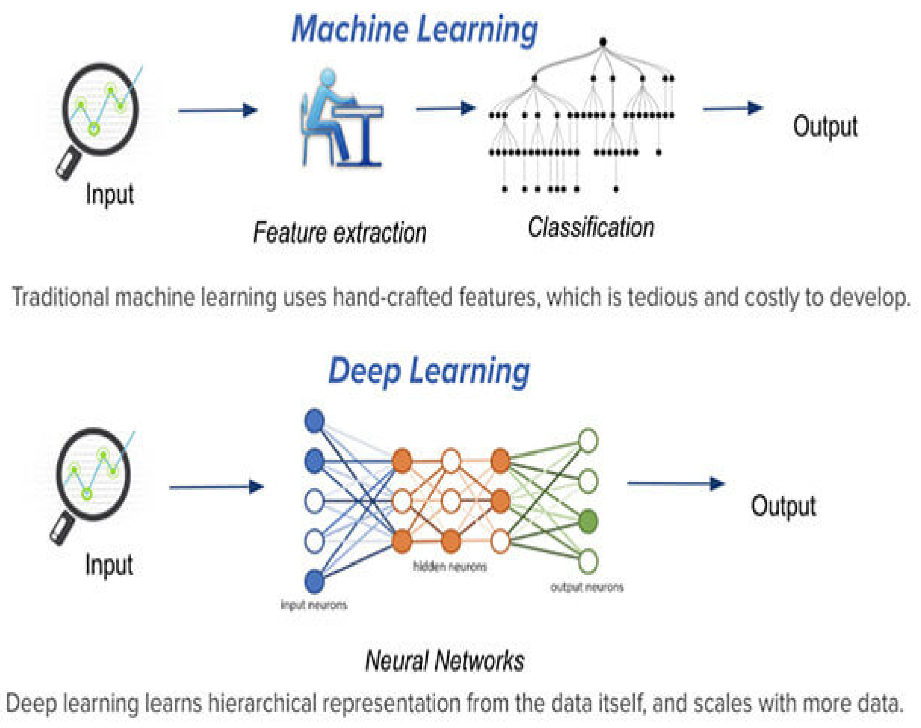

Machine Learning
Input
Feature extraction
Classification
Output
Traditional machine learning uses hand-crafted features, which is tedious and costly to develop.
Deep Learning
Input
input neurons
hidden neurons
output neurons
Output
Neural Networks
Deep learning learns hierarchical representation from the data itself, and scales with more data.

FIGURE 2.3: between machine learning and deep learning.

6 Difference between neural networks and deep learning neural networks

Neural networks can use either feed-forward or recurrent networks with one or two hidden layers. However, when the number of hidden layers increases, i.e., over two, it is known as the Deep Learning Neural Network.[**pyle2015executive**]

- Neural Network is less complicated and requires more information about feature selection and feature engineering methods.

- Deep Learning Neural Network does not need any information about features; instead, they show optimum model tuning and model selection independently

7 Deep Learning techniques

Different techniques of Deep Learning are described below:

7.1 Fully Connected Neural Networks

Fully Connected Neural Networks it is often identified by their multilayer perceptrons.It consists of fully connected layers that connect every neuron in one layer to every neuron in the other layer. The significant advantage of fully connected networks is that they are "structure agnostic," i.e. there are no particular assumptions needed to be made about the input.While being structure agnostic makes fully connected networks very broadly applicable, such networks tend to have weaker performance than special-purpose networks tuned to the structure of a problem space.[Sai+15].
Works best in

- Any table dataset which has rows and columns formatted in CSV.
- Classification and regression issues with the input of real values.
- Any model with the highest flexibility, like that of ANNS

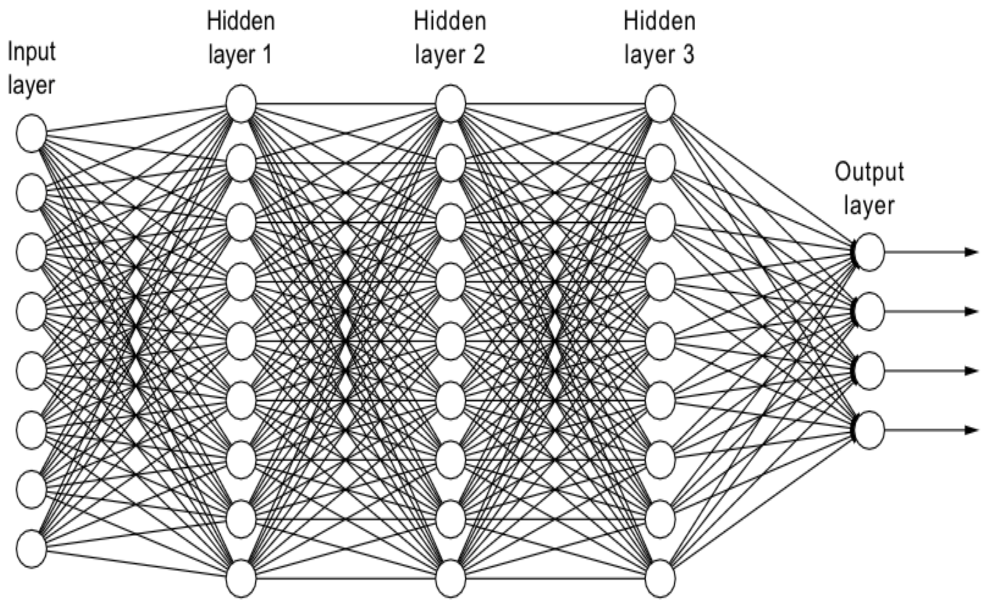


FIGURE 2.4: Fully Connected Neural Networks architecture.

7.2 Recurrent Neural Network (RNN)

A recurrent Neural Network (RNN) is another application of artificial neural networks capable of learning features from sequence data. Similar to neural networks, RNN is made up of several invisible layers, each of which has a weight and a bias. In RNN, the relations between nodes in a direct cycle graph run in sequential order. One advantage of RRN is that it allows the discovery of temporal dynamic behavior. Compared with feedforward networks (FFN), RNN uses internal memory to store the sequence information from previous inputs, which makes it useful in various areas, including natural language analysis and speech recognition. RNN can handle a temporal sequence by introducing a recurrent hidden state, which captures dependencies of different time scales.[Alm21]

- One to One: A single input connected to a single output, like Image classification.
- One to many: A single input linked to output sequences, like image captioning, includes several words from a single image.
- Many to One: Series of inputs generating single output, like Sentiment Analysis.
- Many to many: series of inputs yielding series of outputs, like video classification.
- It is also widely used in language translation, conversation modeling.

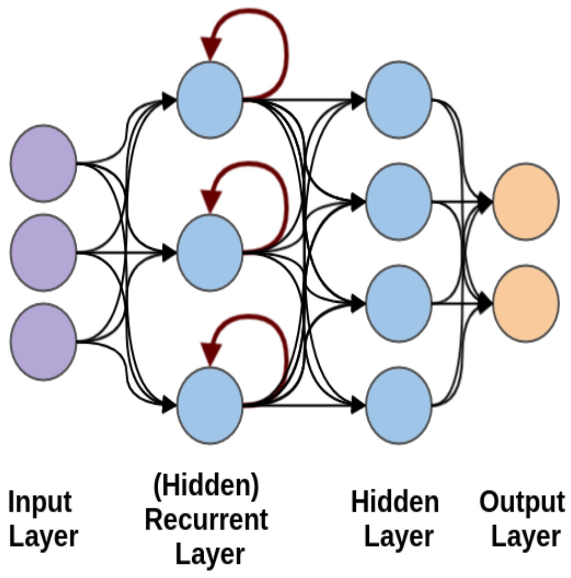


FIGURE 2.5: Recurrent Neural Networks architecture.

7.3 Long Short-Term Memory (LSTM)

LSTM is an artificial recurrent neural network (RNN) that handles long-term dependencies. LSMT contains feedback connections to learn the entire sequence of data. The typical architecture of LSTM consists of an input gate, forget gate, and output gate. The cell state is a long-term memory that remembers values from previous intervals and stores them in the LSTM cell.[Alm21]

- First, the input gate is responsible for selecting the values that should enter the cell state.
- The forget gate is reasonable for determining which information is to forget by applying a sigmoid function, which has a range of [0, 1].
- The output gate determines which information in the current time should be considered in the next step.

It has been applied to many fields based on time-series data, such as classifying, processing, and making predictions.

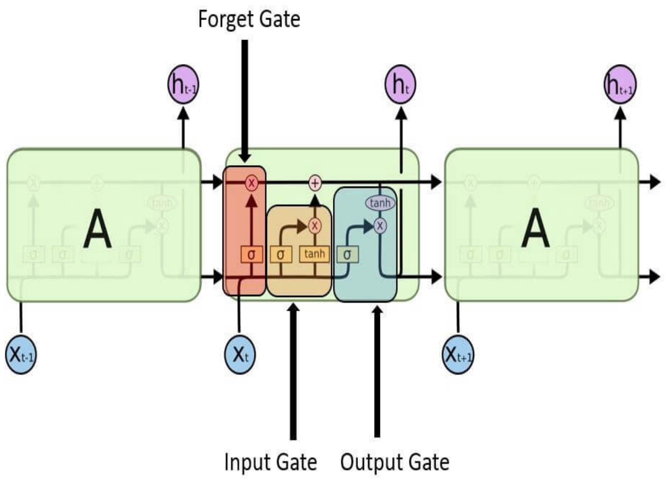


FIGURE 2.6: Long Short Term Memory architecture.

7.4 Convolutional Neural Network (CNN)

A convolutional neural network is a feed-forward neural network that is used to analyze visual images by processing data with a grid-like topology. It is also known as a ConvNet. A convolutional neural network is used to detect and classify objects in an image.[AMAZ17]
A convolution neural network has multiple hidden layers that help extract information from an image. The four important layers on CNN are:

- Convolution layer.
- ReLU layer.
- Pooling layer.
- Fully connected layer.

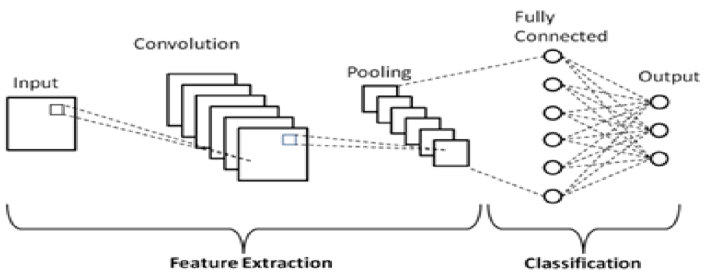
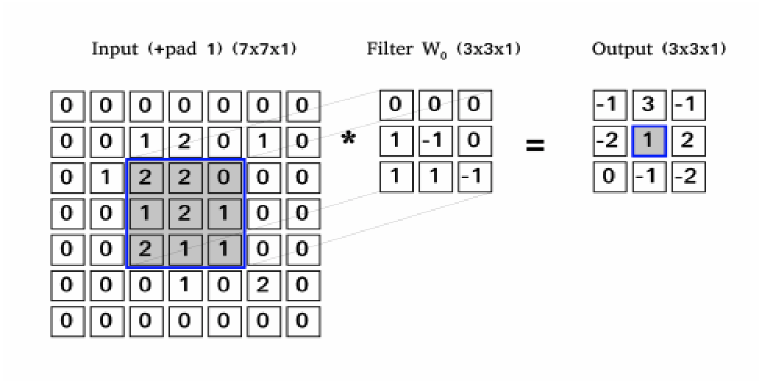


FIGURE 2.7: Convolutional Neural Network architecture.

a. Convolution layer
This is the core layer of the convolutional neural network. Its parameters are composed of a set of filters. These filters are small, but they cover the full depth of the input volume.
The main task performed at the convolutional layer is to extract high-level features. The first one (as shown in the Figure 2.7) extracts low-level features like color, edges, etc. The subsequent convolutional layers take out the high-level features, thus leading to a complete understanding/ perusal of the image.[Wu17]



$$\begin{bmatrix}0&0&0&0&0&0&0\\0&0&1&2&0&1&0\\0&1&2&2&0&0&0\\0&0&1&2&1&0&0\\0&0&2&1&1&0&0\\0&0&0&1&0&2&0\\0&0&0&0&0&0&0\end{bmatrix} * \begin{bmatrix}0&0&0\\1&-1&0\\1&1&-1\end{bmatrix} = \begin{bmatrix}-1&3&-1\\-2&1&2\\0&-1&-2\end{bmatrix}$$

FIGURE 2.8: Filter Matrix.

b. ReLU layer
ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer. ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map. Figure 2.9 represent the graph of ReLU function with other activations functions such as correction by hyperbolic tangent 'tanh' and the correction by the 'sigmoid' function, etc:[AMAZ17]

c. Pooling layer
This layer reduces the spatial size of the image representation. It also helps to reduce the computation and processing amount in the neural network.It also extracts dominant features that are positionally and rotationally invariant.
There most popular pooling are: [Wu17]

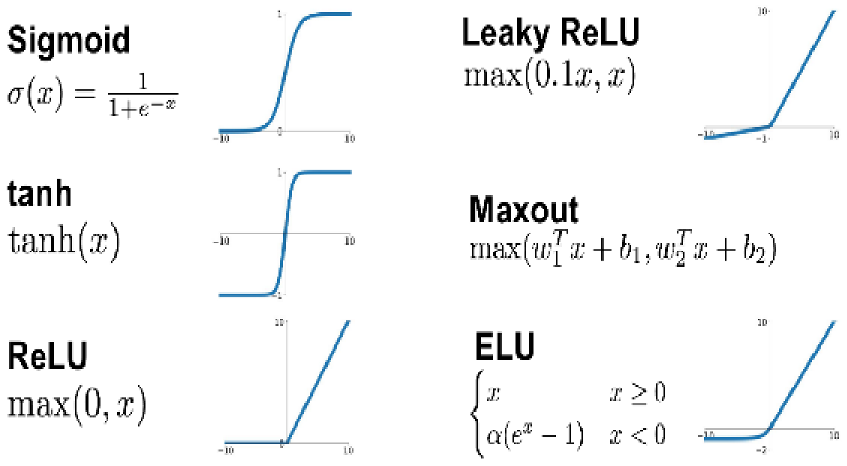


FIGURE 2.9: Activation functions graph.

- **Max pooling operation:** This operation picks the maximum value from each neuron cluster at the last layer.

- **Average pooling:** returns an average value from the cluster.

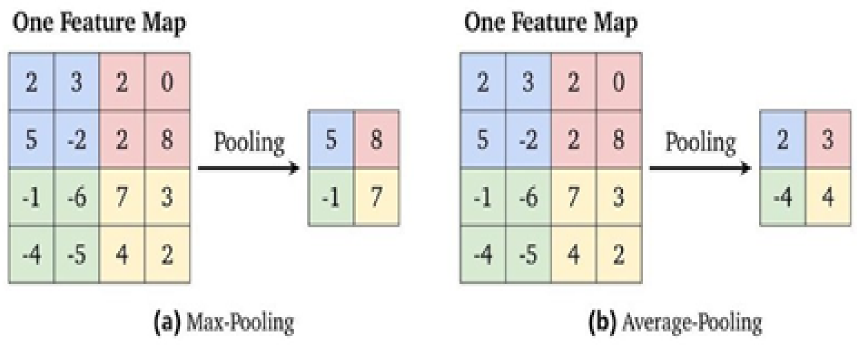


FIGURE 2.10: Max pooling and Average pooling.

Since Max pooling also acts as a noise suppressant, it performs better-than-average pooling.

d. The Fully Connected Layers (FCL) [Wu17]
The Fully Connected layer is simply a feed-forward neural network. The input to the fully connected layer is the flattened output of the last pooling /convolutional layer. To flatten means that the 3-dimensional matrix or array is unrolled into a vector.

For each FC layer, a specific mathematical calculation takes place. After the vector has passed through all the fully connected layers, an activation function is used in the last layer. This is used to compute the probability of the input belonging to a particular task.

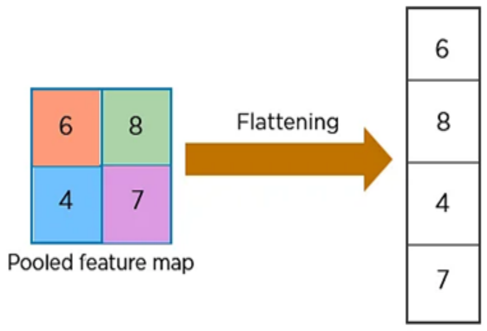


FIGURE 2.11: Flattening operation.

Thus, the result is the different probabilities of the input image belonging to different classes.
The process is repeated for different types of images and individual images within those types. This trains the network and teaches it to differentiate between different objects.

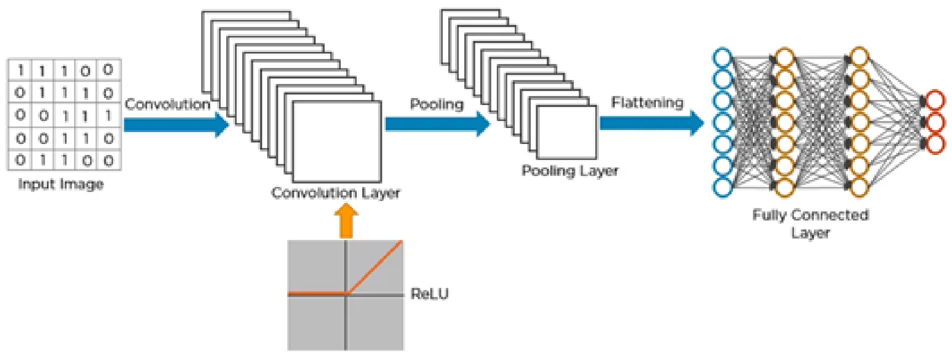


FIGURE 2.12: Convolution neural network process.

8 Networks used in Deepfakes

8.1 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are an emerging technique for both semisupervised and unsupervised learning. They achieve this through implicitly modeling high-dimensional distributions of data. They can be characterized by training a pair of networks in competition with each other. A common analogy, apt for visual data, is to think of one network as an art forger and the other as an art expert. The forger, known in the GAN literature as the generator, G, creates forgeries, with the aim of making realistic images. The expert, known as the discriminator, D, receives both forgeries and real (authentic) images, and aims to tell them apart. Both are trained simultaneously, and in competition with each other.[Cre+18] *a) Generator*

A generator in GANs is a neural network that creates fake data to be trained on the Discriminator. It learns to generate plausible data. The generated examples/instances become negative training examples for the Discriminator. It takes a fixed-length random vector carrying noise as input and generates a sample.[Vas+19]

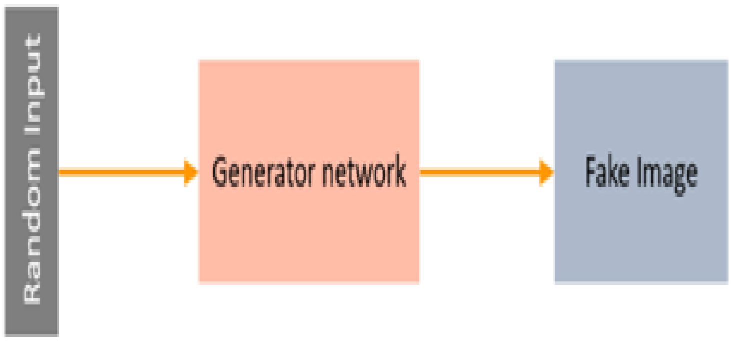


FIGURE 2.13: Generator network.

The main aim of the Generator is to make the Discriminator classify its output as real. The part of the GAN that trains the Generator includes:

- Noisy input vector.
- Generator network which transforms the random input into a data instance.
- Discriminator network, which classifies the generated data.
- Generator loss, which penalizes the Generator for failing to dolt the Discriminator.

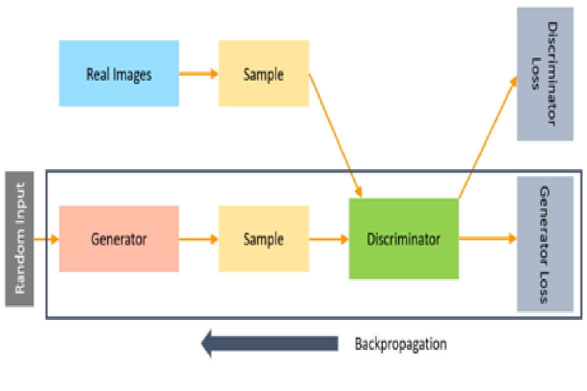


FIGURE 2.14: Generator Training.

The back-propagation method is used to adjust each weight in the right direction by calculating the weight's impact on the output. It is also used to obtain gradients, and these gradients can help change the generator weights.[Vas+19] *b) Discriminator*

The Discriminator is a neural network that identifies real data from the fake data created by the Generator. The Discriminator's training data comes from different two sources [Vas+19]:

- **Real data:** instances, such as real pictures of people. The Discriminator uses these instances as positive examples during training.

- **Fake data:** instances created by the Generator. The Discriminator uses these instances as negative examples during training.

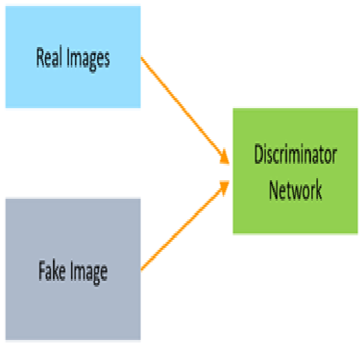


FIGURE 2.15: Discriminator Network.

8.1.1 Training the Discriminator

The Discriminator connects to two loss functions. During discriminator training, the Discriminator ignores the generator loss and uses the discriminator loss. The generator loss is used during generator training.[Vas+19]
During discriminator training:

- The Discriminator classifies both real data and fake data from the Generator.
- The discriminator loss penalizes the Discriminator for misclassifying a real instance as fake or a fake instance as real.
- The Discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network.

8.1.2 Steps for Training GAN

- Define the problem.
- Choose the architecture of GAN.

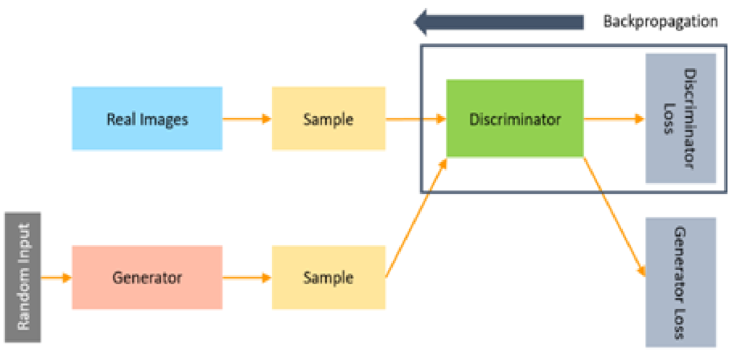


FIGURE 2.16: Discriminator Training.

- Train discriminator on real data.
- Generate fake inputs for the Generator.
- Train discriminator on fake data.
- Train generator with the output of the Discriminator.

8.2 Autoencoders

Auto-encoders are a specific type of feedforward neural network that is trained to copy its input to its output. They compress the input into a lower-dimensional code and then reconstruct the output from this representation. The code is a compact “summary” or “compression” of the input, also called the latent-space representation.[w4]
An auto-encoder consists of three main components [w4]:

a. Encoder: A module that compresses the input data into an encoded representation that is typically smaller than the input data producing the code.

b. Code: A module that contains the compressed data representations and is therefore the most important part of the network.

c. Decoder: A module that helps the network “decompress” and "reconstruct" the data back from its encoded form. The output is then compared with the original input.
The whole architecture is represented in figure 2.17.

Autoencoders are considered an unsupervised learning technique since they don’t need explicit labels to train on. But to be more precise, they are self-supervised because they generate their labels from the training data.[w4]

8.2.1 Autoencoders Architecture

Auto-encoder consists of a network of encoders and decoders. During training, both encoders and decoders are used. Like any deep learning (DL) architecture, AE works in layers of neurons and is trained using backpropagation.[SSR20]

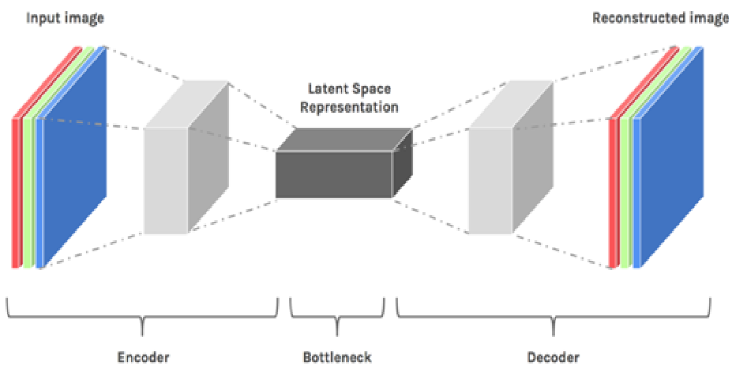


FIGURE 2.17: Autoencoders architecture.

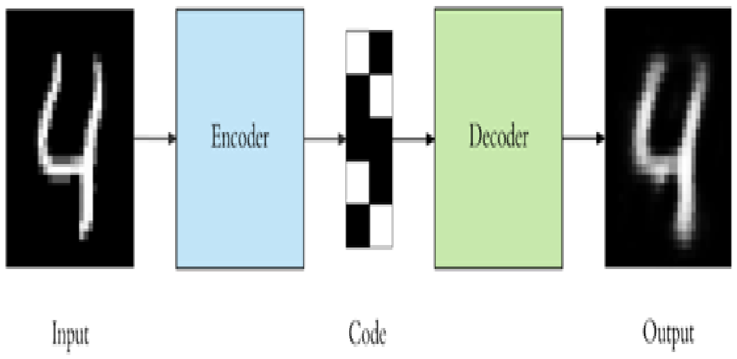


FIGURE 2.18: Autoencoders example.

- The layers are divided into different encoder and decoder layers.
- The input is connected to the first encoder layer. In each subsequent layer of the encoder, the number of neurons is reduced till we reach the last encoded layer, which has the least number of neurons, representing the bottleneck features.
- The inputs are transformed until this layer represents the reduced dimension of the data that is capable of representing the maximal signals in the data.
- After this layer, the decoder layers are set, in which the number of neurons is increased in each layer until the output layer has the same number of neurons as the input.

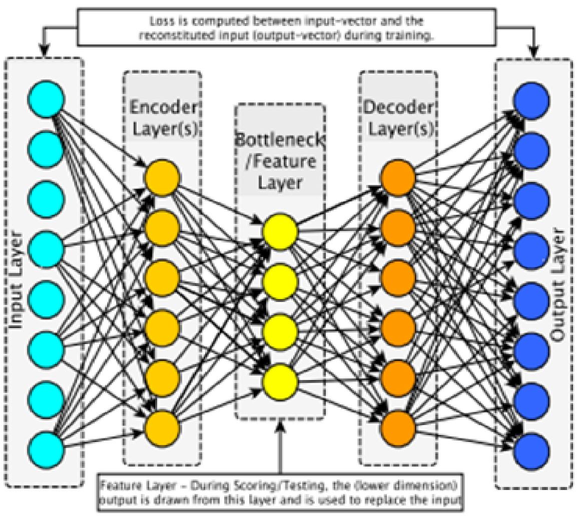

Loss is computed between input-vector and the reconstituted input (output-vector) during training.
Input Layer
Encoder Layer(s)
Bottleneck /Feature Layer
Decoder Layer(s)
Output Layer
Feature Layer - During Scoring/Testing, the (lower dimension) output is drawn from this layer and is used to replace the input

FIGURE 2.19: Schematic of an Autoencoders.

The figure 2.19 shows an encoder connected to the input layer leading to bottleneck layer/ features, followed by a decoder leading to the reconstituted output layer.

8.2.2 Training Autoencoders

We train a mono-layer neural network, and the hidden value h is predicted by the output z of the input x. This is a task called reconstruction (see figure 2.20). Therefore, the criterion is to minimize the reconstruction error L(x;z). Therefore, the hidden layer must contain information relating to the reconstruction.
For example, if the hidden layer contains fewer units than the input, so to properly reconstruct, it must learn to summarize the entry, as all of the necessary information is contained in the hidden layer. As a result, characteristics relevant to the reconstruction must be extracted.

Four hyper-parameters should be set before training an auto-encoder [w4]:

1) Code size:

The code size or the size of the bottleneck, it is the most important hyper-parameter used to tune the auto-encoder. The bottleneck size decides how much the data has to be compressed. This can also act as a regularisation term.

2) Number of layers:

Like all neural networks, an important hyper-parameter to tune auto-encoders is the depth of the encoder and the decoder. While a higher depth increases model complexity, a lower depth is faster to process.

3) Number of nodes per layer:

The number of nodes per layer defines the weights we use per layer. Typically, the nodes number decreases with each subsequent layer in the auto-encoder as the input to each layer becomes smaller across the layers.

4) Reconstruction Loss:

The loss function we used to train the auto-encoder depends on the type of input and output we want the auto-encoder to adapt. The most popular loss functions for the reconstruction of image data are mean squared error (MSE) Loss and L1 Loss. If the inputs and outputs are within the range [0,1], as in MNIST [Den12] (DATABASE of handwritten digits), Binary Cross Entropy can also be used as the reconstruction loss.

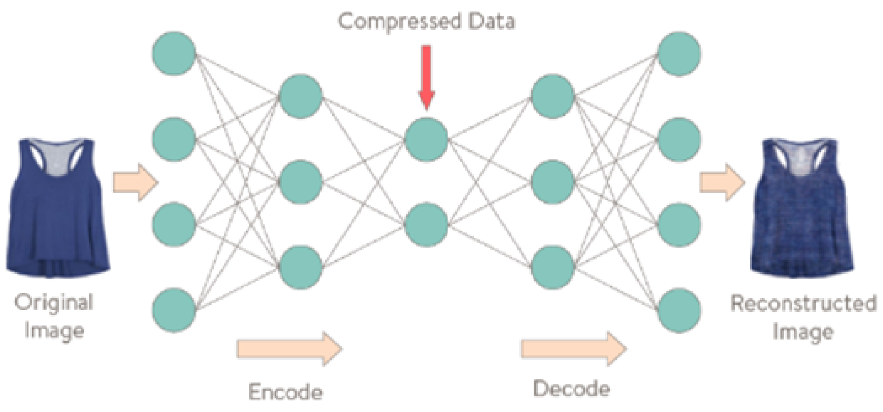


FIGURE 2.20: Principle of autoencoder on image.

8.2.3 Types of Autoencoders

Autoencoders encodes the input values x using a function f. Then decodes the encoded values f(x) using a function "g" to create output values identical to the input values.
There are many types of auto-encoder, ranging from simple to denoising auto-encoders used for various purposes. Here are the popular auto-encoders:

- Undercomplete autoencoders.
- Sparse autoencoders.
- Contractive autoencoders.
- Denoising autoencoders.
- Variational autoencoders (for generative modeling).
- Deep convolutional autoencoder.

a. Undercomplete Autoencoder

Undercomplete autoencoder is one of the simplest types of autoencoders. It is an autoencoder whose code dimension is less than the input dimension.

Learning an undercomplete representation forces the autoencoder to capture the most salient features of the training data.
The way it works is very straightforward, Undercomplete autoencoder takes in an image and tries to predict the same image as output, thus reconstructing the image from the compressed bottleneck (code) region.[BKG20]

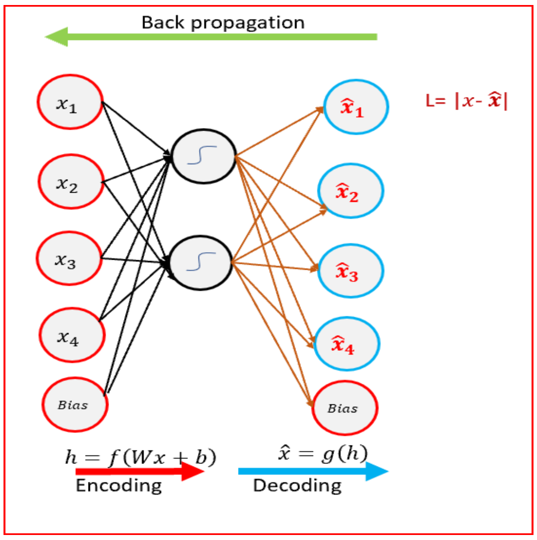


FIGURE 2.21: Undercomplete Autoencoder- Hidden layer has smaller dimension than input layer.

Objective is to minimize the loss function by penalizing the g(f(x)) for being different from the input x.

$$\mathbf{L} = |\mathbf{x} + \hat{x}| \quad (2.1)$$

$$\mathbf{L} = |\mathbf{x} - \mathbf{g}(\mathbf{f}(\mathbf{x}))| \quad (2.2)$$

b. Sparse Autoencoder

Sparse autoencoders are similar to the undercomplete autoencoders in that they use the same image as input and ground truth. However, the means by which the encoding of information is regulated is significantly different. The sparse autoencoder is regulated by changing the number of nodes at each hidden layer.[BKG20]
A sparse autoencoder is where hidden layers outnumber the input layer for the generalization approach to reduce overfitting. It limits the loss function and prevents the autoencoder from overusing all its nodes.[Vas+19]
Sparse autoencoders have a sparsity penalty, (h), a value close to zero but not zero. Sparsity penalty is applied on the hidden layer in addition to the reconstruction error. This prevents overfitting.[GBC16]

$$\mathbf{L} = |\mathbf{x} - \mathbf{g}(\mathbf{f}(\mathbf{x}))| + \Omega(h) \quad (2.3)$$

Sparse autoencoders take the highest activation values in the hidden layer and zero out the rest of the hidden nodes. This prevents autoencoders to use all of the hidden nodes at a time and forcing only a reduced number of hidden nodes to be used.As we activate and inactivate hidden nodes for each row in the dataset. Each hidden node extracts a feature from the data.[GBC16]

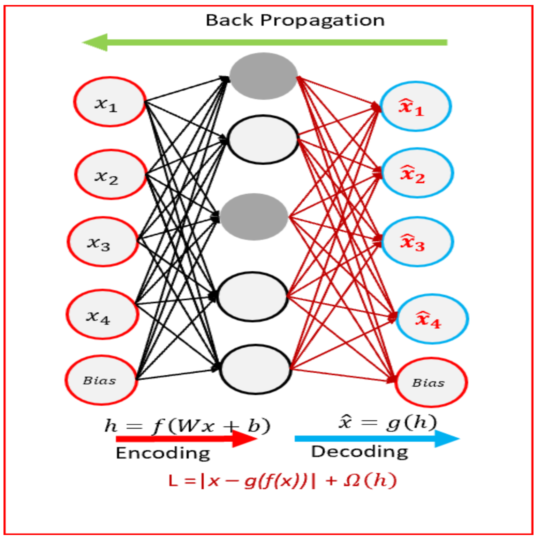


FIGURE 2.22: Sparse Autoencoders use only reduced number of hidden nodes at a time.

c. Contractive autoencoders

Similar to other autoencoders, contractive autoencoders perform task of learning a representation of the image while passing it through a bottleneck and reconstructing it in the decoder.
The contractive autoencoder also has a regularization term to prevent the network from learning the identity function and mapping input into the output.
Contractive autoencoders work on the basis that similar inputs should have similar encoding and a similar latent space representation. It means that the latent space should not vary by a considerable amount for minor variations in the input[SVM+11].

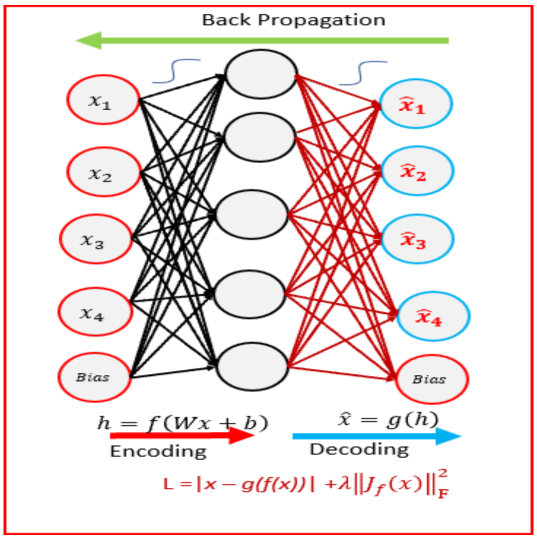


FIGURE 2.23: Contractive Autoencoders.

Robustness of the representation for the data is done by applying a penalty term to the loss function. The penalty term is Frobenius norm of the Jacobian matrix. Frobenius norm of the Jacobian matrix for the hidden layer is calculated with respect to

input. Frobenius norm of the Jacobian matrix is the sum of square of all elements. Contractive autoencoder is another regularization technique like sparse autoencoders and denoising autoencoders. CAE surpasses results obtained by regularizing autoencoder using weight decay or by denoising. CAE is a better choice than denoising autoencoder to learn useful feature extraction.
Penalty term generates mapping which are strongly contracting the data and hence the name contractive autoencoder.[GBC16]

d. Denoising Autoencoder

The denoising autoencoder (DAE) is an autoencoder that receives a corrupted data point as input and is trained to predict the original, uncorrupted data point as its output. Autoencoders that remove noise from an image.[GBC16]

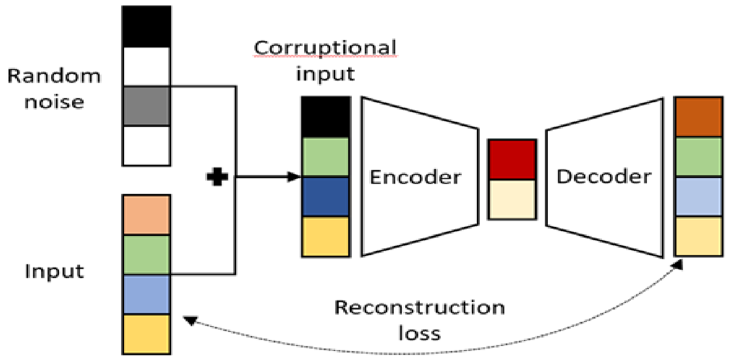


FIGURE 2.24: Denoising Autoencoders Princple.

In denoising autoencoders, they feed a noisy version of the image, where noise has been added via digital alterations. The noisy image is fed to the encoder-decoder architecture, and the output is compared with the ground truth image.[GBC16]
The denoising autoencoder removes noise by learning a representation of the input where the noise can be filtered out.

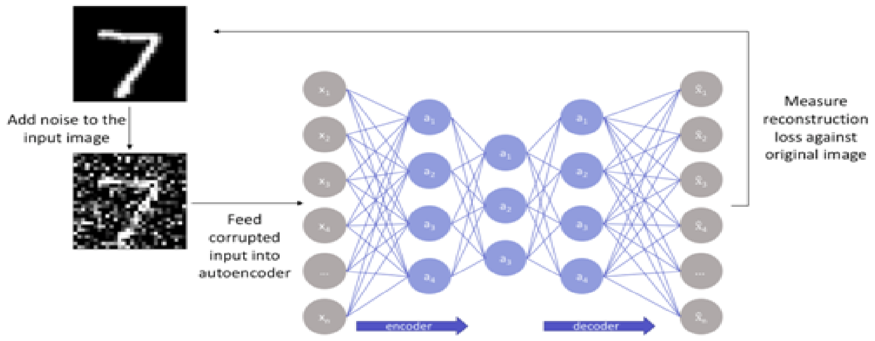

Add noise to the input image
Feed corrupted input into autoencoder
Measure reconstruction loss against original image
encoder
decoder

FIGURE 2.25: Denoising Autoencoder example.

e. Variational Autoencoder

Variational autoencoder can be defined as an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process.

The VAE can be viewed as two coupled, but independently parameterized models: the encoder or recognition model, and the decoder or generative model. These two models support each other. The recognition model delivers to the generative model an approximation to its posterior over latent random variables, which it needs to update its parameters inside an iteration of "expectation maximization" learning. Reversely, the generative model is a scaffolding of sorts for the recognition model to learn meaningful representations of the data, including possibly class-labels. The recognition model is the approximate inverse of the generative model according to Bayes rule.[KW19]

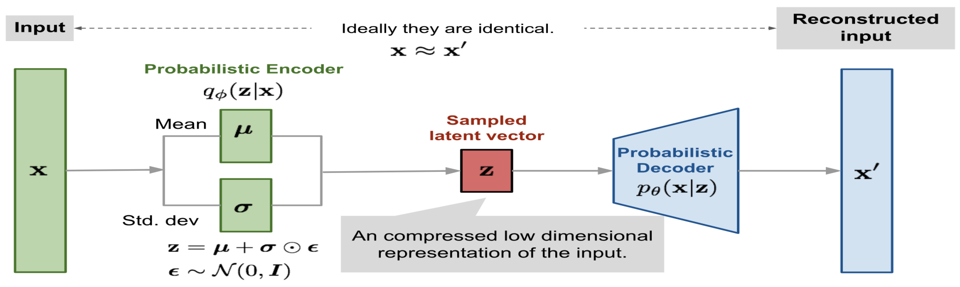

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{x}$
Mean
$\boldsymbol{\mu}$
Std. dev
$\boldsymbol{\sigma}$
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

FIGURE 2.26: Variational Autoencoder Architecture.

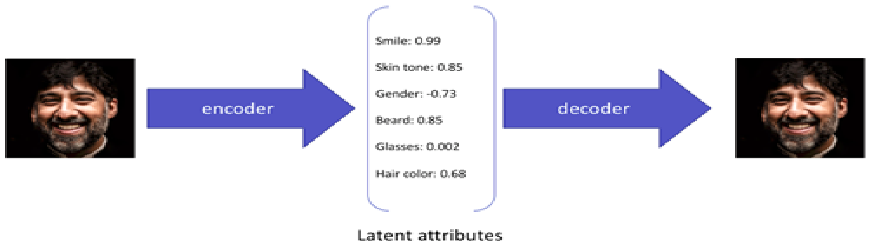


FIGURE 2.27: Variational Autoencoder example.

f. Deep Convolutional Autoencoders

Convolutional Autoencoder is a variant of Convolutional Neural Networks that are used as the tools for unsupervised learning of convolution filters. They are generally applied in the task of image reconstruction to minimize reconstruction errors by learning the optimal filters. Once they are trained in this task, they can be applied to any input in order to extract features. Convolutional Autoencoders are general-purpose feature extractors differently from general autoencoders that completely ignore the 2D image structure. In autoencoders, the image must be unrolled into a single vector and the network must be built following the constraint on the number of inputs [w5].
The block diagram of a Convolutional Autoencoder is given in figure 2.28.

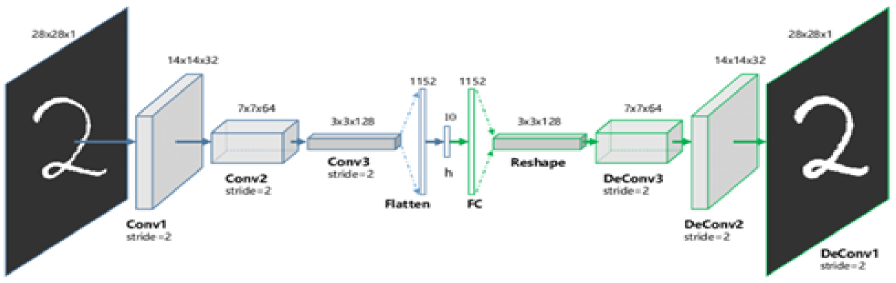


FIGURE 2.28: The structure of Convolutional AutoEncoder.

8.2.4 Application of Autoencoders

- **Dimensionality reduction:**
  Undercomplete autoencoders are those that are used for dimensionality reduction. These can be used as a pre-processing step for dimensionality reduction as they can perform fast and accurate dimensionality reductions without losing much information. Furthermore, while dimensionality reduction procedures like PCA can only perform linear dimensionality reductions, undercomplete autoencoders can perform large-scale non-linear dimensionality reductions.[BKG20] .

- **Image denoising:**
  Autoencoders like the denoising autoencoder can be used for performing efficient and highly accurate image denoising. Unlike traditional methods of denoising, autoencoders do not search for noise, they extract the image from the noisy data that has been fed to them via learning a representation of it. The representation is then decompressed to form a noise-free image. Denoising autoencoders thus can denoise complex images that cannot be denoised via traditional methods.[BKG20]

- **Generation of image and time-series data:**
  Variational Autoencoders can be used to generate both image and time series data. The parameterized distribution at the bottleneck of the autoencoder can be randomly sampled to generate discrete values for latent attributes, which can then be forwarded to the decoder,leading to generation of image data. VAEs can also be used to model time series data like music.[BKG20]

- **Anomaly Detection:**
  Anomaly detection is another unsupervised task, where the objective is to learn a normal profile given only the normal data examples and then identify the samples not conforming to the normal profile as anomalies. This can be applied in different applications such as fraud detection, system monitoring, etc. [BKG20]

9 Performance Metrics

To evaluate a network, it is necessary to calculate a certain number of parameters such as:
True Positive (TP): The cases predicted YES and the actual output was also YES.
True Negative (TN): The cases predicted NO and the actual output was NO.

False Positive (FP): The cases predicted YES and the actual output was NO.
False Negative (FN): The cases predicted NO and the actual output was YES.

9.1 Confusion Matrix

Confusion matrix is a measure used while solving classification problems. It can be applied to binary classification as well as for multiclass classification problems.

		Predicted classes	
		class = Positive	class = Negative
Actual classes	class = positive	TP	FN
	class = Negative	FP	TN

TABLE 2.1: Confusion Matrix.

9.2 Accuracy

Accuracy can be calculated as follows

$$\textbf{Accuracy} = \frac{\textbf{TP}}{\textbf{TP+FP}} \tag{2.4}$$

9.3 True Positive Rate/ Recall/ Sensitivity

A Recall is essentially the ratio of true positives to all the positives in ground truth.

$$\textbf{Recall} = \frac{\textbf{TP}}{\textbf{TP+FN}} \tag{2.5}$$

9.4 True Negative Rate

Probability of a negative prediction in a negative case.

$$\textbf{True Negative Rate} = 1 - \frac{\textbf{TN}}{\textbf{TN+FP}} \tag{2.6}$$

9.5 False Positive Rate

$$\textbf{False Positive Rate} = 1 - \frac{\textbf{TN}}{\textbf{TN+FP}} \tag{2.7}$$

9.6 False Negative Rate

$$\textbf{False Negative Rate} = 1 - \frac{\textbf{TP}}{\textbf{TN+FP}} \tag{2.8}$$

9.7 F-Measure

The F-measure is defined as a harmonic mean of precision (P) and recall (R).

$$\textbf{F1-Mesure} = 2 \times \frac{\textbf{precision * recall}}{\textbf{precision+recall}} \tag{2.9}$$

9.8 ROC curve

ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds [w6]. This curve plots two parameters:

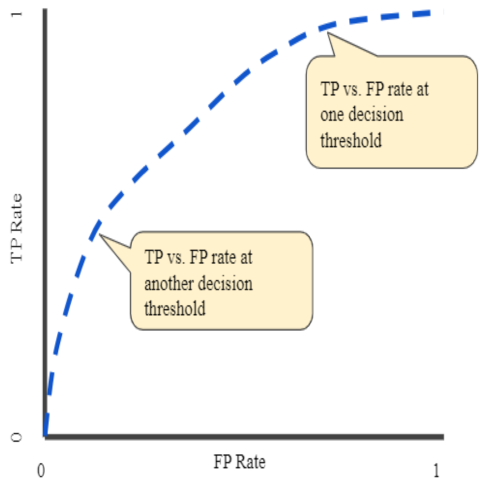


FIGURE 2.29: ROC Curve.

10 Conclusion

Autoencoders are a deep learning technique for current visual recognition tasks. Like all deep learning techniques, they are very dependent on the size and quality of the training data. With a well-prepared dataset, autoencoders can outperform humans in visual recognition tasks.

CHAPTER 3 CONCEPTION

1 Introduction

Artificial intelligence has contributed to the amplification of deepfake fields. Digital images are becoming increasingly realistic by replacing real faces with fake's to arrive at a perfect illusion, making it difficult to distinguish with the naked eye.
Our project focus on deepfakes, in which the main objective is to detect fake faces, using the convolutional autoencoder network.

2 Objective

This work aims to exploit the performance of the convolutional autoencoder network to detect fake faces. The application is embedded in the NVIDIA Jetson Nano 2GB Developer Kit.

2.1 System Architecture

The detailed architecture of our system is shown in figure 3.1:

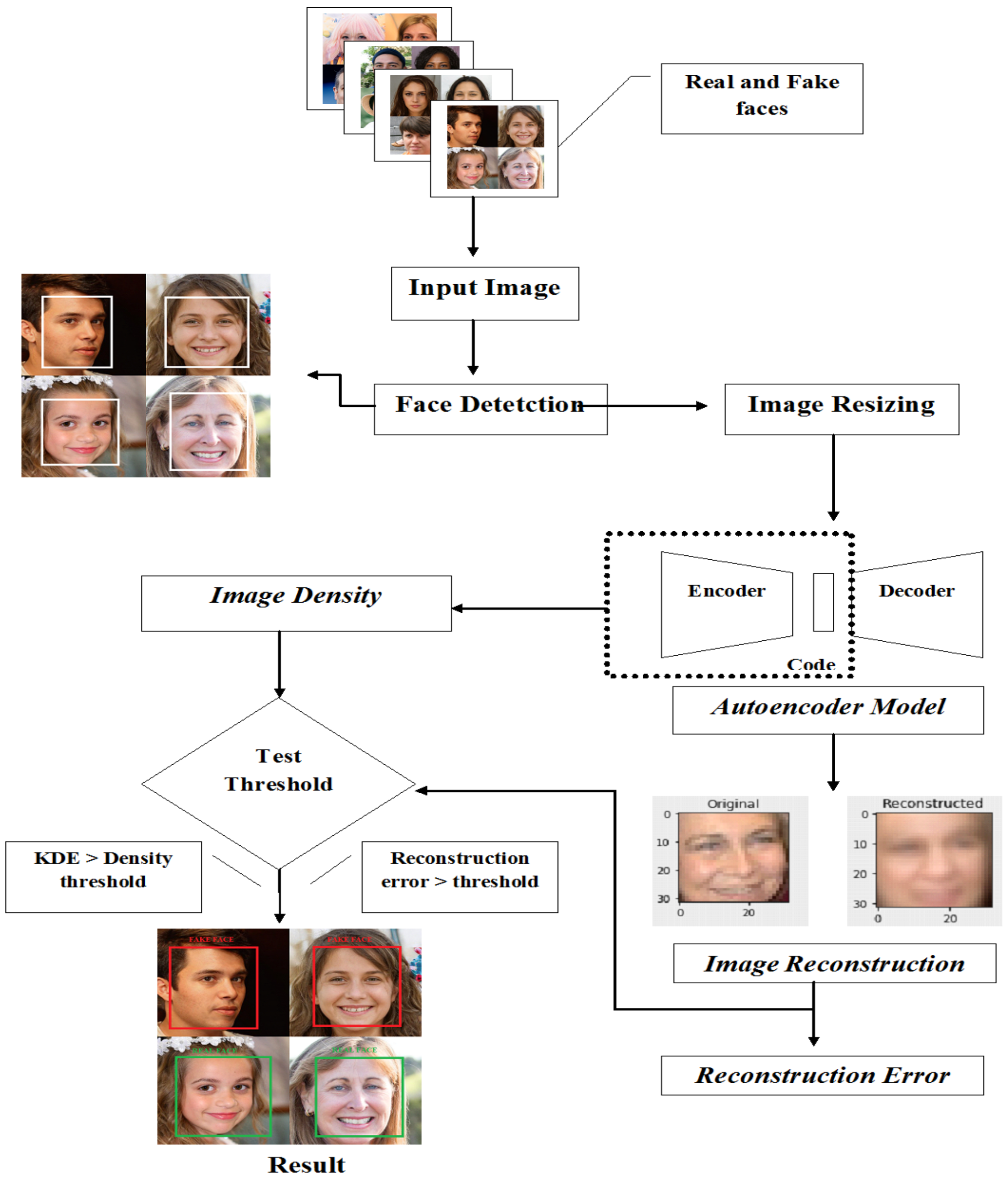


FIGURE 3.1: Model Architecture.

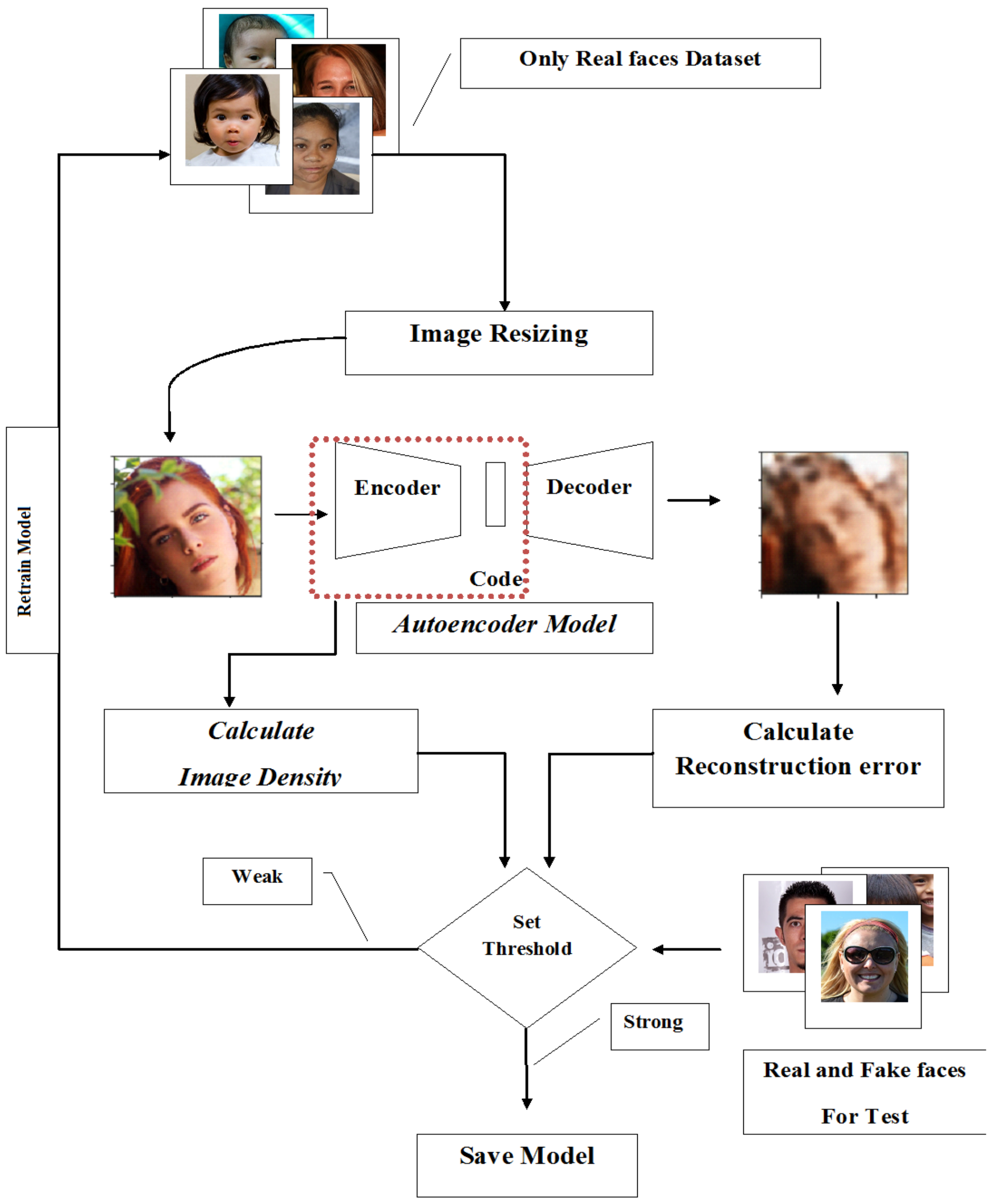


FIGURE 3.2: Model Training Architecture.

3 Face Detection

We evaluated our model on single-face and multi-face images, making it necessary to detect faces before testing. For this purpose, we used the MTCNN [Zha+16] model, a multitask neural network model for face detection.
When MTCNN processes an image, it first performs the image resizing operation to scale the original image to different scales to generate an image pyramid. Then the images of different scales are sent to the three sub-networks for training in order to detect different sizes of human faces and realize multi-scale target detection.[ZLG20]

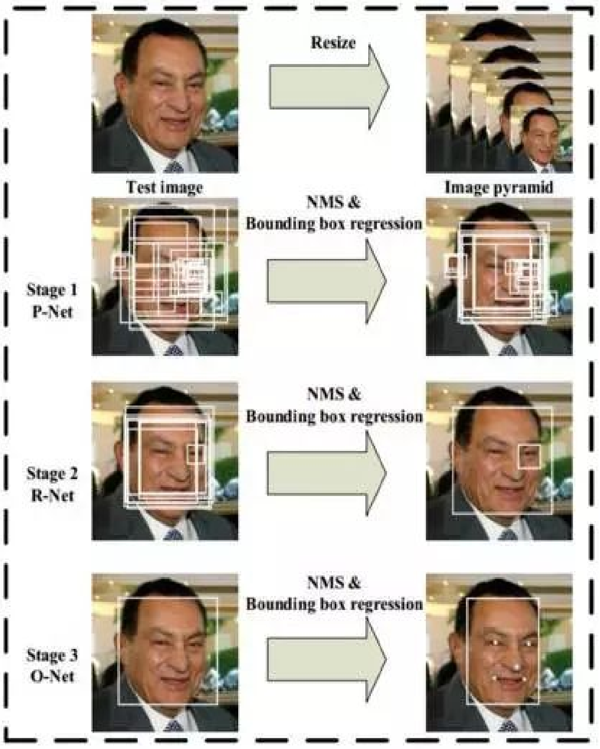


FIGURE 3.3: MTCNN for face detection.

4 Image Resizing

Training images have different sizes; therefore, they must be resized before using model input. We downsampled all images to a fixed resolution of 128 X 128 because our system requires constant input dimensionality.

5 Convolutional Autoencoder Network

A conventional autoencoder is generally composed of two layers, corresponding to encoder *E(x)* and decoder *D(y)* respectively. It aims to find a code for each input sample by minimizing the mean squared errors (MSE) between its input and output over all samples [Guo+17].
Like all autoencoders, a Convolutional autoencoder is composed of two major parts, Encoder, and Decoder.

5.1 Encoder

An encoder consists of layers of neurons that process data to build new so-called encoded representations. In turn, layers of neurons in the Decoder receive these representations and process them in an attempt to reconstruct the original data.
The difference between the reconstructed data and the original data makes it possible to measure the error made by the auto-encoder. The learning consists of modifying the auto-encoder parameters to reduce the reconstruction error measured on different dataset examples.

5.2 Decoder

The Decoder is the last layer that only contains the original data reconstruction, but it is the new representation created by the Encoder.
The simplest architecture of an autoencoder is similar to a multilayer perceptron. However, depending on the processed data, we can use different topologies of neural networks, for example, Convolutional Layers to analyze images or Recurrent Neural Layers to process time series or sequences.

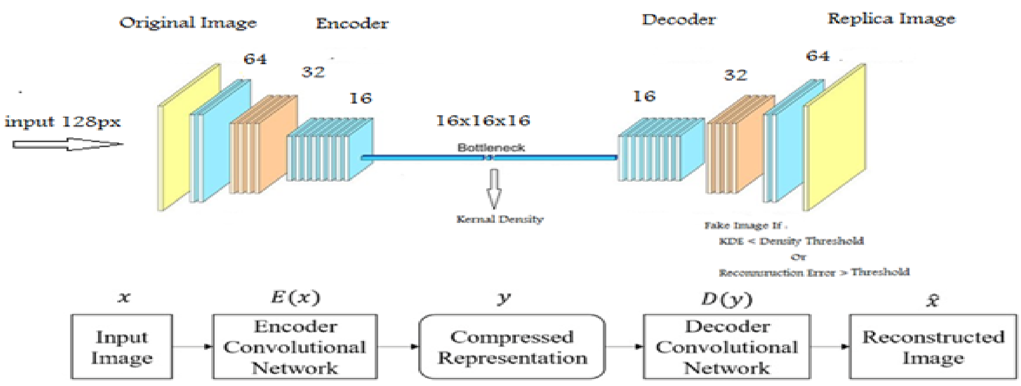
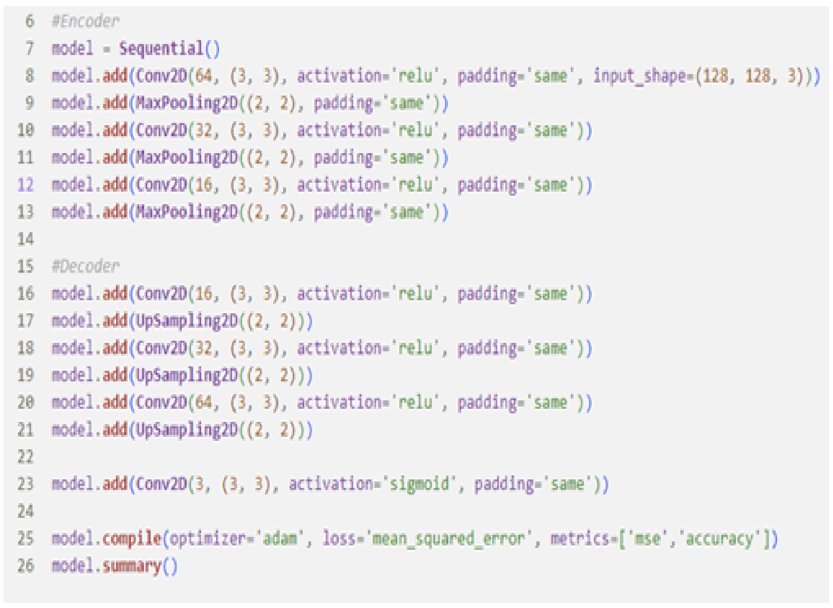


FIGURE 3.4: Convolutional Autoencoder Architecture.

6 Model Configuration

We will train a convolutional neural network. The network consists of 7 convolutional layers. A Max-Pooling layer is placed after each convolutional encoding layer and an Up-Sampling layer between every two convolutional decoding layers.
For faster learning, ReLu is the activation function. Figure 3.5 shows the structure of our model.



```
#Encoder
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(128, 128, 3)))
model.add(MaxPooling2D((2, 2), padding='same'))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2), padding='same'))
model.add(Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2), padding='same'))

#Decoder
model.add(Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))

model.add(Conv2D(3, (3, 3), activation='sigmoid', padding='same'))

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mse','accuracy'])
model.summary()
```

FIGURE 3.5: Components of our neural network.

From Figure 3.5, our CNN Autoencoder model consists of the following components:

The input layer: Have dimension of 128*128*3, where 3 represents the number of color matrix (Red, Green and Blue). **The convolution layer:** in this layer, the size of the filter is fixed at 3*3.

- The number of filters where the depth has been set to 64, 32, and 16 successively for the layers encoding, is the reverse for the decoding layers
- The ReLU activation function: forces the network to return positive values. Any number less than 0 is converted to 0, this technique is applied for all layers except the last layer that uses the sigmoid function.

The pooling layer: we used a max-pooling of size 2x2 to reduce the size of the picture settings.

The Up-Sampling layer: is the reverse pooling layer that allows reverse pooling or reconstructing of the image. The size used is 2x2.

6.1 Encoder and Decoder Configuration details

Layer	Dimension	Kernel size	Activation
input	128X128 px	-	-
Conv2D	128X128X64	3*3	ReLU
Pooling Layer	2*2 window	-	-
Conv2D	64X64X32	3*3	ReLU
Pooling Layer	2*2 window	-	-
Conv2D	32X32X16	3*3	ReLU
Pooling Layer	2*2 window	-	-

TABLE 3.1: Encoder Configuration.

Layer	Dimension	Kernel size	Activation
Conv2D	16X16X16	3*3	ReLU
UpSumpling Layer	2*2 window	-	-
Conv2D	32X32X32	3*3	ReLU
UpSumpling Layer	2*2 window	-	-
Conv2D	64X64X64	3*3	ReLU
UpSumpling Layer	2*2 window	-	-
Conv2D	64X64X64	3*3	Sigmoid

TABLE 3.2: Decoder Configuration.

7 Image Reconstruction

Autoencoders work by learning a representation from a given unlabeled data and reconstructing the data from that representation as accurately as possible using a latent space, a representation of compressed data containing all the important information needed to represent the original data points.
We train our model by inserting real face images into the Encoder, which is used to learn essential and representative features of a given image and represent them in a latent space. The Decoder is then used to reconstruct the image from the latent space by removing noise and unimportant features from the image. These will produce a compressed image. Compression can be lossy because some features are lost in compression, and the resulting image can be blurry.
The loss is calculated by comparing the original image and the reconstructed image, i.e., calculating the difference between pixels in two images. Note that the output of the Decoder must be the same size as the original image, and from this process, we extract the reconstruction error for our training dataset images.

7.1 Mean Square Error (MSE)

Mean squared error (MSE) [GBG22] is one of the most widely used metrics to expression differences between multi-dimensional entities, including images, it is expressed sf follow:

Below x and y are D dimensional vectors, and x_i denotes the value on the ith dimension of x.

$$\textbf{MSE} = \sum_{i=1}^{D}(x_i - y_i)^2 \tag{3.1}$$

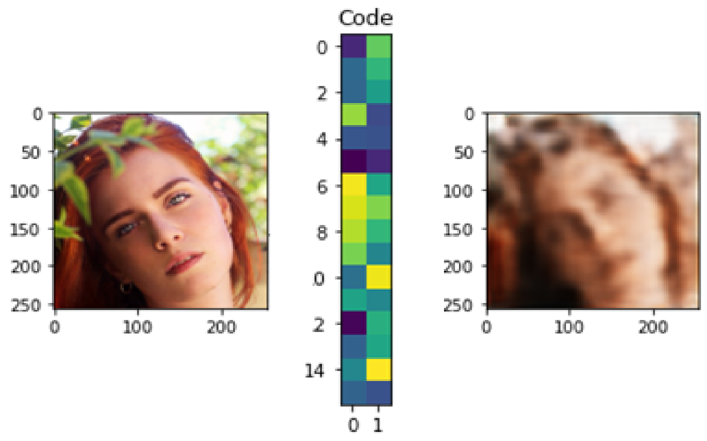


FIGURE 3.6: Reconstructed image from our model.

8 Encoder Model

The encoder network model is extracted and built from the trained autoencoder model with trained weights, then this model will used to get the compressed output from the latent space of the input image. The compressed output is then used to calculate the kernel density estimation (KDE) of each image.

8.1 Kernel density Estimation (KDE)

Kernel density Estimation (KDE) [CNM16] is a non-parametric method of estimating probability density given a sample. Let x1, x2, ..., xn be a set of d-dimensional samples in Rd drawn from an unknown distribution with density function p(x). An estimate $\hat{p}$(x) of the density at x can be calculated using

$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} k_h\Big(x - x_i\Big) \tag{3.2}$$

where Kh : Rd̂ → R is a kernel function with a parameter h called the bandwidth. The Gaussian kernel is common in applications, in KDE each point contributes a small "bump" to the overall density, with its shape controlled by the kernel and bandwidth. The bandwidth parameter h controls the trade-off between bias of the estimator and its variance.

$$k_h = exp(-\tfrac{x^2}{2h^2}). \tag{3.3}$$

9 Evaluation

We use kernel density estimation (KDE) to calculate the likelihood of an image belonging to the real image's class to evaluate our mode and Mean Square Error (MSE) to calculate the reconstruction error of an image.
KDE (kernel density estimation) for training data provides an estimate of where the input image vector space lies in the latent space. Then we assume that:

- Fake images have densities and reconstruction error values farther away from the training images' densities and reconstruction error.

Finally, we set a threshold on KDE (kernel density estimation) and the reconstruction error to identify fakes. This threshold will be chosen based on observations using 'real' and 'fake' data.

10 Conclusion

The fake face detection system that we have implemented in this project is based on the training of the Convolutional Auto-Encoder Network, whose performance mainly depends on the constructed latent space. During the construction phase of latent space, it is essential to set the thresholds of MSE(Mean Squared Error) and KDE(Kernel Density Estimation) parameters, which the system will then use to decide whether an arbitrary image belongs to the class 'Fake' or 'Real'.

CHAPTER 4 IMPLEMENTATION

1 Introduction

The architecture of our system is fixed; we proceed in this chapter to the details concerning the hardware and the software used to implement it. It is essential to validate our project by evaluating the system implemented after training for about a month, the tests were carried out on single-face images of datasets and multi-face images built by us.

2 Environment

2.1 Hardware

1. Jetson Nano

a) Technical Details

Jetson Nano 2GB Developer Kit (figure 4.1) is a new system designed for "learning, building and teaching AI and robotics." The Linux-based system is built around a 128-core NVIDIA Maxwell GPU and a 1.43 GHz quad-core ARM A57 CPU. It offers 2 GB of 64-bit LPDDR4 (25.6 GB/s) memory and microSD-based storage.

- 1. Micro SD card slot (bottom side)
- 2. 40-pin expansion header
- 3. USB 2.0 Micro-B
- 4. Gigabit Ethernet port: 10/100/1000Base-T auto-negotiation
- 5. USB 2.0 type A
- 6. USB 3.0 type A
- 7. HDMI output port
- 8. USB-C 5V 3A: for 5V power input
- 9. MIPI camera connector

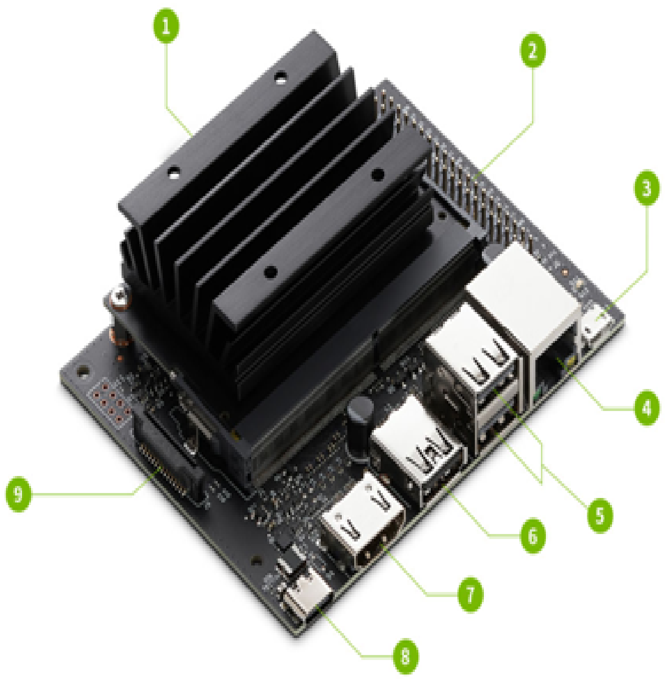
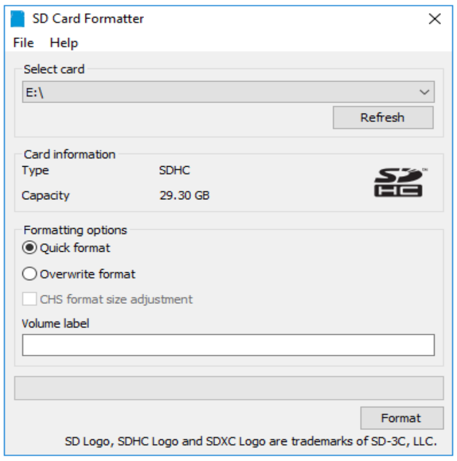


FIGURE 4.1: Jetson Nano 2GB Developer Kit.

b) Start-Up

STEP 1: *Write Image to the MicroSD Card*
To start with, we need to write the Operating System Image to the microSD Card by downloading the "Jetson Nano 2GB Developer Kit SD Card Image ", the jetson nano uses a microSD card as a boot device and for primary storage. It's essential to have a fast and large card for our projects; the minimum requirement is a 32GB card. We need to follow these steps to correctly write the image to the microSD card using our windows computer.

1. Format the microSD card using SD Memory Card Formatter (figure 4.2) from the SD Association.


SD Card Formatter

Select card: E:\

Card information
Type SDHC
Capacity 29.30 GB

Formatting options
[✓ Quick format]
Overwrite format
CHS format size adjustment
Volume label

SD Logo, SDHC Logo and SDXC Logo are trademarks of SD-3C, LLC.

FIGURE 4.2: SD Memory Card Formatter.

2. Use Etcher to write the Jetson Nano Developer Kit SD Card Image to your microSD card.

- Download, install and launch Etcher (figure 4.3).

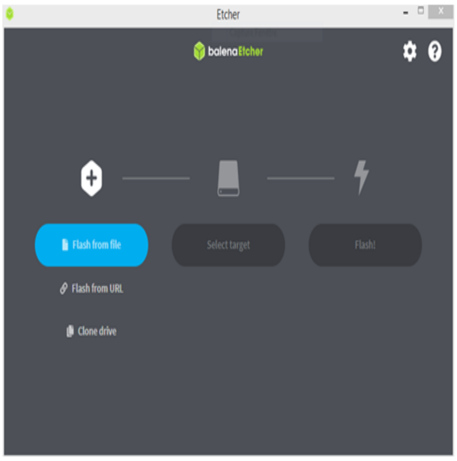


FIGURE 4.3: Balena Etcher.

- Click "Select image" and choose the zipped image file downloaded earlier.
- Insert the microSD card. If not already inserted the Click "Flash!" (figure 4.4)

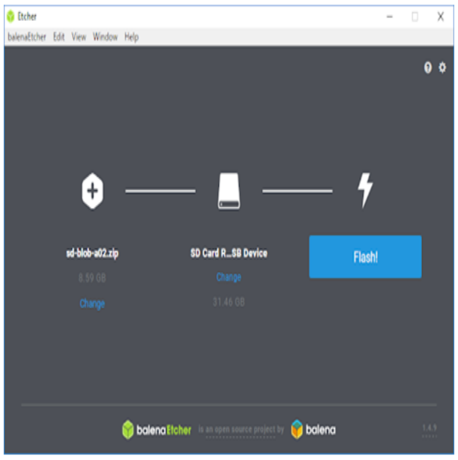

Etcher
balenaEtcher Edit View Window Help
sd-blob-a02.zip
6.59 GB
Change
SD Card R...SB Device
Change
31.46 GB
Flash!
balenaEtcher is an open source project by balena

FIGURE 4.4: Flashing microSD.

After our microSD card is ready, we proceed to set up the developer kit.
STEP 2: *Setup and First Boot*
1. SETUP
There are two ways to interact with the developer kit:
1) with display, keyboard and mouse attached.
2) in "headless mode" via a connection from another computer.
We used the first way with display, keyboard and mouse attached.
Next, we insert the microSD card (with the system image already written to it) into the slot on the underside of the Jetson Nano module, as shown in (figure 4.5). Then we follow the other procedures, including:

- Power on the computer display and connect it.
- Connect our USB keyboard and mouse using the USB 2.0 ports.
- Connect our USB-C power supply (5V3A). The developer kit will power on and boot automatically.

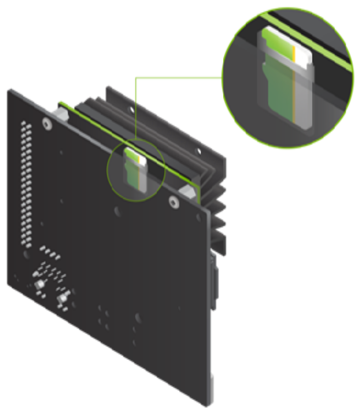


FIGURE 4.5: Insertion of the microSD.

2. *Fisrt Boot*

After connecting the developer kit to the power supply, a green LED next to the Micro-USB connector will light, then we go through some other initial setup, including:

- Review and accept NVIDIA Jetson software EULA.
- Select system language, keyboard layout, and time zone.
- Create username, password, and computer name.
- Optionally configure wireless networking
- Select APP partition size. It is recommended to use the max size suggested
- Create a swap file. It is recommended to create a swap file

And after Logging In, a green screen will be shown (figure 4.6) Installation finished and our jetson nano 2 gb is booted.

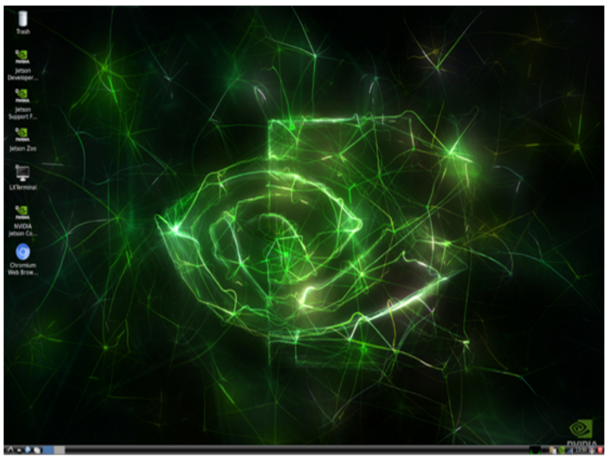


FIGURE 4.6: Jetson nano Desktop Screen.

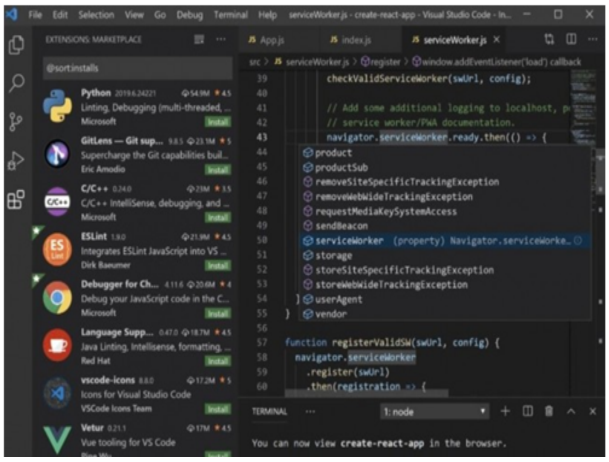
2. Computing Station

The training of our model was carried out in a high-performance computing machine (HPC) from LAIG (Laboratoire d'Automatique et Informatique de Guelma) [site] on 8 Mai 1945 Guelma University [site] . with an edition of Windows 10 system 2021 and RAM equals to 32 Gb, the processor is an Intel(R)Core TM i7 CPU and an Nvidia GPU.

2.2 Software

Visual Studio Code

Visual Studio Code referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux, and macOS. Its features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.
It can also be used with various programming languages, including Java, JavaScript, Go, Node.js, Python, C++, and Fortran.[w7]


File Edit Selection View Go Debug Terminal Help serviceWorker.js - create-react-app - Visual Studio Code - In...
EXTENSIONS: MARKETPLACE
@sort:installs
Python 2019.6.24221 54.9M 4.5 Linting, Debugging (multi-threaded, ... Microsoft Install
GitLens — Git sup... 9.8.5 23.1M 5 Supercharge the Git capabilities buil... Eric Amodio Install
C/C++ 0.24.0 23M 3.5 C/C++ IntelliSense, debugging, and ... Microsoft Install
ESLint 1.9.0 21.9M 4.5 Integrates ESLint JavaScript into VS ... Dirk Baeumer Install
Debugger for Ch... 4.11.6 20.6M 4 Debug your JavaScript code in the C... Microsoft Install
Language Supp... 0.47.0 18.7M 4.5 Java Linting, Intellisense, formatting, ... Red Hat Install
vscode-icons 8.8.0 17.2M 5 Icons for Visual Studio Code VSCode Icons Team Install
Vetur 0.21.1 17M 4.5 Vue tooling for VS Code
App.js
index.js
serviceWorker.js
src > serviceWorker.js > register > window.addEventListener('load') callback
TERMINAL
1: node

```
        checkValidServiceWorker(swUrl, config);

        // Add some additional logging to localhost, p
        // service worker/PWA documentation.
        navigator.serviceWorker.ready.then(() => {
          product
          productSub
          removeSiteSpecificTrackingException
          removeWebWideTrackingException
          requestMediaKeySystemAccess
          sendBeacon
          serviceWorker  (property) Navigator.serviceWorke...
          storage
          storeSiteSpecificTrackingException
          storeWebWideTrackingException
      } userAgent
  }     vendor

function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
```

```
You can now view create-react-app in the browser.
```

FIGURE 4.7: Visual Studio Interface.

Python

Python is a high-level, interpreted, object-oriented programming language. It is in high demand by a large community of developers and programmers. Python is a simple and easy language to learn. The Python library is available for most platforms and is free to redistribute.[VRD03]

Libraries Used

Tensorflow

TensorFlow is an open source machine learning platform that we used to define the basic components of our autoencoder architecture. It offers a comprehensive and flexible ecosystem of tools, libraries, and community resources that allow researchers to advance in the field of machine learning, and developers to easily create and deploy applications that use this technology.[w8]

Keras

This library is a compact and easy-to-learn high-level Python library for deep learning that can run on top of TensorFlow, it provides highly powerful and abstract building blocks that we used to build our deep learning networks and also it allows developers to focus on the main concepts of deep learning, TensorFlow has to be the back end for Keras. We can use Keras for deep learning applications without interacting with the relatively complex TensorFlow.[Man18]

Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O (Input/Output), discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation…etc[Oli06]

Matplotlib

Matplotlib is a Python package for 2D plotting that generates production-quality graphs. It supports interactive and non-interactive plotting, and can save images in several output formats (PNG, PS, and others). It can use multiple window toolkits (GTK+, wxWidgets, Qt, and so on) and it provides a wide variety of plot types (lines, bars, pie charts, histograms, and many more). In addition to this, it is highly customizable, flexible, and easy to use.[Tos09]

Pillow

Pillow is an image processing library, which is a fork and successor of the PIL (Python Imaging Library) project. It is designed to provide quick access to the data contained in an image and offers support for various file formats such as PPM, PNG, JPEG, GIF, TIFF, and BMP. Pillow has relatively powerful image processing capabilities and is intended to provide a solid foundation for any general image processing application.[w9]

OS

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system such as creating and removing a directory (folder), fetching its contents, changing and identifying the current directory,...etc [w10]

PySimpleGUI

We used the PySimpleGUI module which is a python library that wraps Tkinter, PySimpleGUIs main attribute is its simplicity, since it requires up to one half less amount of code in order to produce the same result in comparison to other frameworks and removes the requirement for object-oriented programming. PySimpleGUI was actually built with the purpose to offer the developers to get the required output in as small amount of code as possible and provides a quick and easy way to create graphical applications that work across multiple platforms.[Pod19]

3 Dataset

We used a base of 140,000 color images that contains 70,000 real images from the Flickr dataset[KLA19] collected by Nvidia, as well as 70,000 fake faces images generated by StyleGAN[KLA19]. All the fake and real images we used in our project were resized to 256px before submitting them to the network.

4 Database Splitting

For the "Convolutional Autoencoder Model"
The dataset is divided into three parts (Table 4.1):

- The training images (training set), will be used to train the network. This represents 50,000 images of real faces.
- The test images (test set), are used to evaluate the progress of our model. They present 20,000 Images divided between 10,000 real images and 10,000 fake images.
- The validation images (validation set), present 10,000 images of real faces.

	Train	Validation	Test
Real Faces	50000	10000	10000
Fake Faces	-	-	10000

TABLE 4.1: Model Dataset.

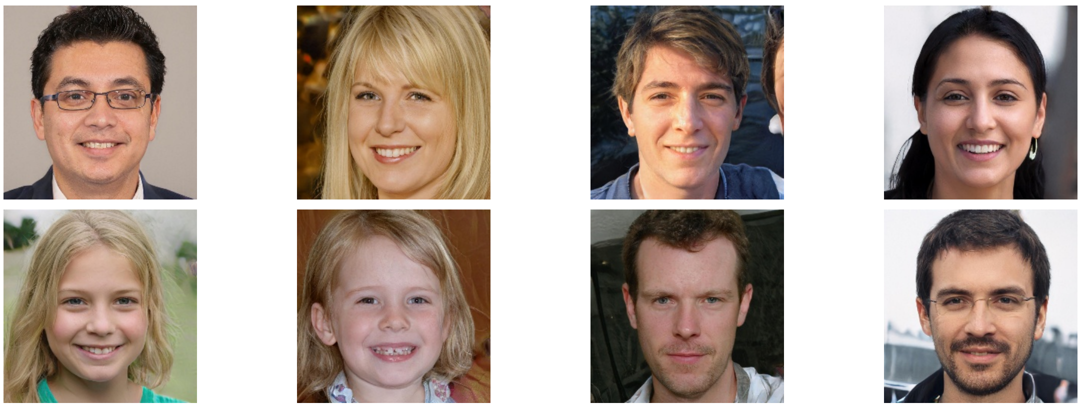


FIGURE 4.8: Fake Faces examples.

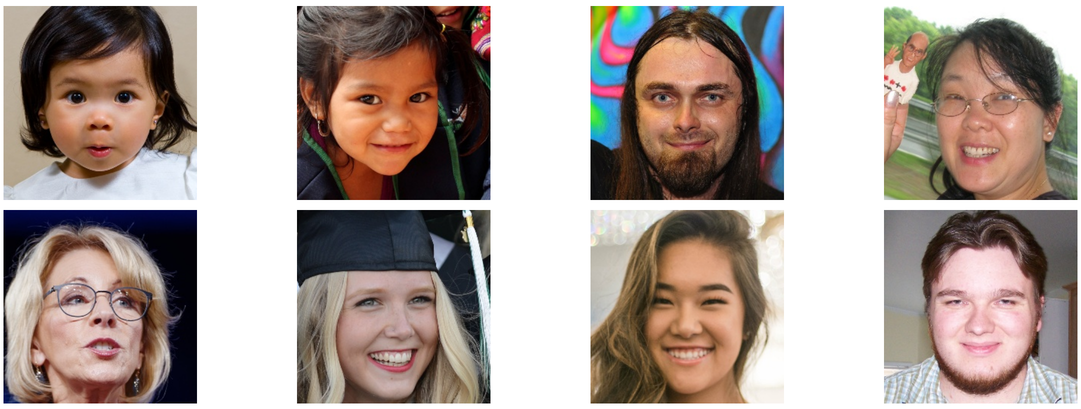


FIGURE 4.9: Real Faces examples.

5 Training and Test

During the learning phase, we experiment several configurations by altering network parameters as the optimizer, the number of iterations, the number of epochs, and the batch size.
For training our network, we have chosen the following configuration:

- Optimizer: "Adam".
- Batch size: 32.
- Epochs number: 50.
- The cost function used: MSE.
- All convolutional layers use the "ReLu" function except the output layer uses the "Sigmoid" function.
- Filter size (3x3), down sampling factor (2x2).

The graphs in Figure 4.10 show that the precision of learning and validation increases with the number of epochs. At the same time, the training and validation loss decreases. So in each epoch, our model learns to recognize more information in a faster way.

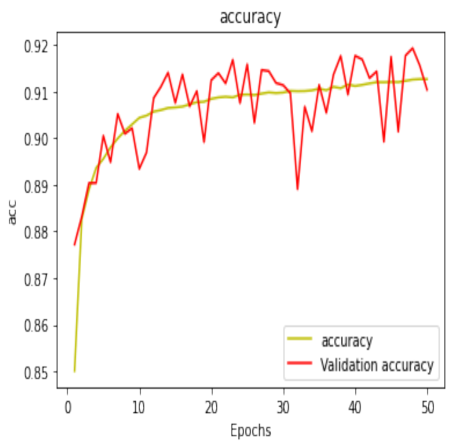


FIGURE 4.10: Accuracy Train and Validation graph.

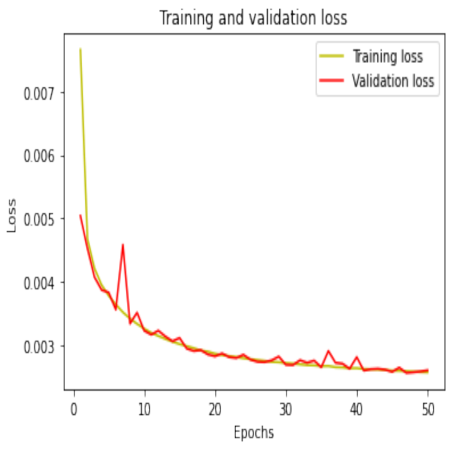


FIGURE 4.11: Train Loss and Validation graph.

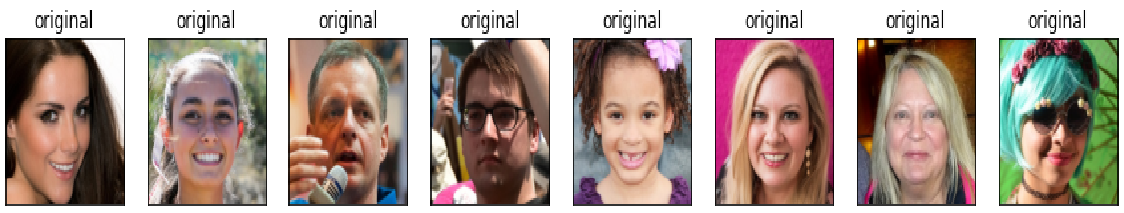


FIGURE 4.12: Original images.

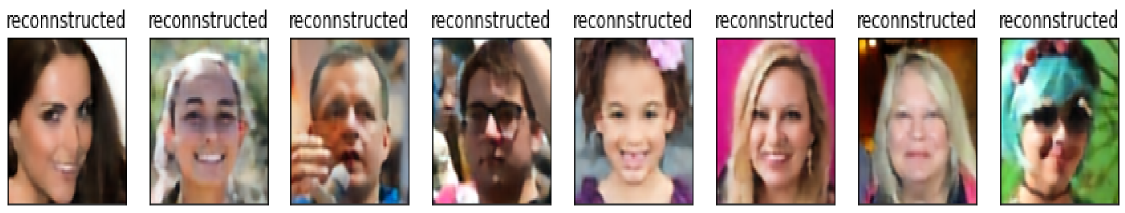


FIGURE 4.13: Reconstructed images.

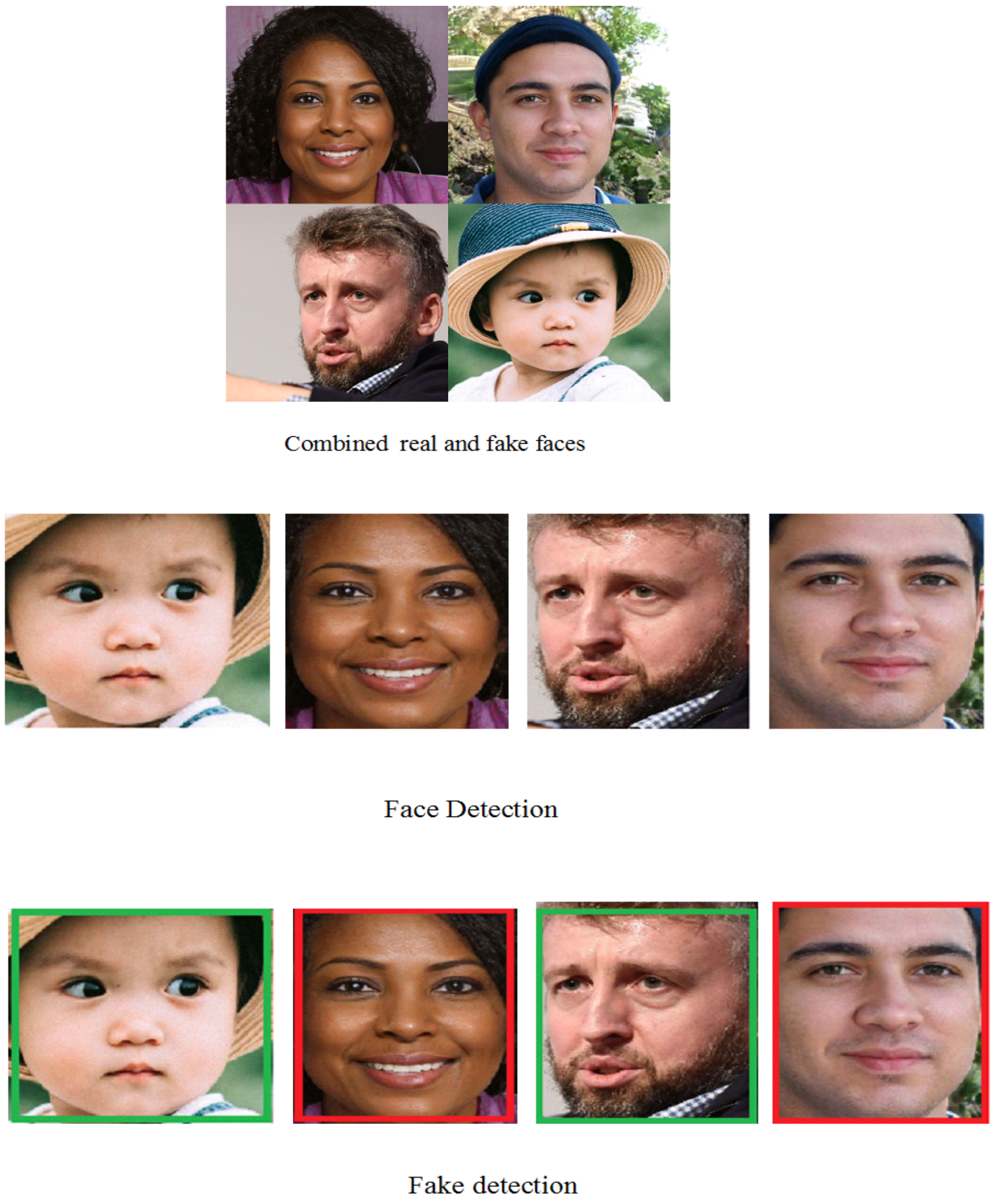


FIGURE 4.14: Test Multi Faces Fake (red rectangle) and Real (green rectangle).

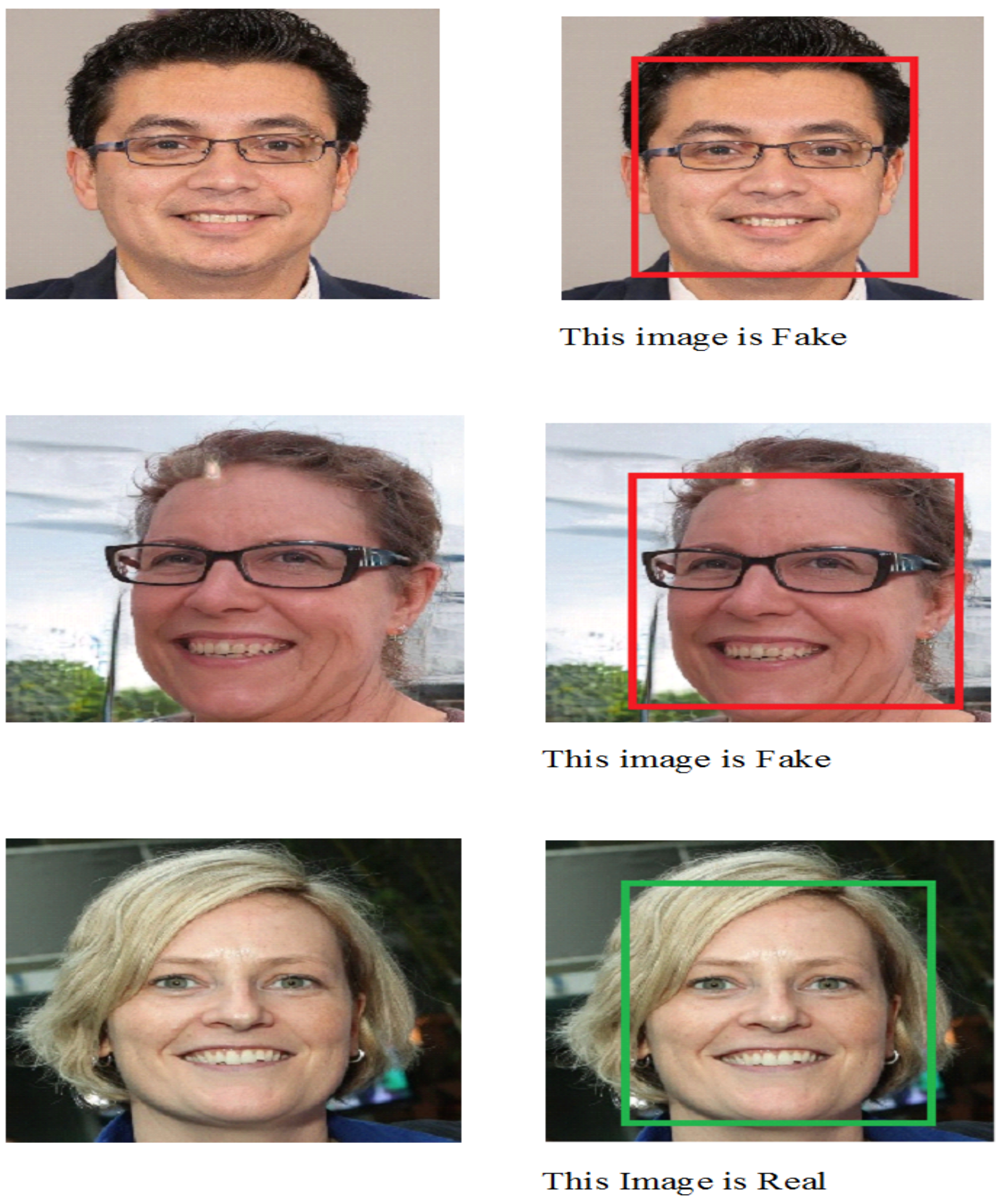


FIGURE 4.15: Test One Face.

6 Conclusion

The results of our system are promising and the tests carried out on the detection of fake faces are encouraging in the field of deepfake detection. The performance of this system can be optimized by using high-quality images and extending the training phase using another type of Auto-encoders.

GENERAL CONCLUSION

Fake face detection systems have become extremely important in the face of the fast dissemination of faked images made by artificial intelligence. It is now feasible to manipulate images and generate fake ones that look so real that even humans cannot tell the difference.
Deepfake is a program belonging to the GAN family, the Generative Adversarial Network that relies on an auto-encoder consisting of an encoder and a decoder network.
In this work, we focus on the Auto-Encoder, a powerful dimensionality reduction tool, and propose a fake face detection system based on a Convolutional Auto-Encoder (CAE). This unsupervised deep learning network allows us to obtain a simple representation of our data on which we then use the estimate of kernel density and reconstruction error to distinguish between "false" and "real" faces.
The choice of the model, its development, and its training was a very important phase that took us more than 3 months before achieving the expected results.
The training of the system was carried out on 50,000 samples of real faces. The tests performed on the fake faces StyleGan dataset yielded favorable results but can be refined by prolonging the training.
The application is embedded in the NVIDIA Jetson Nano 2GB Developer Kit, which will allow our application to be used in robotics applications.
The results of our project have been the subject of participation in an international conference and a national conference as follows:

- 1. Ch. Bencheriet, Z. Taba, M.T. Taba, “Challenge of deep neural network in fake face detection”, 1st International Conference on Engineering and Applied Natural Sciences on 10-13 May in 2022 at Konya/Turkey.
- 2. Ch. Bencheriet, Z. Taba, M.T. Taba, “Fake Face Detection based on Deep Neural Network”, Joumée Scientifique des Mathématiques et de l’Informatique (]SMI2022) on May 16,2022, Djilali Bounaâma Khemis Milliana University, Algeria.

The efficiency of this detection system could be improved. For this purpose, we propose the following:

- Use other Autoencoder models.
- Allow the networks to train longer.
- Use a larger training dataset.

BIBLIOGRAPHY

[Aba+07] AF Abate et al. *Pattern Recognit*. 2007.

[Alm21] Abdulqader M Almars. "Deepfakes detection techniques using deep learning: a survey". In: *Journal of Computer and Communications* 9.5 (2021), pp. 20–35.

[AMAZ17] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 international conference on engineering and technology (ICET)*. Ieee. 2017, pp. 1–6.

[BCS97] Christoph Bregler, Michele Covell, and Malcolm Slaney. "Video rewrite: Driving visual speech with audio". In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 1997, pp. 353–360.

[BKG20] Dor Bank, Noam Koenigstein, and Raja Giryes. "Autoencoders". In: *arXiv preprint arXiv:2003.05991* (2020).

[CET01] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. "Active appearance models". In: *IEEE Transactions on pattern analysis and machine intelligence* 23.6 (2001), pp. 681–685.

[CNM16] Van Loi Cao, Miguel Nicolau, and James McDermott. "A hybrid autoencoder and density estimation model for anomaly detection". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2016, pp. 717–726.

[Cre+18] Antonia Creswell et al. "Generative adversarial networks: An overview". In: *IEEE signal processing magazine* 35.1 (2018), pp. 53–65.

[Dan+19] L Minh Dang et al. "Face image manipulation detection based on a convolutional neural network". In: *Expert Systems with Applications* 129 (2019), pp. 156–168.

[Den12] Li Deng. "The mnist database of handwritten digit images for machine learning research [best of the web]". In: *IEEE signal processing magazine* 29.6 (2012), pp. 141–142.

[Din+20] Xinyi Ding et al. "Swapped face detection using deep learning and subjective assessment". In: *EURASIP Journal on Information Security* 2020.1 (2020), pp. 1–12.

[DW21] Anushree Deshmukh and Sunil B Wankhade. "Deepfake Detection Approaches Using Deep Learning: A Systematic Review". In: *Intelligent Computing and Networking* (2021), pp. 293–302.

[ENM15] Issam El Naqa and Martin J Murphy. “What is machine learning?” In: *machine learning in radiation oncology*. Springer, 2015, pp. 3–11.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[GBG22] Amogh Gudi, Fritjof Büttner, and Jan van Gemert. “Proximally Sensitive Error for Anomaly Detection and Feature Learning”. In: *arXiv preprint arXiv:2206.00506* (2022).

[Guo+17] Xifeng Guo et al. “Deep clustering with convolutional autoencoders”. In: *International conference on neural information processing*. Springer. 2017, pp. 373–382.

[Guo+21] Zhiqing Guo et al. “Fake face detection via adaptive manipulation traces extraction network”. In: *Computer Vision and Image Understanding* 204 (2021), p. 103170.

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. “A style-based generator architecture for generative adversarial networks”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.

[KM18] Pavel Korshunov and Sébastien Marcel. “Deepfakes: a new threat to face recognition? assessment and detection”. In: *arXiv preprint arXiv:1812.08685* (2018).

[KW19] Diederik P Kingma and Max Welling. “An introduction to variational autoencoders”. In: *arXiv preprint arXiv:1906.02691* (2019).

[Man18] Navin Kumar Manaswi. “Understanding and working with Keras”. In: *Deep Learning with Applications Using Python*. Springer, 2018, pp. 31–43.

[Ngu+21] Xuan Hau Nguyen et al. “Learning spatio-temporal features to detect manipulated facial videos created by the deepfake techniques”. In: *Forensic Science International: Digital Investigation* 36 (2021), p. 301108.

[Nir+18] Yuval Nirkin et al. “On face segmentation, face swapping, and face perception”. In: *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE. 2018, pp. 98–105.

[Oli06] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006.

[Pod19] Primoz Podrzaj. “A brief demonstration of some Python GUI libraries”. In: *Proceedings of the 8th International Conference on Informatics and Applications ICIA2019*. 2019, pp. 1–6.

[PRZ21] Zhaoguang Pan, Yanli Ren, and Xinpeng Zhang. “Low-complexity fake face detection based on forensic similarity”. In: *Multimedia Systems* 27.3 (2021), pp. 353–361.

[Sai+15] Tara N Sainath et al. “Convolutional, long short-term memory, fully connected deep neural networks”. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 4580–4584.

[Sou+18] Luiz Souza et al. “How far did we get in face spoofing detection?” In: *Engineering Applications of Artificial Intelligence* 72 (2018), pp. 368–381.

[SSKS17] Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. “Synthesizing obama: learning lip sync from audio”. In: *ACM Transactions on Graphics (ToG)* 36.4 (2017), pp. 1–13.

[SSR20] Mohit Sewak, Sanjay K Sahay, and Hemant Rathore. "An overview of deep learning architecture of deep neural networks and autoencoders". In: *Journal of Computational and Theoretical Nanoscience* 17.1 (2020), pp. 182–188.

[SVM+11] Rifai Salah, P Vincent, X Muller, et al. "Contractive auto-encoders: Explicit invariance during feature extraction". In: *Proc. of the 28th International Conference on Machine Learning*. 2011, pp. 833–840.

[Thi+16] J. Thies et al. "Face2Face: Real-time Face Capture and Reenactment of RGB Videos". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2016.

[Tol+20] Ruben Tolosana et al. "Deepfakes and beyond: A survey of face manipulation and fake detection". In: *Information Fusion* 64 (2020), pp. 131–148.

[Tos09] Sandro Tosi. *Matplotlib for Python developers*. Packt Publishing Ltd, 2009.

[Vas+19] Ivan Vasilev et al. *Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow*. Packt Publishing Ltd, 2019.

[VRD03] Guido Van Rossum and Fred L Drake. *An introduction to Python*. Network Theory Ltd. Bristol, 2003.

[Wan+21] Yongwei Wang et al. "Perception matters: Exploring imperceptible and transferable anti-forensics for GAN-generated fake face imagery detection". In: *Pattern Recognition Letters* 146 (2021), pp. 15–22.

[Wes19] Mika Westerlund. "The emergence of deepfake technology: A review". In: *Technology Innovation Management Review* 9.11 (2019).

[Wu17] Jianxin Wu. "Introduction to convolutional neural networks". In: *National Key Lab for Novel Software Technology. Nanjing University. China* 5.23 (2017), p. 495.

[Zha+16] Kaipeng Zhang et al. "Joint face detection and alignment using multitask cascaded convolutional networks". In: *IEEE signal processing letters* 23.10 (2016), pp. 1499–1503.

[ZLG20] Ning Zhang, Junmin Luo, and Wuqi Gao. "Research on face detection technology based on MTCNN". In: *2020 International Conference on Computer Network, Electronic and Automation (ICCNEA)*. IEEE. 2020, pp. 154–158.

WEBOGRAPHY

[w1] Deepfake: Everything You Need to Know About What It Is How It Works, https://recfaces.com /articles/what-is-deepfake.

[w2] https://medium.datadriveninvestor.com/deep-learning-fundamental-important-concepts-59d7ae90901

[w3] https://medium.com/ai%C2%B3-theory-practice-business/understanding-autoencoders-part-i-116ed2272d35.

[w4] Applied Deep Learning - Part 3: Autoencoders https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

[w5] How to Implement Convolutional Autoencoder in PyTorch with CUDA https://analyticsindiamag.com/how-to-implement-convolutional-autoencoder-in-pytorch-with-cuda/

[w6] Classification: ROC Curve and AUC https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=fr

[w7] Visual Studio Code Visual Studio Code - Wikipedia

[w8] https://www.tensorflow.org/?hl=fr

[w9] Python Imaging Library (PIL) https://he-arc.github.io/livre python/pillow/index.html

[w10] https://docs.python.org/3/library/os.html

Annex

CERTIFICATE Of PARTICIPATION

Z. Taba

has participated in 1st International Conference on Engineering and Applied Natural Sciences on 10-13 May in 2022 at Konya/Turkey.

PAPER TITLE *Challenge of deep neural network in fake face detection*

PRESENTATION TYPE *Oral*

ICEANS 2022

ICEANS 2022 CONFERENCE CHAIRMAN
Asst. Prof. Dr. Umut ÖZKAYA

[illegible]

JSMI
Journée Scientifique des Mathématiques et de l'Informatique 2022

CERTIFICATE

OF PARTICIPATION

This is to certify that

Chemesse Ennehar Bencheriet

has participated as a **Communicant** by the presentation entitled by:

"Fake Face Detection based on Deep Neural Network"

under the following authorship order : **Chemesse Ennehar Bencheriet, Zahra Taba, Mohamed Tahar Taba**
at **the study day "Journée Scientifique des Mathématiques et de l'Informatique (JSMI2022)"**
on **May 16, 2022** organised by the **mathematics and computer science department (M.I)**
inside the **faculty of sciences and technology (F.S.T)** under the full-supervision
of **Djilali Bounaâma Khemis Milliana University** as part of its 2022 program .

Our organization is honoured by your presence as a good asset for this event.

ORGANISATION COMMITTEE CHAIR	DEAN OF F.S.T	HEAD OF MI DEPARTMENT
Imène AIT ABDERRAHIM	Djelloul BENZAID	Ali KRELIFA
[illegible]	D. Benzaid	[illegible]

2022