MONASH University

Neural Network-Based Deepfake Detection

Seow Jia Wen

Doctor of Philosophy

A Thesis Submitted for the Degree of Doctor of Philosophy at
Monash University in 2023
School of Information Technology

Copyright notice

I certify that I have made all reasonable efforts to secure copyright permissions for third-party content included in this thesis and have not knowingly added copyright content to my work without the owner's permission.

Abstract

Deepfake is an advanced synthetic media technology that can generate deceptively authentic yet forged images and videos by modifying a person's likeliness. The term "Deepfake" is a portmanteau of "Deep learning" and "Fake," which reflects the utilization of artificial intelligence and deep learning algorithms in creating deepfake. The deepfake generation involved training to learn the nuances of facial attributes, facial expressions, motion movement, and speech patterns to produce fabricated media that are indistinguishable from the actual footage.

When wielded with malicious intent, deepfakes could cause detrimental impacts on individuals, organizations, politics, and society. Numerous deepfake detectors have been introduced to mitigate these challenges, including conventional and deep learning-based models. While deep learning-based deepfake detection models have shown promising results, several limitations still require improvement. Most deep learning-based detectors are computationally intensive and difficult to be replicated. Due to the lack of diversity in training data, they do not generalize well against unseen data and could easily overfit to specific deepfake types. Moreover, adversarial attacks could easily fool the neural network-based detection models.

This thesis attempts to resolve the drawbacks of deep learning-based deepfake detection models. It draws inspiration from Mesonet, which exploits discriminative features at the mesoscopic level to ensure a low computational training cost. Still, it focuses more on local spatial feature learning and global spatial feature preservation. The idea of the proposed network is based on the nature of deepfake data. Deepfake is often used to manipulate human content, especially the invariant facial regions. The spatial relationship between the facial attributes is vital for generating a convincing hyper-realistic deepfake output. The subtle inconsistency between face features, such as eye spacing, skin color, and mouth shape, could be used as a telltale sign of deepfake discrimination. Using the key concept of separable convolution, the reduction block "bottleneck" structure introduced from the ResNet, and strided-convolution, the first contribution of this thesis is the development of SparcoNet, a spatial cost-efficient deep neural network for deepfake detection. Experiments results have shown that the proposed SparcoNet achieves an average of 0.985 AUC among six state-of-the-art deepfake datasets and obtains a similar result for intra-datasets evaluation as the complex detection network with less than 1% differences in the generalization gap index.

The second contribution of the thesis proposed to further improve the network performance in inter-datasets evaluation and adversarial attack defense ability by training the SparcoNet in a Self-Supervised Learning (SSL) manner with different data transformations. It incorporates a normalized temperature-scaled cross-entropy loss function to

facilitate the learning of discriminative features in various data augmentations. Experiment results have shown that the SSL-SparcoNet has reduced the success rate of the Fast Gradient Sign Method (FGSM) adversarial attack by, at most 85% and improved the inter-cross data evaluation performance by an average of 12%.

The third contribution of the thesis aims to enhance the defense ability of SSL-SparcoNet against black-box adversarial attacks by introducing Block Switching (BS) as a framework protector. The BS-SSL SparcoNet switches the runtime channel randomly to confuse the attacker in predicting the network information. Compared to other protection techniques, such as adversarial training, defensive distillation, and ensemble methods, the proposed BS-SSL SparcoNet obtains higher flexibility in retaining network configurations. Experiment results have shown that the proposed model reduced the attack success rate of black-box adversarial attacks from a surrogate model by more than 90%.

Declaration

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature: _____

Print Name: Seow Jia Wen

Date: _____

Thesis including published works declaration

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

This thesis includes 1 original paper published in a journal and 1 submitted conference publication. The core theme of the thesis is *Neural Network-Based Deepfake Detection*. The ideas, development, and writing up of all the papers in the thesis were the principal responsibility of myself, the student, working within the *Faculty of Information Technology* under the supervision of *Dr. Lim Mei Kuan, Prof. Raphaël C.-W. Phan, and Prof. Joseph Liu*.

I have renumbered sections of submitted or published papers in order to generate a consistent presentation within the thesis.

Student name: Seow Jia Wen

Student signature:

Date:

I hereby certify that the above declaration correctly reflects the nature and extent of the student's and co-authors' contributions to this work. In instances where I am not the responsible author, I have consulted with the responsible author to agree on the respective contributions of the authors.

Main Supervisor name: Dr. Lim Mei Kuan

Main Supervisor signature:

Date:

Publications during enrolment

1) Jia Wen Seow, Mei Kuan Lim, Raphaël C.W. Phan, Joseph K. Liu (2022). A comprehensive overview of Deepfake: Generation, detection, datasets, and opportunities. Neurocomputing, 513, 351-371.

2) Jia Wen Seow, Mei Kuan Lim, Raphaël C.W. Phan, Joseph K. Liu (2023). SparcoNet with Block-Switched Self-Supervised Learning: An Effective Framework for Deepfake Detection with Improved Adversarial Defense. Submitted to Information Science Journal [Submitted].

Acknowledgements

Embarking on this research project was intimidating and filled with moments of triumph and defeat. I am grateful to have been surrounded by a group of trusted and wonderful individuals who stood by me and supported me in conquering the challenges. First, I would like to express my heartiest gratitude to my main supervisor, Dr. Lim Mei Kuan. She played a crucial role throughout my Ph.D. journey with her immense guidance and encouragement. As a newbie researcher, I went through countless frustrations and self-doubts. With regular meetings and prompt responses, her advice has given me the confidence to overcome the unforeseen challenges I faced. She was always there for me and motivated me with her enthusiasm, and led me back in the right direction. Her belief in me and my abilities has been invaluable, and I truly appreciate her presence in my life as a mentor and friend.

I also want to express my heartfelt thanks to my co-supervisors, Prof. Raphaël Phan and Prof. Joseph Liu. Their expertise and valuable insights are pivotal in shaping my research project and helping me overcome obstacles. Their deep research understanding and rich publication experiences have provided me with informative input in building the foundation of writing research papers and selecting publication venues. I am deeply grateful for their generous sharing of time, knowledge, and effort, as well as their unwavering support along the way. Apart from that, a billion thanks to my panels and examiners, who provided me with constructive feedback and insightful comments for each review milestone. Their professionalism and attention to detail have helped me to refine and enhance my work in countless ways. I am indebted to their valuable contributions and grateful for the time and effort they devoted to reviewing my research.

Furthermore, I would like to express my deepest gratitude for the kind support received from the Monash Graduate Research Office (MGRO), Faculty of Information Technology Monash, Monash University Postgraduate Association (MUPA), and all the incredible people who provided me assistance throughout the journey, especially Mr. Pratheeban and Ms. Leena from the operation management office; Ms. Adleen and Ms. Azizun from the research management office; Ms. Rachel and Ms. Julie from the Monash University Clayton. I would also like to thank Monash University for awarding me the Monash Graduate Merit Scholarship and Vice-Chancellor's International Inter-Campus Ph.D. Travel Grant, which not only provided me with the opportunity to attend the FIT Retreat Program at Monash University Clayton in Australia but also enabled me to pursue my research goals with more significant resources and support.

Behind the scenes, a remarkable group of people have provided me with tremendous support during these 3.5 years. I am deeply grateful for the unwavering support and understanding that my family has given me throughout my research journey. Their

encouragement, love, and patience have been crucial to my success, and I could not have accomplished this without them. I sincerely thank Tin, Wen Xin, Fan Yi, Dilshani, Jessie, Heng Ee, Zun Ci, Ming Jie, and Lucas. The times we spent together in the lab, badminton court, and food court will always be missed. Although the pandemic has destroyed our peaceful research lifestyle, the memory will always be there. I would also like to extend thanks to Yin Yin and Vinky; a thousand thanks to Monash for letting us meet and travel together. Both of you are charming and delightful; you brightened my Ph.D. research journey again before it ended. Research during the pandemic has proven to be more challenging than anticipated. I have missed the times when we could offer each other support and warm hugs during our deep conversations about our struggles throughout the research process. I am immensely grateful to my dear friends Moon and Clara for their support during all the stressful times. They bought me food, listened to me, and provided emotional support when needed. Lastly, I want to thank my lovely puppies, Tiffie and Cobbie. Their companionship during the challenging times of the pandemic helped me overcome stress and depression.

Contents

Copyright notice i

Abstract ii

Declaration iv

Thesis including published works declaration v

Publications during enrolment vi

Acknowledgements vii

List of Figures xiii

List of Tables xv

Abbreviations xvii

Constants xviii

Symbols xix

1 Introduction 1
- 1.1 Thesis Motivation & Methodology . . . 4
- 1.2 Research Question & Research Objectives . . . 6
- 1.3 Research Scope . . . 6
- 1.4 Thesis Outline . . . 7

2 Fundamental of Neural Networks 8
- 2.1 Introduction to Artificial Neural Network . . . 8
  - 2.1.1 Terminology Behind Neural Networks . . . 9
- 2.2 Type of Neural Network . . . 11
  - 2.2.1 Percepton . . . 11
  - 2.2.2 Feed-Forward Neural Network . . . 12
  - 2.2.3 Multilayer-Percepton (MLP) . . . 13
  - 2.2.4 Convolutional Neural Network (CNN) . . . 13
  - 2.2.5 Recurrent Neural Network (RNN) & Long Short-Term Memory (LSTM) . . . 14
- 2.3 Neural Network Training . . . 16
  - 2.3.1 Supervised Learning . . . 16

2.3.2 Unsupervised Learning . . . 16
2.3.3 Semi-supervised Learning . . . 18
2.3.4 Self-supervised Learning . . . 19
2.3.5 Transfer Learning . . . 19
2.3.6 Overfitting & Underfitting . . . 19
2.3.7 Regularization . . . 22
2.4 Parameter Configuration . . . 23
2.5 Activation Function . . . 24
2.6 Loss Function . . . 26
2.7 Gradient Descent . . . 28
2.7.1 Batch, Mini-Batch, and Stochastic Gradient Descent . . . 28
2.7.2 Optimizers . . . 29
2.7.3 Vanishing Gradient & Exploding Gradient . . . 30
2.8 Development Framework . . . 31
2.9 Backbone CNN Models . . . 31
2.10 Training Constraint . . . 32

3 Overview of Deepfake 33
3.1 Deepfake Generation . . . 33
3.1.1 Deepfake Generation Techniques . . . 34
3.1.2 Deepfake Types . . . 35
3.1.2.1 Face Systhesis . . . 35
3.1.2.2 Reenactment . . . 37
3.1.2.3 Facial Attributes Manipulation . . . 42
3.1.2.4 Face Swapping . . . 43
3.1.3 Available Deepfake Generation Tool . . . 45
3.1.4 Disscusion . . . 45
3.2 Deepfake Detection . . . 46
3.2.1 Traditional Deepfake Detection Methods . . . 49
3.2.2 Deep learning-based Deepfake Detection Methods . . . 52
Biometric artifact . . . 52
Spatio-Temporal Feature . . . 54
Pixel & Statistical Feature . . . 54
Face Recognition & Artifacts Discrepancies . . . 56
Multi-Stream & Multi-Stack Neural Network . . . 57
Shallow-CNN . . . 57
Attention Mechanism . . . 58
Contrastive Loss & Triplet Loss . . . 59
CNN Rearchitecture . . . 59
Capsule Network . . . 60
Recurrent Neural Network (RNN) . . . 60
Autoencoder . . . 61
Multi-Person Forgery . . . 61
Self-supervised learning . . . 62
3.2.3 Deepfake Datasets . . . 62
3.2.4 Evaluation Metrics . . . 65
3.2.5 Discussion . . . 65

3.3 Challenges . . . 67
3.3.1 Opportunities in Deepfake Generation . . . 73
Few-shot Learning . . . 73
Deepfake Quality . . . 73
Real-time Deepfake . . . 74
3.3.2 Opportunities in Deepfake Detection . . . 74
Adversarial Attack . . . 74
Model Generalization . . . 74
Application & Platform-friendly . . . 75
3.3.3 Summary . . . 75

4 A shallower and spatially cost-efficient network structure (SparcoNet) 77
4.1 Motivation . . . 77
4.2 SparcoNet Development . . . 78
4.2.1 Model Architecture . . . 80
4.2.2 Network parameters . . . 83
4.2.3 Ablation study . . . 83
4.2.4 Data Preprocessing . . . 84
4.3 Experimental Setup . . . 85
4.3.1 Datasets . . . 85
4.3.2 Evaluation Metric . . . 86
4.3.3 Experiments & Results . . . 86
4.3.3.1 Cross-dataset Evaluation . . . 87
4.4 Visualization . . . 91
4.4.1 Feature Map . . . 92
4.4.2 Grad-CAM . . . 93
4.5 Summary . . . 95

5 SparcoNet with Self-Supervised Learning (SSL-SparcoNet) 96
5.1 Motivation . . . 96
5.2 SSL SparcoNet Development . . . 97
5.2.1 Model Enhancement . . . 99
5.2.2 Data Preprocessing . . . 100
5.3 Experimental Setup . . . 101
5.3.1 Evaluation Metric . . . 103
5.3.2 Experiment & Results . . . 103
5.4 Summary . . . 106

6 Block-Switching SSL SparcoNet (BS-SSL SparcoNet) 107
6.1 Motivation . . . 107
6.2 BS-SSL SparcoNet Development . . . 108
6.2.1 Training Algorithm . . . 109
6.2.2 Channel Selection . . . 110
6.3 Experiments & Result . . . 111
6.4 Summary . . . 112

7 Conclusions 114
7.1 Contributions . . . 114

7.2 Limitations 115
7.3 Future Directions 116

Bibliography **117**

List of Figures

1.1 Examples of Deepfake . . . 2
1.2 The overall architecture of the proposed methodology . . . 5

2.1 Network illustration and matrix calculation representation of neural networks . . . 11
2.2 Percepton network illustration . . . 12
2.3 Feed-Forward neural network illustration . . . 12
2.4 MLP network illustration . . . 13
2.5 CNN network illustration . . . 14
2.6 RNN network illustration . . . 15
2.7 LSTM network illustration [1] . . . 15
2.8 Example of Supervised Learning . . . 17
2.9 Example of Unsupervised Learning . . . 17
2.10 Example of Semi-supervised Learning . . . 18
2.11 Example of Self-supervised Learning . . . 20
2.12 Example of Transfer Learning . . . 21
2.13 Trade-off between Variance and Bias [2] . . . 22
2.14 Graph representation of activation functions [3] . . . 26

3.1 The encoder, pθ converts the input data, x into a latent representation, y, and the decoder pφ reconstruct the data back as output, x'. . . . 35
3.2 The four main types of deepfake. . . . 36
3.3 Deepfake detection . . . 49
3.4 Common detection pipeline . . . 49
3.5 Example of handcrafted features . . . 50
3.6 The implementation frequency of deep learning & non-deep learning-based deepfake detection model structure based on discussed studies . . . 67
3.7 The implementation frequency of the popular CNN architecture for deepfake detection model based on discussed studies . . . 67

4.1 The structure of an Inception module [4] . . . 80
4.2 The structure of a residual connection [5] . . . 80
4.3 The structure of an Inception module with residual connection [6] . . . 81
4.4 The architecture of the proposed SparcoNet. . . . 81
4.5 Sample inaccurate detection (where real images are detected as deepfake) for the different face detectors to demonstrate bias caused by incorrect face region, (a) Dlib (b) Cascade Classifier (c) MTCNN. . . . 85
4.6 The AUC Graph for SparcoNet performance evaluation on state-of-the-art datasets . . . 88

4.7 Example of Raw, C23, and C40 data 88
4.8 The feature map of the four major blocks of Mesonet 93
4.9 The feature map of the four major blocks of SparcoNet 93
4.10 The feature map with 16 filters of the four major blocks of SparcoNet . . 94
4.11 The gradient map comparison between real, FFF2F, and FFDF output . 94

5.1 Example of Adversarial Attack 97
5.2 The concept of Self-Supervised SparcoNet Model 98
5.3 The architecture of Self-Supervised SparcoNet Model. 99
5.4 Examples of self-supervised augmented training data that applied to the original fake and real data 102
5.5 Examples of the tuple data with random occlusion transformation 102
5.6 Attack Success Rate Against FGSM Epsilon 106

6.1 The workflow of Block Switching 109
6.2 The attack success rate against number of channel when dealing with FGSM at 0.35 epsilson 111
6.3 The average ASR deviation against number of channel 113

List of Tables

2.1 Total Parameters of each backbone CNN 32

3.1 Available deepfake generation tools or applications 46
3.2 Summary of deepfake generation part 1 47
3.3 Summary of deepfake generation part 2 48
3.4 Confusion matrix 65
3.5 Formula and explanation for each evaluation metric 66
3.6 Summary of pure handcrafted feature & handcrafted-feature with machine learning based deepfake detection approaches 68
3.7 Summary of deep Learning-based deepfake detection part 1 69
3.8 Summary of deep Learning-based deepfake detection part 2 70
3.9 Summary of deep Learning-based deepfake detection part 3 71
3.10 List of deepfake dataset 72

4.1 The Inner block details of the proposed SparcoNet, the (16,32,64) indicated the neuron numbers. 82
4.2 Comparison of model parameters, network depth, and model complexity(FLOPS) between Mesonet, Inception-Resnet-V2, Xception network, and the proposed SparcoNet 83
4.3 Performances comparison of ablated SparcoNet structure on FF+ dataset 84
4.4 Detection performances of each face detector for Face2Face dataset 85
4.5 SparcoNet performance evaluation on state-of-the-art datasets 87
4.6 Comparison with state-of-the-art models on Raw, C23, and C40 compression factors of FF++ F2F dataset 89
4.7 Intra-datasets Evaluation with state-of-the-art baseline models on C23 compression factor between FF++ F2F & DF datasets 90
4.8 Inter-dataset evaluation Comparison with state-of-the-art baseline models trained with FF++ C23 dataset 91
4.9 SparcoNet robustness evaluation against Raw, C23, and C40 compression factor of F2F dataset 91

5.1 Comparison of inter cross-datasets evaluation (Trained with Face Forensics++ F2F) between the original SparcoNet with SSL-SparcoNet trained with 1 transformation on different augmentaions 104
5.2 Comparison of intra cross-datasets evaluation (Trained with Face Forensics++ F2F) between the original SparcoNet with SSL-SparcoNet trained with 1 transformation on different augmentaions 105
5.3 Comparison of FGSM Attack Success Rate between the original SparcoNet with SSL-SparcoNet trained on different augmentaions 105

6.1 Comparison of FGSM Attack Success Rate at different Block Switching channels 112

Abbreviations

CV	**C**omputer **V**ision
ML	**M**achine **L**earning
DL	**D**eep **L**earning
GAN	**G**enerative **A**dversarial **N**etwork
AI	**A**rtificial **I**ntelligence
SSL	**S**elf-**S**upervised **L**earning
LSTM	**L**ong **S**hort **T**erm **M**emory
CNN	**C**onvolutional **N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
BS	**B**lock **S**witching
FF	**F**ace **F**orensics
DFDC	**D**eep**F**ake **D**etection **C**hallenge
VAE	**V**ariational **A**utoencoders
MFCC	**M**el-frequency **C**epstral **C**oefficients
PCA	**P**rincipal **C**omponent **A**nalysis
BoW	**B**ag **o**f **W**ords
SURF	**S**peeded **U**p **R**obust **F**eatures
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SVM	**S**upport **v**ector **M**achine
PRNU	**P**hoto **R**esponse **N**on-Uniformity

Constants

Speed of Light $c = 2.997\ 924\ 58 \times 10^8\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	rads^{-1}

Chapter 1

Introduction

Visual aids are commonly utilized across various industries such as law, medicine, and entertainment [7, 8]. However, the extensive usage of visual media also presents a risk of misuse. Media forgery has been prevalent in digital culture for a while, where software tools like Photoshop are used for manual manipulation of media content. With the recent advancements in Computer Vision (CV) and Machine Learning (ML) technologies, media forgery has become more accessible and widespread.

In 2012, the field of CV experienced a significant breakthrough when AlexNet, an AI model developed by Alex *et al.* [9], outperformed other models in the image recognition challenge by a large margin. Since then, AlexNet, which is a classic convolutional neural network architecture, has been instrumental in many CV applications. Another leap forward in CV research was made in 2014 when Goodfellow *et al.* [10] introduced the Generative Adversarial Network (GAN). GAN enables the creation of realistic-looking images from scratch without human intervention or manual editing.

The rapid evolution of hardware that supports artificial neural network models' training has catalyzed the growth of deep learning. In 2017, a novel deep learning-based media forgery algorithm called 'Deepfake' emerged and wreaked havoc, threatening society's security and privacy. Deepfake is a synthetic technique that replaces the person in an existing image or video with someone else's likeness or characteristics. It is a portmanteau of 'deep learning' and 'fake'. It originated from an anonymous individual under the pseudonym 'deepfake' who uploaded numerous pornographic videos to the Reddit website. The actresses' faces in the videos were swapped with those of other celebrities

using deep learning [8]. Figure 1.1 outlines the examples of deepfake based on different generation methods. Based on the figure, a. Puppet Master refers to the transfer of motion movement by synthesizing the motion of the source and regenerating onto the target output [11]; b. Face Swapping involves swapping the facial regions between two people from one to another [12]; c. and e. involve facial reenactment where Neural Textures focuses on deferred neural rendering to integrate neural textures in the parametric vectors for facial synthesis [13] while Face2Face uses GAN, such as CycleGAN and StarGAN to achieve the synthesis output [14]; d. and h. present entire face synthesis to produce non-exist human outputs by the training on the different source data to capture their significant facial characteristics [15, 16]; f. and g. leverages GAN's capability to modify certain facial attributes on target.

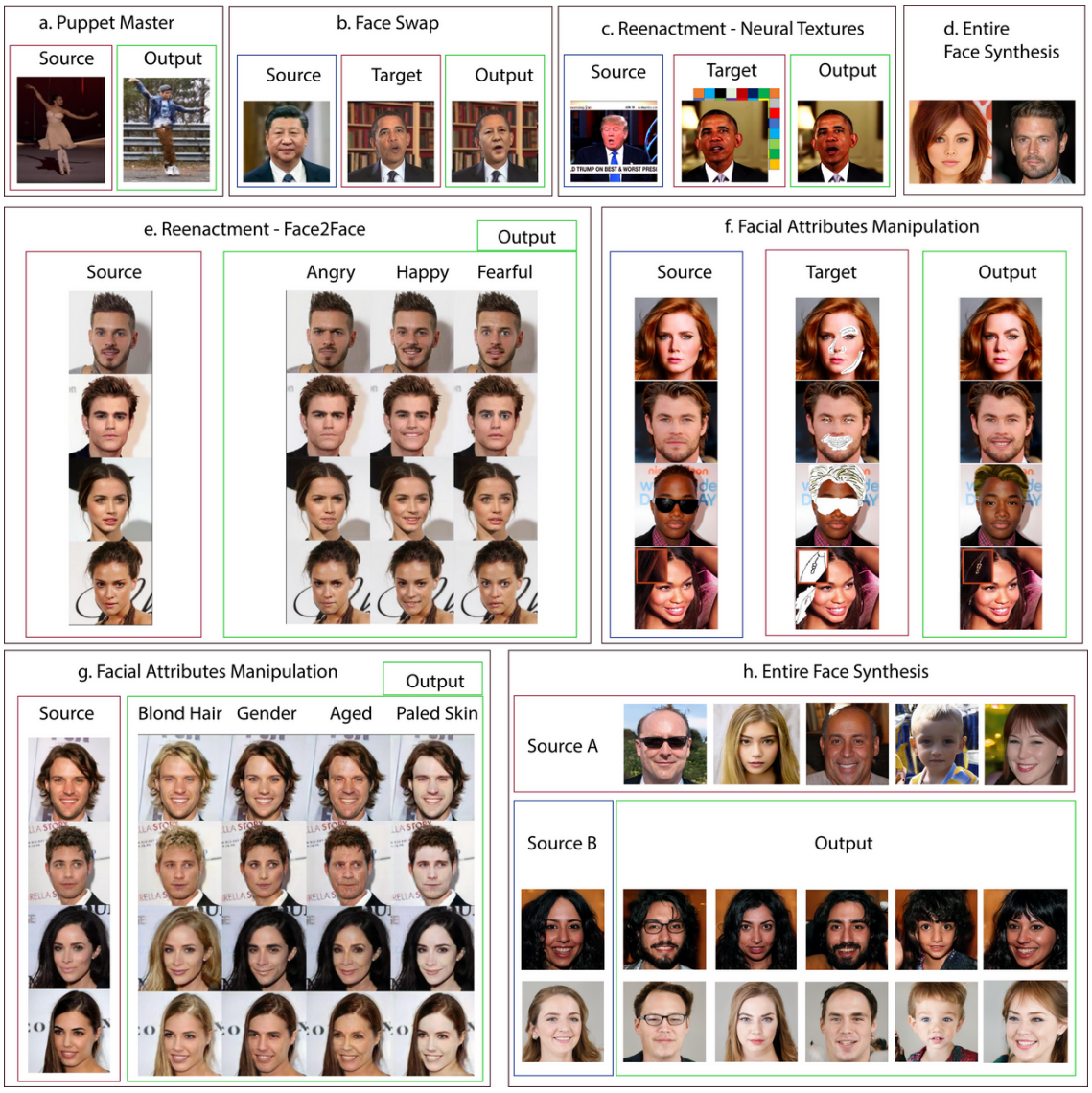


FIGURE 1.1: Examples of Deepfake

Nonetheless, the alarm about deepfake was raised by Barack Obama in 2018 when Buzzfeed published a fake video of him talking about his opinion on 'Black Panther' and insulting the previous US President, Donald Trump [17, 18]. The creator of this fake video was Jordan Peele, a renowned US actor or director, who generated the footage using Adobe After Effects and an AI face-swapping tool called FakeApp to match the lip movements and facial expressions of two people perfectly. This video has successfully gained public attention and went viral on social media platforms. Deepfake is more likely to target well-known people due to the large volume of easily accessible data available online, which can be used as a dataset for deep neural network training. The ripe environment of social media platforms promotes the circulation of deepfake as the deceptive hyper-photorealism falsified information that could easily gain attention and spread through the internet. Can you imagine the consequences if a deepfake video of a country leader announcing a war on another country went viral on the internet? Can you imagine seeing your face in a video talking about something that you have never said, and no one believes it was not you? The malicious use of deepfake media can exploit, sabotage, threaten, blackmail, inflict psychological harm, and damage reputations. It is a powerful technology that could lead to individual loss, social panic, or even place the world in danger when abused.

On the other hand, the proper use of deepfake can be beneficial to society. Deepfake can enhance traditional pedagogical methods to increase students' interest in learning. For instance, deepfake videos of historical figures narrating their stories directly to students during lectures can be created [7]. This teaching method is more attractive and can keep students engaged. It can also be used to resurrect deceased artists for new performances. For example, the late Peter Cushing portrayed Grand Moff Tarkin in 2016's Rogue One [19], and the famous Chinese singer Jay Chou performed with a virtual version of the long-dead mandopop icon Teresa Teng in his concert using a hologram. These performances demonstrate the potential growth of holographic deepfake [20]. Deepfake can also be utilized to create a memorial for those who have passed away or to comfort individuals who have lost their beloved ones [21]. From a medical perspective, individuals who suffer from specific forms of paralysis or physical disabilities can use deepfake technology to create a virtual engagement with others in real life to give them a sense of participation in activities they cannot take part in [7].

Despite the potential benefits of deepfake technology, the negative impacts of its misuse

are significant and outweigh the positives. As a result, it is imperative to develop a reliable deepfake detector capable of distinguishing between real and fake content. In 2019, Facebook partnered with Microsoft and various academic institutions to establish The Partnership on AI, which launched a competition called the 'Deepfakes Detection Challenge' (DFDC). The competition offered a substantial reward of up to $10 million to encourage research on deepfake detection [22]. This initiative underscores the importance and urgency of accelerating the development of deepfake detection technology.

1.1 Thesis Motivation & Methodology

The traditional handcrafted methods for deepfake detection are becoming less practical due to the continuous improvement of deepfake generation approaches. Deepfakes tend to have fewer intrinsic features and subtle traces that are typically used as hints or fingerprints for deepfake detection. As a result, the recent trend in deepfake detection has shifted towards deep learning approaches. However, most current deep learning-based detection models have several pitfalls: *i.* expensive computational costs for model training, *ii.* model is not generalized enough when dealing with unseen deepfake datasets, and *iii.* the models are less robust against adversarial attacks.

Therefore, this thesis proposes a deep learning-based deepfake detection model called SparcoNet, which is cost-efficient and focuses on local spatial feature learning and global spatial feature preservation. SparcoNet uses separable convolution, reduction block bottleneck structure, and strided-convolution to design the network feature extractor with higher receptive fields but at a lower computational cost. To further improve its performance against cross-dataset evaluation and adversarial attacks, the thesis proposes the SSL-SparcoNet, where the SparcoNet is trained using a Self-Supervised Learning (SSL) manner with different data transformations and a normalized temperature-scaled cross-entropy loss function. The idea is to improve the model generalization capability in dealing with unseen datasets via SSL training with various transformations. Additionally, the thesis implements Block Switching (BS) as a framework protector to enhance the defense capability of the SSL-SparcoNet against black-box adversarial attacks. Figure 1.2 outlines the overall architecture of the proposed methodology.

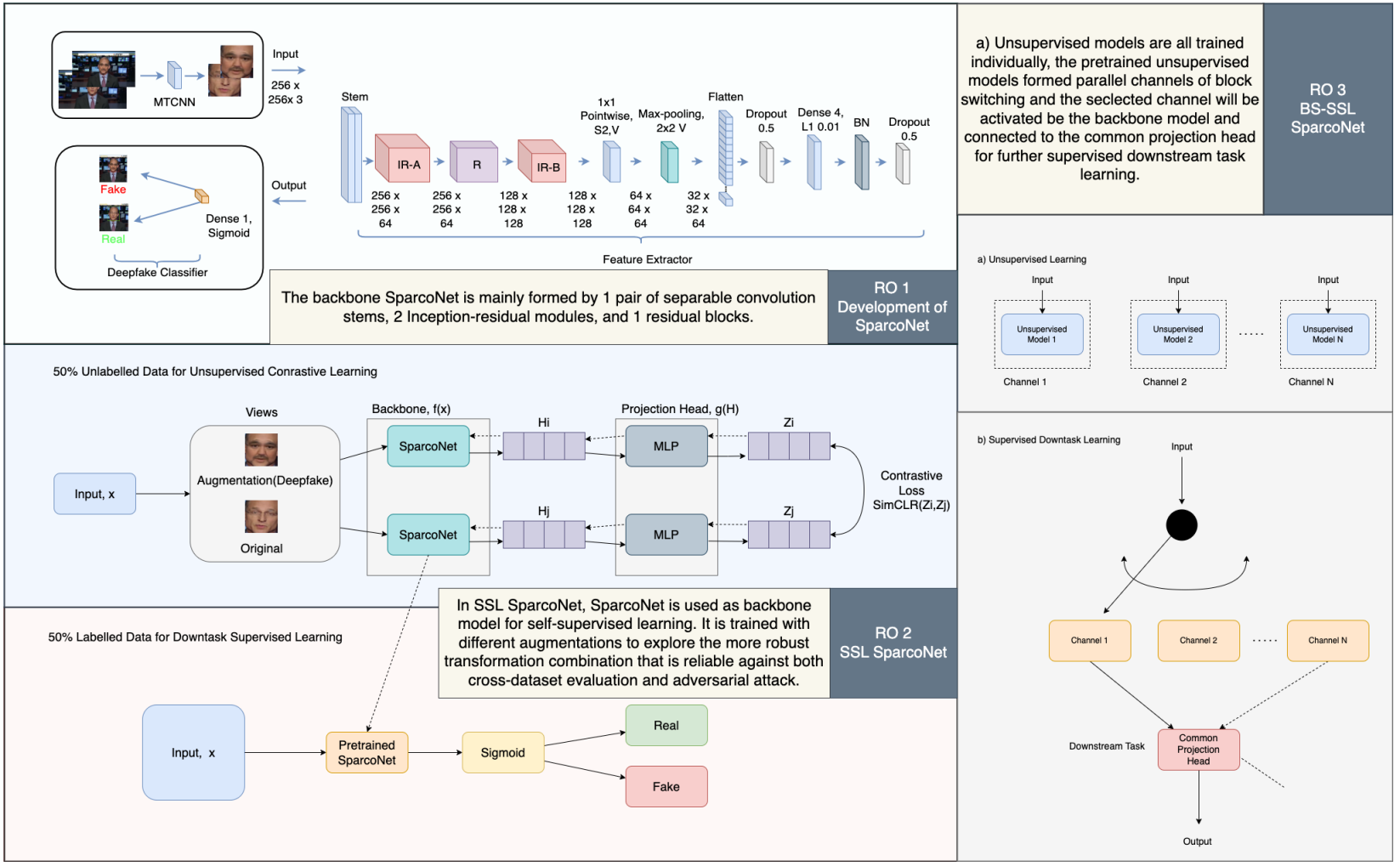


FIGURE 1.2: The overall architecture of the proposed methodology

1.2 Research Question & Research Objectives

The following questions are constructed to address the issues mentioned in the previous section:

1. How to achieve a deepfake detection model that has a low computational cost?
2. How to ensure the consistency in detection performance of the proposed deepfake detection model across different datasets?
3. How robust is the deepfake detection model in defending against adversarial network attacks?

Three objectives are formulated based on the research questions:

1. To devise a deepfake detection model with a low computational cost yet capable to achieve high detection accuracy
2. To embed a tertiary algorithm with the proposed detection model to enhance its robustness in dealing with unseen datasets
3. To integrate a defense mechanism in the proposed detection model to defend against adversarial attacks

1.3 Research Scope

This thesis aims to overcome the limitations of conventional deepfake detection approaches by developing a reliable neural network-based detector model for different types of deepfake, including entire face synthesis, face attribute manipulation, face reenactment, motion reenactment, and face-swapping. The research covers the entire deepfake detection pipeline process, including data collection, data preprocessing, model development, hyperparameter finetuning, data visualization, analysis of the tertiary framework, and support defense mechanism. The experiments implemented several datasets with different resolutions to simulate real-case scenarios of deepfake data, including FaceForensics++ [23], Deepfake Detection Dataset [24], Celeb-DF [25], DFDC Dataset [26], MesoDF Dataset [27], and Deeper-Forensics-1.0 [28]. The thesis focuses on the research

and development of image and video data. A complete BS-SSL SparcoNet will be presented at the end of the research.

1.4 Thesis Outline

This thesis comprises seven chapters, with the introduction chapter being the first. The subsequent chapters are outlined as follows:

Chapter 2 provides a broad overview of the neural network model, introducing its fundamental theory and architecture structure. This includes the basic machine learning training concepts of supervised and unsupervised learning, overfitting, underfitting, network capacity, and evaluation metrics. It also provides insight into hyperparameter configuration, such as activation functions, loss functions, optimization, regularization, learning rate, batch size, epoch, backpropagation, and gradient descent. Furthermore, the chapter briefly explains different network structures and implementation frameworks.

Chapter 3 covers the contents published in the review paper [29]. It presents a comprehensive background of deepfake technology, including the introduction of various types of deepfakes, recent progress in deepfake generation and detection methodologies, as well as its evolution and development from different perspectives. The chapter also highlights publicly available deepfake generation tools and open-source benchmark datasets. Additionally, it introduces SSL and its application in the deepfake field.

Chapter 4, 5, 6 discusses the development of the proposed BS-SSL SparcoNet deepfake detection model stage-by-stage. It presents the model concept and network architecture and investigates the network's performance under various robustness testing. Furthermore, it examines the limitations and relationship between network complexity and detection performance. These chapters include the material presented in the paper [30].

Chapter 7 concludes the thesis by summarizing the main findings of this research project and exploring future directions that can expand on this work.

Chapter 2

Fundamental of Neural Networks

This chapter provides a comprehensive introduction to neural networks, covering both theoretical concepts and practical implementation. It explains the terminology and constraints related to the training process and delves into the intricacies of hyperparameter configuration.

2.1 Introduction to Artificial Neural Network

Artificial Neural Networks (ANNs) is a machine learning model class that mimics the human brain's biological neural networks [31]. They are used to model complex, non-linear relationships between inputs and outputs, making them suitable for various applications, including facial and object detection, image and speech recognition, natural language processing, and predictive modeling.

In the 1940s, Warren McCulloch and Walter Pitts introduced the first model of a simplified neuron, which brought artificial neural networks into the public eye. In the following twenty years, the research community started developing more complex neural network models, including the perceptron for learning simple patterns. The development of this technique was still restricted during the 1980s due to the limitations in computational capabilities [32]. However, in the early 1990s, backpropagation sparked renewed interest

in the neural network. The multi-layer perceptron and convolutional neural network are two significant network architectures created with this technique. The neural networks continued to evolve with the introduction of deep learning development involving multi-layer neural networks with many hidden layers in the 2000s [32]. As computing power and data availability increase, neural networks have become more powerful and ubiquitous in solving real-case scenarios.

2.1.1 Terminology Behind Neural Networks

The ANNs are composed of interconnected nodes, called neurons, organized into layers. Each neuron takes in one or more inputs and produces an output, which is then passed on to the next layer. The overall picture of a neural network can be assumed as a weighted graph. The output of the final layer represents the network's prediction or classification for a given input. The ANNs will be trained to make accurate predictions by adjusting their weights and biases through backpropagation for network learning. It uses a mathematical optimization algorithm to minimize the difference between the predicted and actual outputs. The number of layers in a neural network can be recognized as network depth, and the number of nodes it has in each layer will be the network width [33].

Below outlines the key components of neural networks:

- **A neuron** is a basic unit in a neural network that receives one or more inputs and produces an output.
- **The input layer** of the network is the first layer of neurons that receives the input data.
- **The hidden layers** process intermediate representations of the input data and are not directly connected to the input or output layers.
- **The output layer** is the final layer of neurons that produces the output of the network.
- **The activation function** is a mathematical function that is applied to the output of a neuron to determine its final output value.

- **The weights** in the network represent the strength of the connections between neurons and are assigned numerical values like a cost.
- **Biases** are numerical values added to the output of a neuron to allow the network to learn offset values and biases in the data.
- **The backpropagation algorithm** is used to adjust the weights and biases of the network to minimize the difference between the predicted and actual outputs.
- **Gradient descent**, a mathematical optimization algorithm, is employed in backpropagation to adjust the weights and biases of the network in the direction of the steepest descent of the loss function.

The equation 2.1 outlined the formula used to compute the single output of the neural network. For a multi-layer neural network, the output of one layer will serve as the input to the subsequent layer, and the whole process is repeated until the final output is obtained. The formula of the multi-layer network is illustrated in equation 2.2. Let x be the input, w be the weight matrix, b be the bias, z be the output, and σ be the activation function applied element-wise to the weighted sum of the input and bias. The formula can be computed as follow:

$$z = \sigma(x \cdot w + b) \tag{2.1}$$

$$z = \sigma(\sum_{i=1}^{n} \mathrm{x}_i \cdot \mathrm{w}_i + b) \tag{2.2}$$

Figure 2.1 illustrates a neural network with its corresponding matrix calculation representation. By applying equation 2.1, the output of the input layer, which serves as the input of hidden layer 1, can be obtained. Since the network has multiple layers, it is necessary to compute equation 2.2 to complete the network calculation and obtain the final output z. In this example, the network depth will be 3, with a maximum network width of 4 and minimum network width of 1.

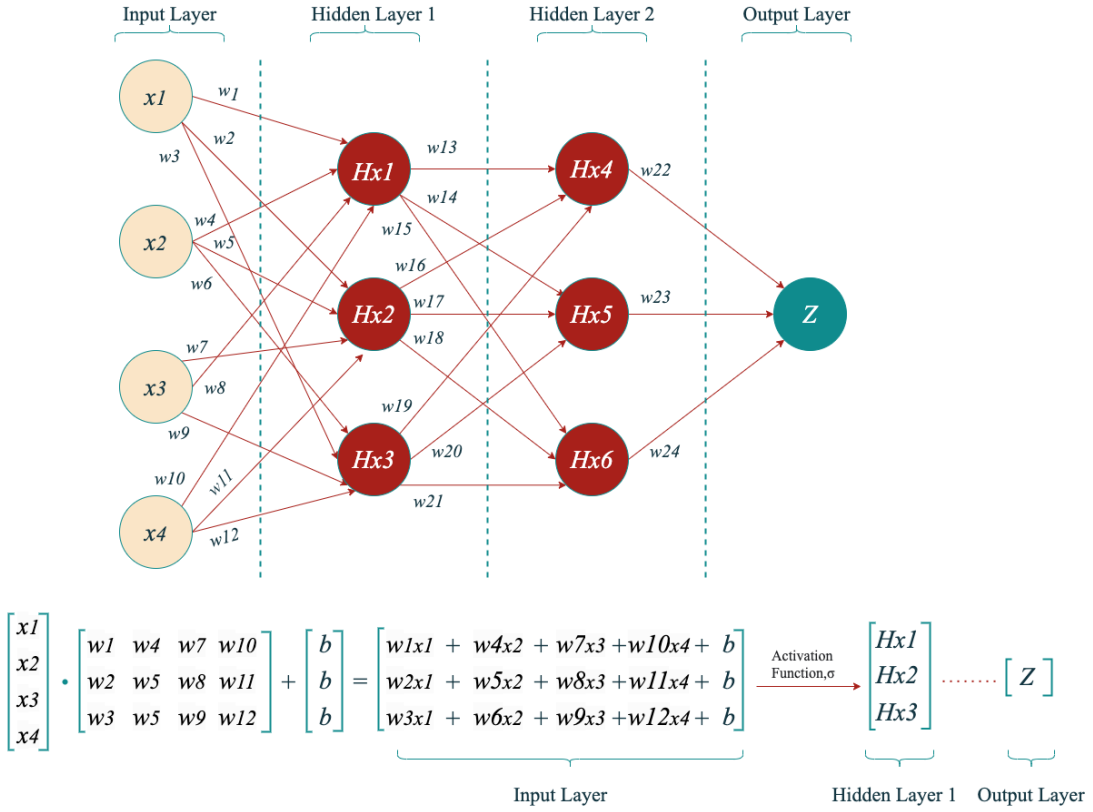


FIGURE 2.1: Network illustration and matrix calculation representation of neural networks

2.2 Type of Neural Network

There are different types of neural networks, where each designed to solve specific problems. They can be differentiated by their structure design, network depth, data flow, and neurons used. The sections below provided a brief introduction to each of the common types of neural networks.

2.2.1 Percepton

Perceptron, also known as TLU (threshold logic unit) is one of the smallest and oldest neuron models proposed by Minsky *et al.* [34]. It is a simple model that receives inputs with weights assigned to them, applies an activation function, and outputs the result as the final output. As a binary classifier, the Perceptron is a supervised learning algorithm that separates data into two categories. Figure 2.2 illustrates the representation of percepton network.

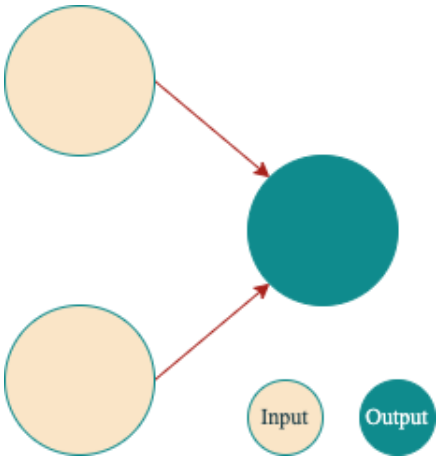


FIGURE 2.2: Percepton network illustration

2.2.2 Feed-Forward Neural Network

A feed-forward neural network processes information in a forward direction only, moving from input to output without any feedback loops [35]. While optional, the network may include hidden layers, but it always has input and output layers. The network uses fixed weights and an activation function that classifies inputs to generate an output. Neurons activate when their value exceeds the threshold, typically set to 0, and produce an output of 1. Conversely, neurons do not activate when their value falls below the threshold, typically 0, which is considered as -1. Feed-forward neural networks are widely used in various applications, including image and speech recognition, natural language processing, and predictive modeling. Figure 2.3 presents the representation of the feed-forward neural network.

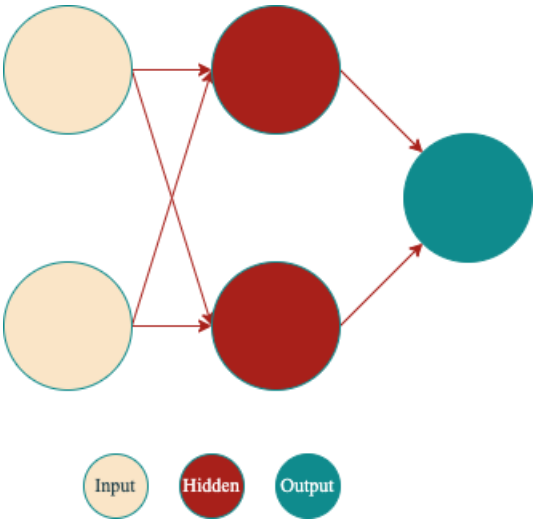


FIGURE 2.3: Feed-Forward neural network illustration

2.2.3 Multilayer-Percepton (MLP)

Multi-Layer Perceptron (MLP) is a feed-forward neural network with at least a total of three or more layers [36]. The hidden layers in an MLP comprise neurons that process information from the preceding layer and transmit it to the next layer until the output is generated. Each neuron in an MLP is associated with a set of weights and biases that are adjusted during training using backpropagation. Backpropagation updates the weights by utilizing the discrepancy between the predicted and actual outputs. MLPs are helpful for classification or regression tasks that entail intricate, non-linear relationships between the input and output variables. Figure 2.4 outlines the representation of the MLP network.

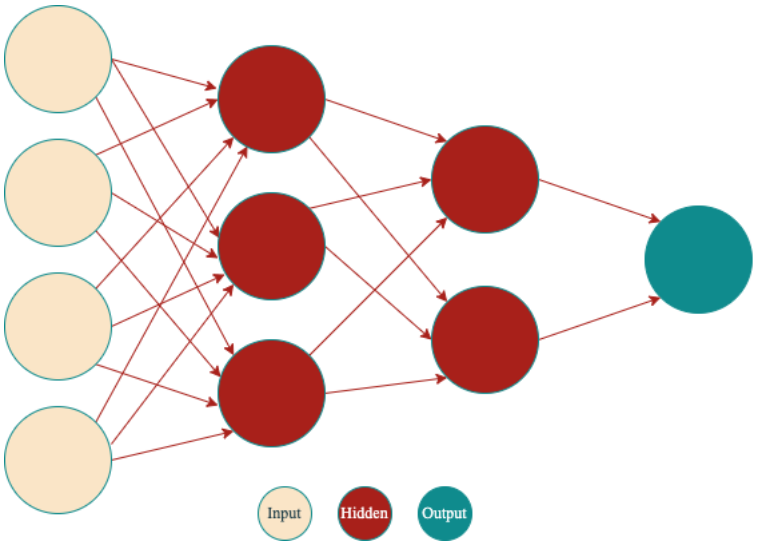


FIGURE 2.4: MLP network illustration

2.2.4 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a neural network with three dimensions: width, depth, and channel. It consists of convolutional, pooling, and fully connected layers [37]. The convolutional layers detect features like edges or shapes by applying filters to the input. Pooling layers reduce the network parameters and feature map dimensions, while fully connected layers classify the output based on the features extracted by previous layers. Due to the fact that each neuron in the convolutional layer handles information from a small portion of the visual field, the network is required to perform computations multiple times to process the entire image. CNNs have proven to

be effective in image recognition, object detection, and autonomous driving applications, among others. Figure 2.1 illustrates the representation of the CNN network.

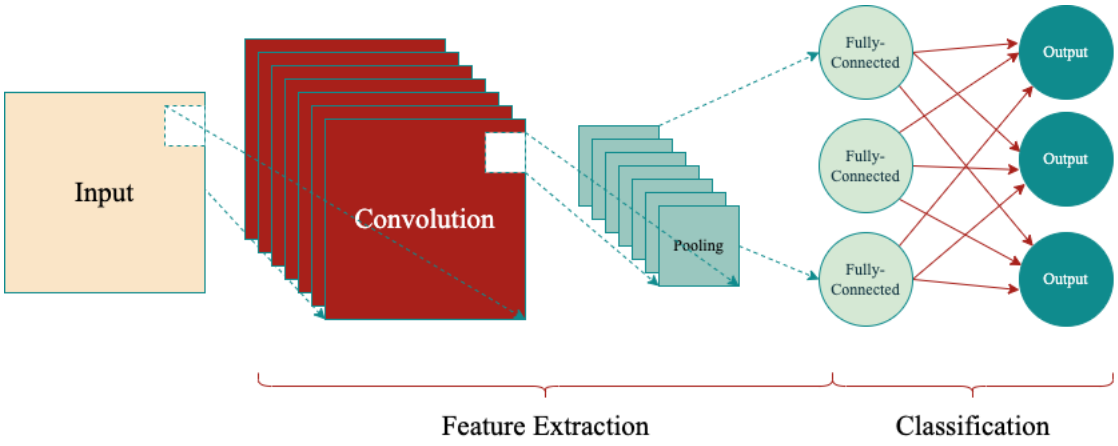


FIGURE 2.5: CNN network illustration

2.2.5 Recurrent Neural Network (RNN) & Long Short-Term Memory (LSTM)

Recurrent Neural Network (RNN) is a type of neural network that can process sequential data by retaining information from previous inputs [38]. It has a feedback loop that allows information to persist over time. This architecture design makes it useful to capture long-term dependencies in the data, such as tasks in natural language processing, speech recognition, and time series analysis. However, traditional RNNs suffer from the vanishing gradient problem, which occurs when the gradients required to update the parameters of the network become very small, making it difficult to train the network. Figure 2.6 presents the representation of the RNN network.

Long Short-Term Memory (LSTM) is a type of RNN that addresses this problem by introducing a gating mechanism that allows the network to forget or remember information from the previous time steps selectively [38]. This design enables LSTMs to learn long-term dependencies more effectively than traditional RNNs. Three gates control the flow of information in LSTM: the forget gate, input gate, and output gate. The forget gate determines what information to discard from the previous cell state, while the input gate decides what new information to add to the cell state. The output gate controls what information to output from the cell state. By selectively controlling the

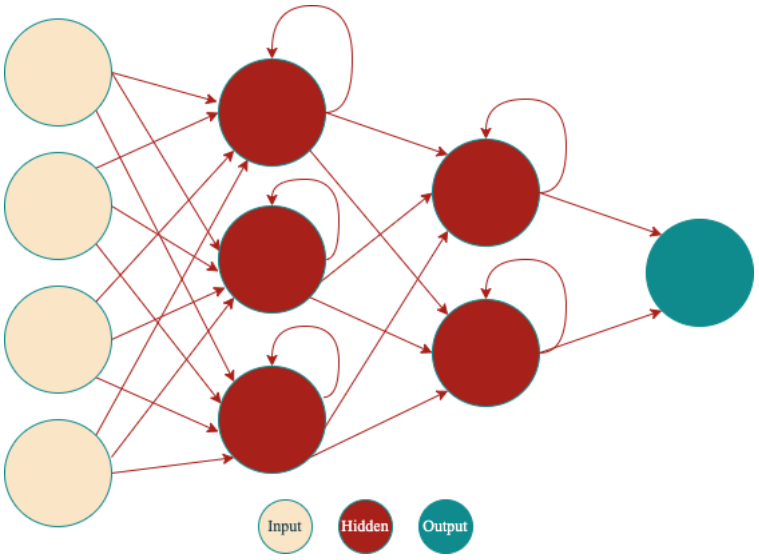


FIGURE 2.6: RNN network illustration

gates, an LSTM can effectively remember or forget information from previous time steps as needed. Figure 2.7 shows the representation of the LSTM network.

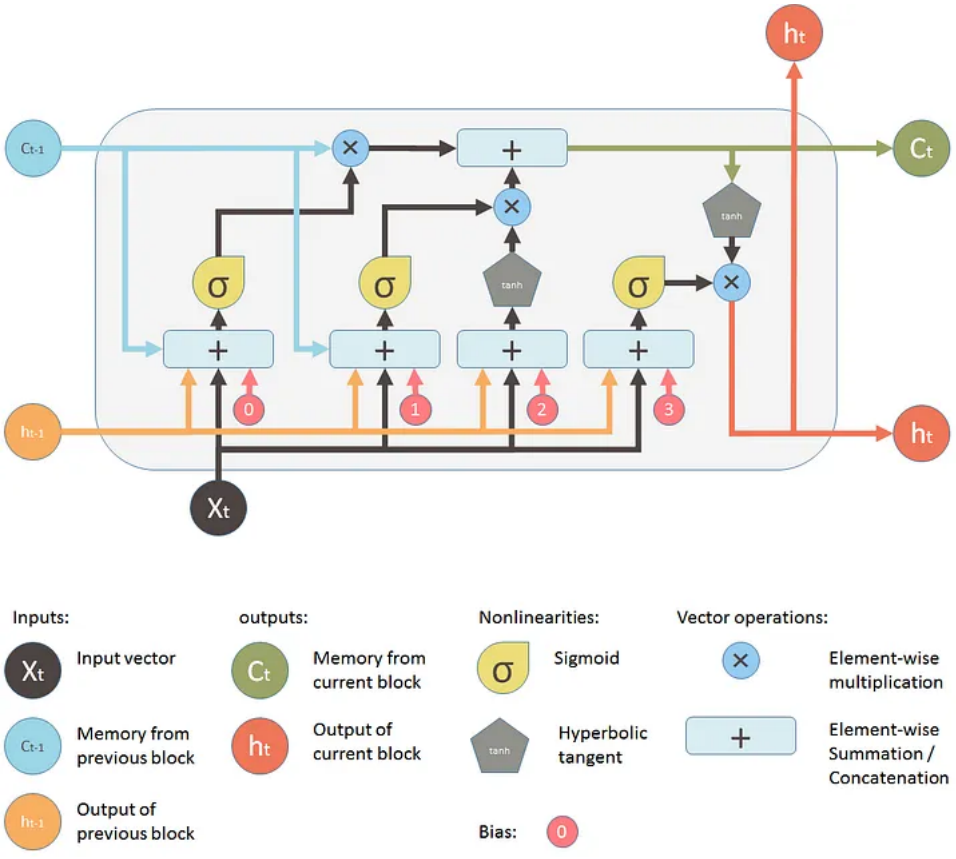


FIGURE 2.7: LSTM network illustration [1]

2.3 Neural Network Training

Unlike other machine learning-based algorithms, a neural network does not require a set of programming rules for output prediction. However, it involves an iterative training process to optimize performance by learning the feature patterns and relationships in the data. There are several types of learning in neural networks, including supervised learning, unsupervised learning, self-supervised learning, semi-supervised learning, and transfer learning.

2.3.1 Supervised Learning

Supervised learning involves training a neural network with labeled data, where the inputs are associated with known output labels. During training, the neural network learns to map the inputs to the correct output labels. This type of learning is used in applications such as image classification, speech recognition, and natural language processing [39]. Classification and regression are two significant types of supervised learning. For example, a neural network can be trained on a labeled image dataset for image classification, such as each image being labeled with a corresponding class: dog, cat, fish, and others. The neural network learns to recognize the related feature representations of each class and uses them to predict unseen data. However, supervised learning requires a large amount of labeled data for training, which might not be friendly for tasks where the required labels are difficult or expensive to obtain. This trade-off limited its flexibility for adapting the model when dealing with a new task or data variation. Figure 2.8 shows an example of supervised learning in a shape classification task.

2.3.2 Unsupervised Learning

In contrast to supervised learning, the network is trained with an unlabeled dataset for unsupervised learning. Unsupervised learning aims to discover the underlying patterns and feature relationships in the input data without prior knowledge of the desired output [39]. It is commonly used to perform clustering, dimensionality reduction, and anomaly detection tasks. For example, in a clustering task, the network model will be trained to group the input data based on their feature patterns without knowing the output labels.

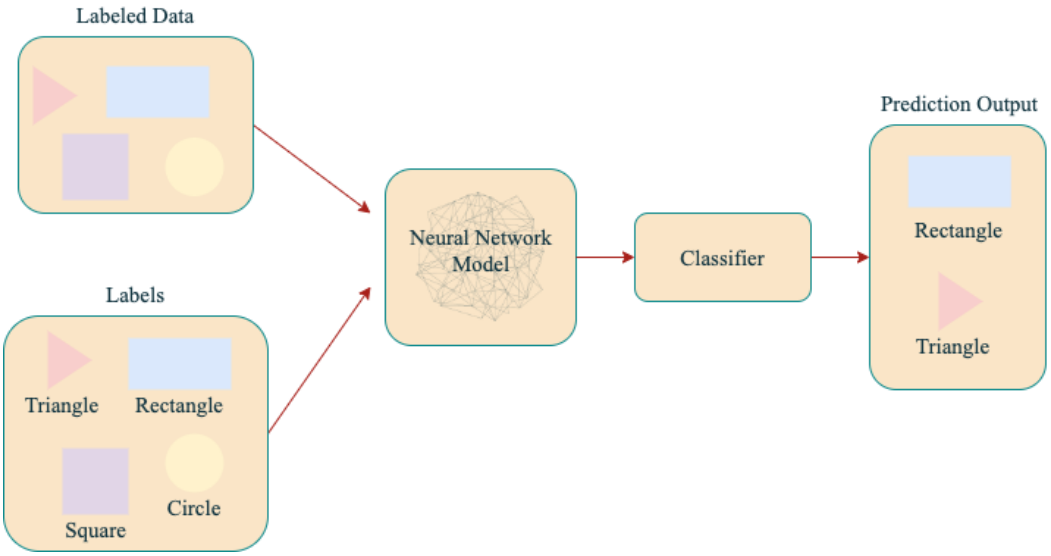


FIGURE 2.8: Example of Supervised Learning

The distance between the feature representation of similar input will be closer than the input with different classes. The neural network adjusts its weights and biases during training to minimize the chosen objective function. Unsupervised learning is valuable for uncovering unforeseen patterns or anomalies in data. Nevertheless, the training time for this technique may be longer, as it necessitates a larger dataset to learn from the raw data to enhance the accuracy of output predictions. Figure 2.9 illustrates the example of unsupervised learning in a shape clustering task.

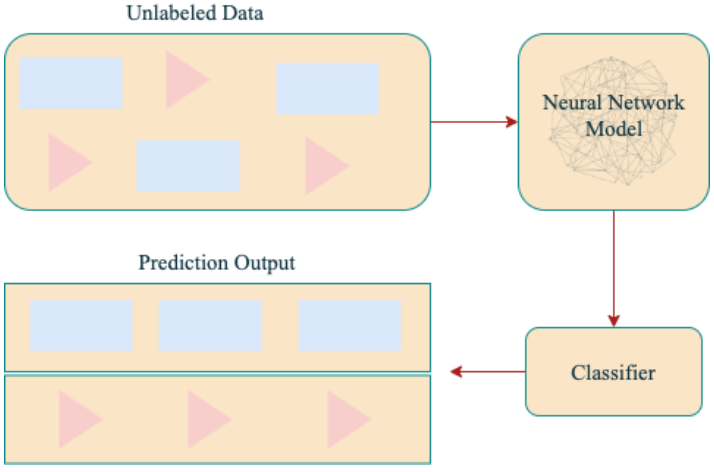


FIGURE 2.9: Example of Unsupervised Learning

2.3.3 Semi-supervised Learning

Semi-supervised learning involves training a network with both labeled and unlabeled datasets instead of relying on only one of them. This technique is often employed to address the expensive labeling issue that arises in supervised learning [39]. Initially, the model undergoes supervised training using the labeled dataset, and subsequently, the trained model is used to predict the label for the unlabeled dataset. The unlabeled data that receives high-confidence predictions is then added to the labeled dataset, and the model is retrained on the expanded dataset. This iterative process is repeated, with the model becoming more accurate as it incorporates more labeled data [40]. However, if there is only a limited number of labeled data for initial training, the model has a high opportunity of encountering overfitting and outputting false pseudo-labels, which might lead the model to perform in a different direction. Figure 2.10 illustrates the example of semi-supervised learning in a shape classification task.

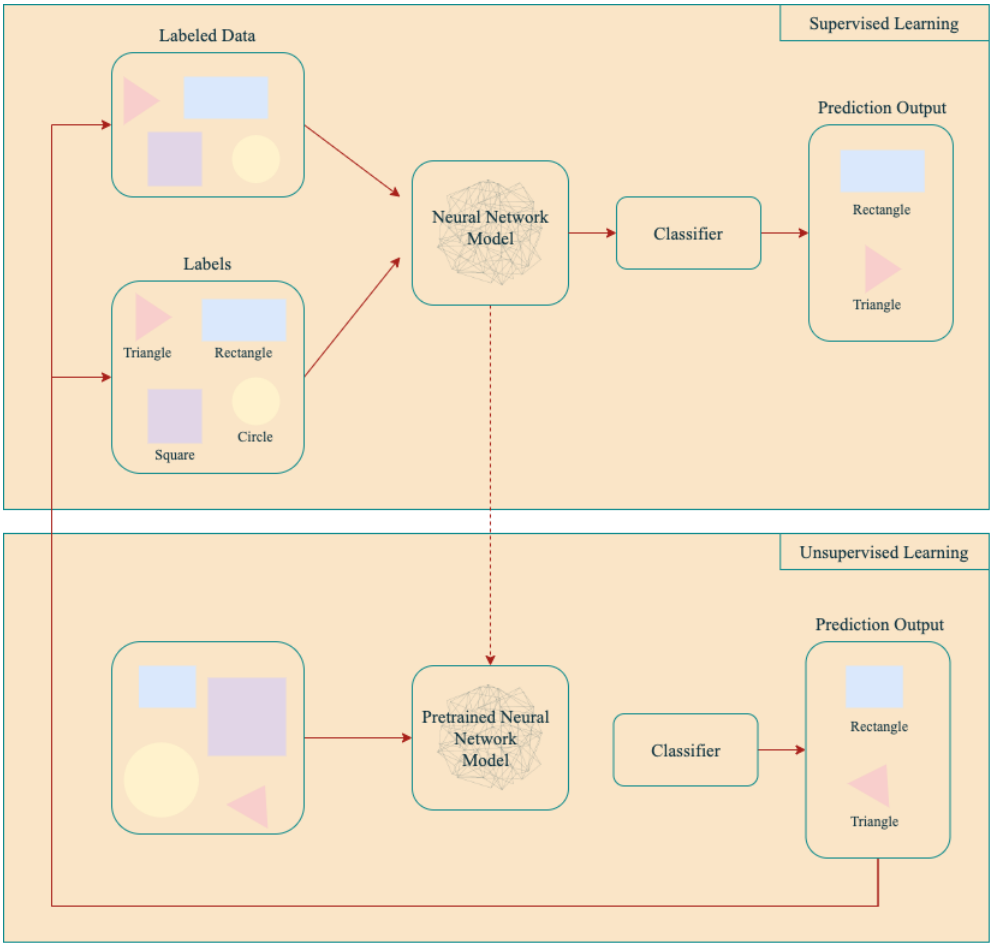


FIGURE 2.10: Example of Semi-supervised Learning

2.3.4 Self-supervised Learning

Self-supervised learning is a machine learning technique that falls under the umbrella of unsupervised learning. It begins by performing self-supervised pretext task learning using an unlabeled augmented dataset. The augmented data from the same images are considered positive pairs, while different images are negative pairs. During training, the model is designed to push the learned features of negative pairs away while bringing the positive features closer together. The aim is to allow the model to learn the intermediate representation of the data [40]. Through self-supervision of the pretext task, the model can predict a subset of the input data from the remaining part of the input data, with examples including inpainting, super-resolution, and classification tasks.

Subsequently, the pre-trained model can be further fine-tuned through supervised training using labeled data for downstream task classification, thereby enhancing the overall performance of the model. Figure 2.11 illustrates the example of self-supervised learning in a shape classification task.

2.3.5 Transfer Learning

Transfer learning can also be referred to as domain adaptation, wherein a model trained for a task is reused to transfer knowledge to a different yet related task [41]. Instead of developing from scratch and training a new model for each task, transfer learning allows us to reuse the pre-trained model, which reduces effort, time, and computational resources while achieving higher performance on the target task. This approach has been successfully applied to various domains, including natural language processing, computer vision, and speech recognition, making it a valuable tool for many real-world applications. Figure 2.12 shows the example of transfer learning in adapting a 2D shape classification task to a 3D shape classification task.

2.3.6 Overfitting & Underfitting

Overfitting and underfitting are common issues that often occur during model training. They reflect how the model deals with the training dataset. Underfitting is often characterized by high bias and low variance, whereas overfitting is characterized by low bias but high variance [42].

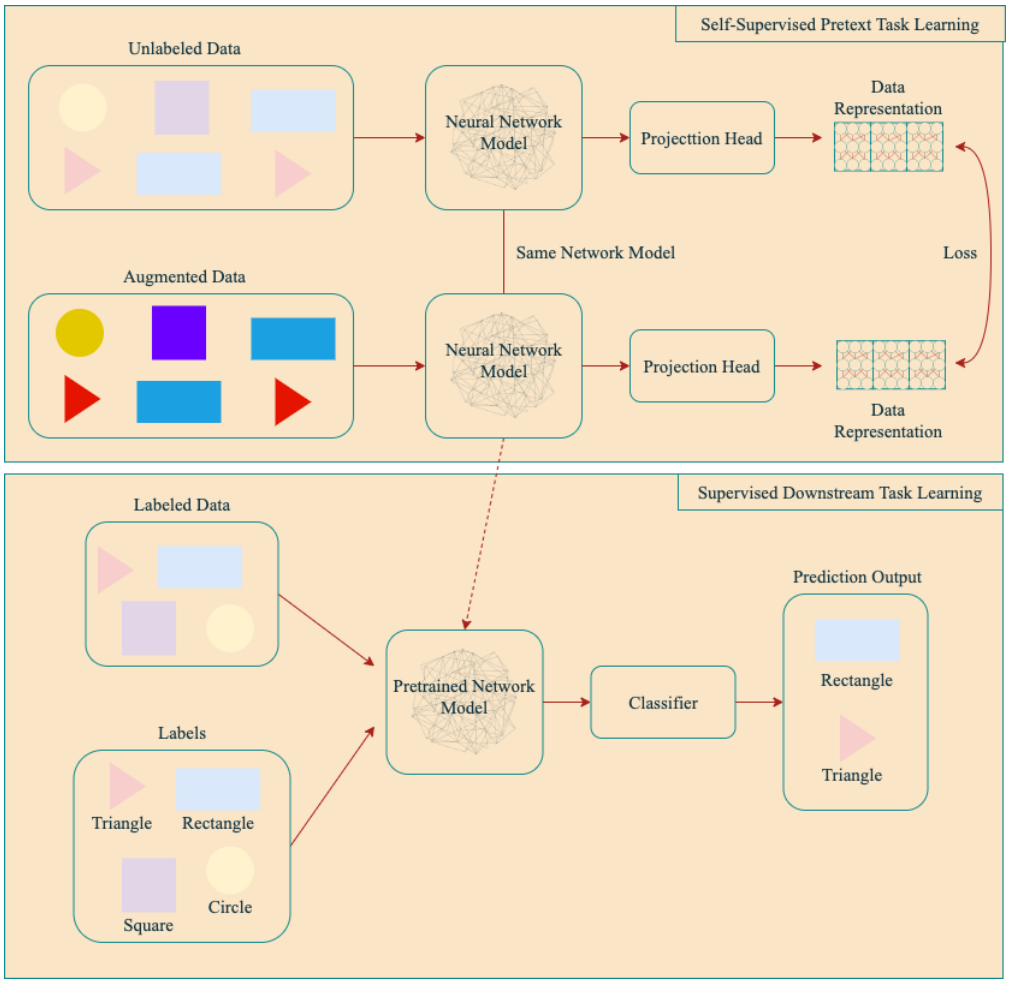


FIGURE 2.11: Example of Self-supervised Learning

Overfitting occurs when a model is overly complex and fits the training data too closely, resulting in poor generalization to unseen data [33]. This situation can arise when a model is trained for too long or has too many trainable parameters, causing it to become too specialized to the training data and compromising its ability to generalize to new data. This issue can be detected by evaluating the model's performance on a separate validation set, where it may exhibit high accuracy on the training set but significantly lower accuracy on the validation set or a low training loss but high validation loss [42].

Conversely, underfitting occurs when a model is too simplistic and fails to capture the underlying patterns in the data, resulting in poor performance on both the training and validation sets. This situation is in contrast to overfitting, as it occurs when the model capacity is too shallow and insufficient to learn the data features. Similar to overfitting, underfitting can be detected by evaluating the model's performance on the validation set, where it may exhibit both low training and validation accuracy or when the training

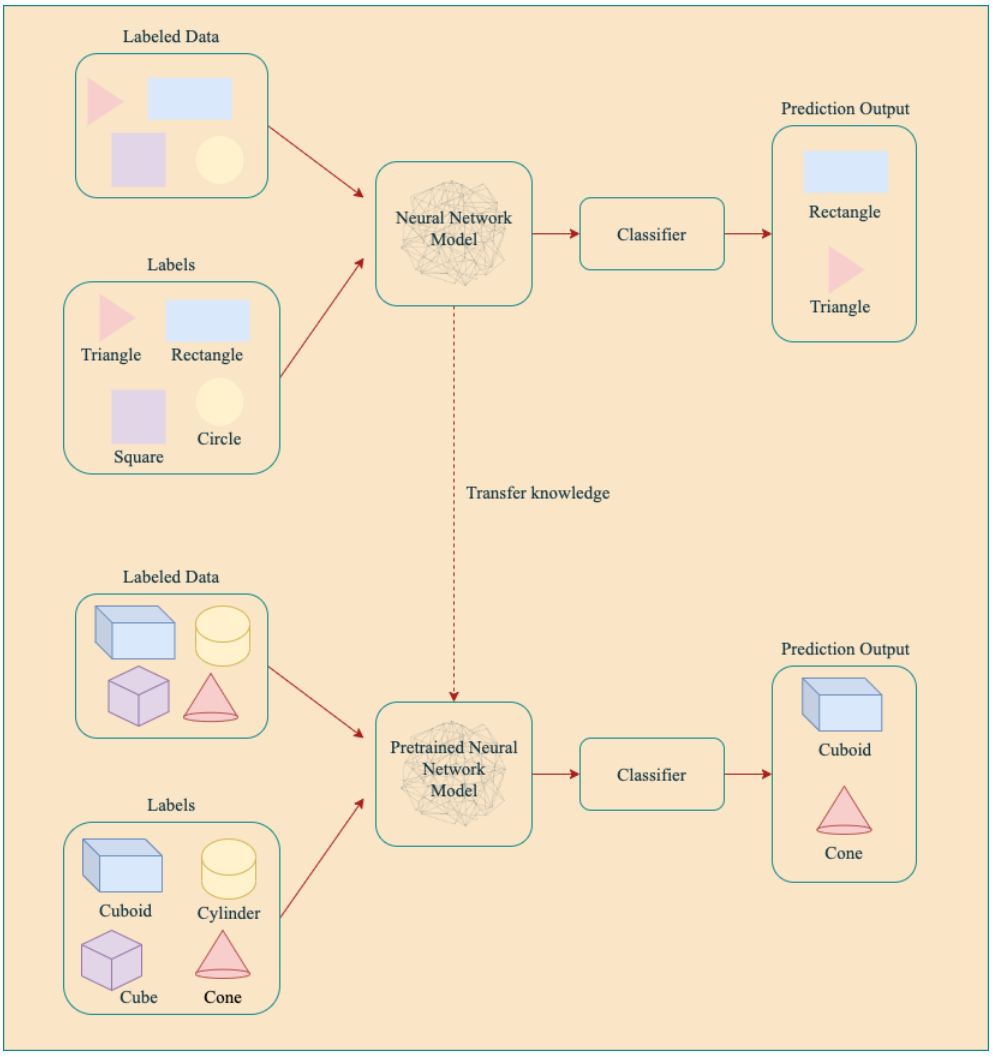


FIGURE 2.12: Example of Transfer Learning

loss is much higher than the validation loss [42]. Figure 2.13 illustrates the trade-off between bias and variance [2]. Data that underfits the model have less complexity but show a high error rate and high bias (blue box). Conversely, overfitting data lead to low bias but high variance (yellow box). The optimal zone lies between overfitting and underfitting data and may require several testing attempts to achieve the best results (red line).

Hence, it is crucial to finetune the model to ensure its complexity is suitable and appropriately fits the training dataset. Apart from examining the model architecture, the finetuning process involves the configuration and optimization of hyperparameters as well as regularization techniques.

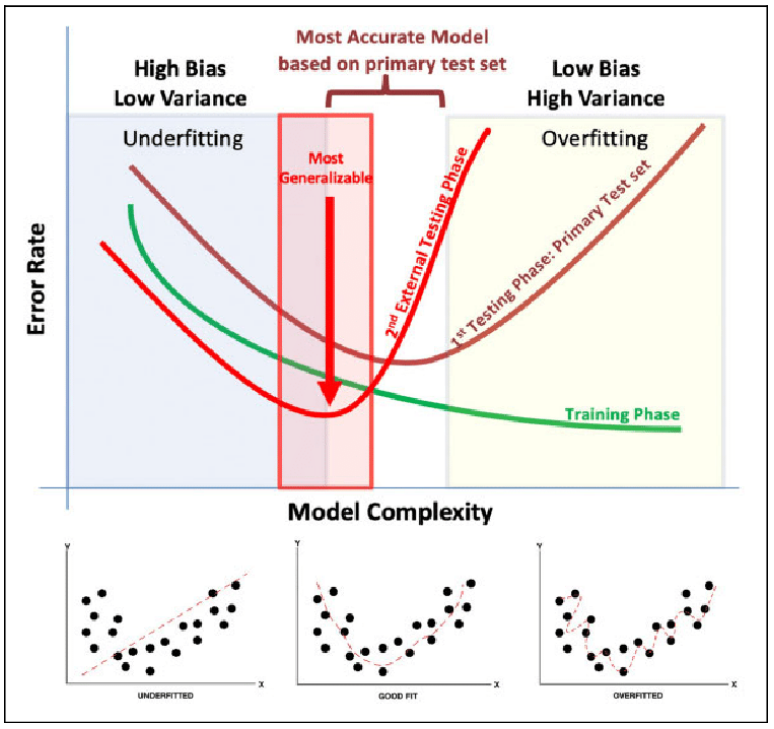


FIGURE 2.13: Trade-off between Variance and Bias [2]

2.3.7 Regularization

Regularization is the technique used to prevent model overfitting to the training dataset for improving the model's generalization against unseen data [37]. Examples of regularization techniques are:

- **L1 (Lasso Regression) and L2 (Ridge Regression) regularization** that add a penalty term to the loss function to ensure smaller weights. L1 regularization encourages sparse weights, while L2 regularization encourages small but non-zero weights.
- **Dropout** is a technique that randomly drops out units from the network during training. It forces the network to learn with a subset of the units at each training iteration.
- **Batch normalization** normalizes the activations of each layer in the network during training. It reduces the internal covariate shift to stabilize the gradients during backpropagation and improve the training speed and network performance.
- **Data augmentation** is a technique that expands the training set's size by applying random transformations to the data, such as rotation and splicing.

- **Early stopping** stops the training before the model degrades by monitoring the model performance on the validation set during training.

Since regularization is meant to avoid overfitting, it is worth mentioning that reducing the regularization strength can help to prevent underfitting as well.

2.4 Parameter Configuration

Parameter configuration refers to presetting the optimal values for the hyperparameters of a neural network model before training. The hyperparameters are parameters that cannot be learned from the data but are set before the training begins. The explanation of the common hyperparameter used in neural network training is explained below [43].

- **Learning rate** determines the step size at which the optimizer updates the weights during training. A greater learning rate can result in faster convergence but may cause the model to overshoot the optimal solution. A lower learning rate may result in slower convergence but produce a more accurate model.
- An **epoch number** determines the number of times the model will iterate over the entire dataset during training. Increasing the number of epochs may result in better accuracy but can also lead to overfitting.
- **Batch size** controls the number of training examples used in each update of the model weights. A larger batch size may result in faster training but may also lead to poor generalization. In contrast, a smaller batch size may lead to slower training but can result in a better-performing model.
- **Hidden layers mumber** specifies the number of hidden layers in the neural network. Increasing the number of hidden layers allows the model to capture more complex patterns in the data. However, it can also lead to overfitting and longer training time.
- The **dropout rate** controls the probability of dropping out a neuron during training. It is a regularization technique that can prevent overfitting by randomly dropping out neurons during training. A higher dropout rate can result in a more regularized model but can also lead to underfitting.

- **Weight decay** specifies the penalty added to the loss function for large weights. It is a regularization technique that can prevent overfitting by adding a penalty for large weights. A larger weight decay coefficient can result in a more regularized model but can also lead to underfitting.

The process of hyperparameters configuration can be time-consuming and computationally expensive, as it involves iterative training and testing processes to evaluate the model's performance until an optimal set of configurations is achieved. Since the optimal configuration of hyperparameters is specific to the task at hand, it is rare for a configuration to be shared among different tasks.

2.5 Activation Function

Activation functions are functions that are applied to the output of each neuron in a neural network. They introduce non-linearity to the output of the neuron, which allows the neural network to capture more sophisticated relationships between inputs and outputs. Activation functions are used to determine whether a neuron should be activated during training. Below presents the equation and explains the popular activation functions used in neural networks [44].

Linear activation is the simple function where the input is directly proportional to the output and is commonly used in the output layer for regression tasks. The formula is presented in Equation 2.3 below:

$$f(x) = x \tag{2.3}$$

, where x is the input data

Sigmoid activation maps the input to a range between 0 and 1, which can be interpreted as a probability of belonging to a particular class. The formula is presented in Equation 2.4 below:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

, where x is the input data

Softmax activation commonly used in the output layer for multi-class classification problems. It normalizes the output of the network to a probability distribution over the classes. The formula is presented in Equation 2.5 below:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2.5}$$

, where x_i is the output of the i-th neuron in the output layer, K is the total number of classes, and f(x_i) is the probability of the input belonging to the i-th class.

Rectified linear unit (ReLU) activation sets the output to zero if the input is negative and passes the input through if it is positive. The formula is presented in Equation 2.6 below:

$$f(x) = max(0, x) \tag{2.6}$$

, where x is the input data

Leaky Rectified linear unit (Leaky ReLU) activation is a variant of the ReLU activation function that addresses the "dying ReLU" problem where the ReLU activation function can lead to dead neurons with a zero output. The Leaky ReLU activation function introduces a small slope for negative input values to prevent the neuron from completely dying. The formula is presented in Equation 2.7 below:

$$f(x) = max(x, x \cdot 0.01) \tag{2.7}$$

, where x is the input data

Hyperbolic Tangent (tanh) activation function maps the input to a range between -1 and 1. The formula is presented in Equation 2.8 below:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.8}$$

, where x is the input data

As a general guideline, the ReLU activation function is a good starting point and can be used in the hidden layers. If ReLU does not yield optimal results, other activation functions can be tried. However, it should only be implemented in the hidden layer. In contrast to ReLU, the Sigmoid and Tanh functions are not suggested to be used in

the hidden layers due to their higher likelihood of causing vanishing gradients issues [3]. Figure 2.14 shows the graph presentation of the activation functions.

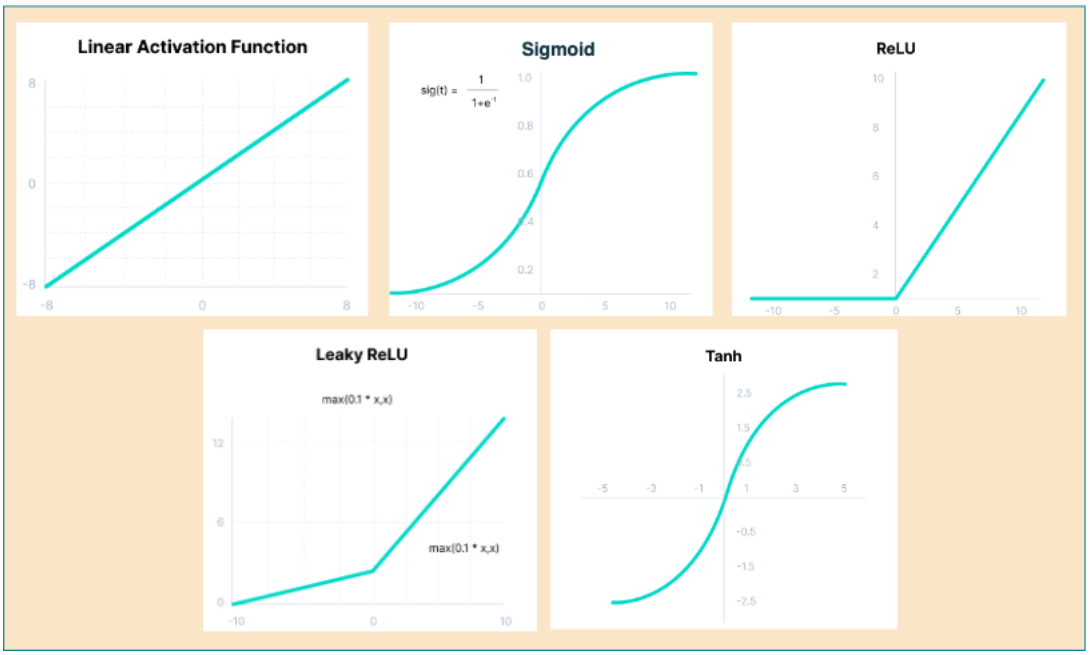


FIGURE 2.14: Graph representation of activation functions [3]

2.6 Loss Function

The loss function, also known as a cost function, is designed to measure the training performance of a neural network model. The objective is to minimize the loss value between the predicted output and the actual ground truth. The lower the loss value, the better the model's performance on the given task. To optimize the loss value, gradient descent is implemented to adjust the weights and biases of the network during the training process. This helps the model improve its performance by gradually reducing the loss value.

Mean Squared Error (MSE) measures the average squared difference between the predicted and actual output. It is commonly used for Regression tasks. Equation 2.9 presents the formula for computing MSE.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.9}$$

, where y_i is the actual output for the i^{th} example, $\hat{y}_i$ is the predicted output for the i^{th} example, and n is the number of examples.

Mean Absolute Error (MAE) is similar to the MSE, but it measures the average absolute difference between the predicted and actual output. It is commonly utilized for Regression tasks as well. Its formula is presented in Equation 2.10 below.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.10}$$

, where y_i is the actual output for the i^{th} example, $\hat{y}_i$ is the predicted output for the i^{th} example, and n is the number of examples.

Binary Cross-Entropy measures the difference between the predicted and actual output, taking into account the probability of each class. It is normally implemented for Binary classification. The Equation 2.11 is outlined below.

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)] \tag{2.11}$$

, where y_i is the actual output for the i^{th} example, $\hat{y}_i$ is the predicted output for the i^{th} example, and n is the number of examples.

Categorical Cross-Entropy measures the difference between the predicted and actual output, taking into account the probability of each class. It is suitable for Multi-class classification. The Equation 2.12 is presented below.

$$CCE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} \log \hat{y_{ij}} \tag{2.12}$$

, where y_{ij} is the actual output for the i^{th} example and j^{th} class, $\hat{y_{ij}}$ is the predicted output for the i^{th} example and j^{th} class, n is the number of examples, and m is the number of classes.

Kullback-Leibler Divergence (KL divergence) measures the difference between two probability distributions and is normally used for Generative models. The Equation 2.13 is presented below.

$$KL = \sum_{i=1}^{n} y_i \log \frac{y_i}{\hat{y}_i} \tag{2.13}$$

, where y_i is the actual output for the i^{th} example, $\hat{y}_i$ is the predicted output for the i^{th} example, and n is the number of examples.

2.7 Gradient Descent

Backpropagation is an algorithm that computes the gradients of the loss function for the weights and biases of the neural network. Meanwhile, gradient descent is an optimization algorithm that uses the gradients computed by backpropagation to minimize the value of a loss function by iteratively adjusting the model parameters. The idea is to find the direction of the steepest descent in the loss function and follow that direction to reach the minimum value of the loss function. At each iteration, the gradient is computed to update the parameters [45]. The formula to update the parameters in gradient descent is:

$$\theta = \theta - \alpha * \nabla L(\theta) \tag{2.14}$$

, where θ is the parameter, α is the learning rate, and $\nabla L(\theta)$ is the gradient derivative of the loss function with respect to the parameter.

2.7.1 Batch, Mini-Batch, and Stochastic Gradient Descent

There are different variants of gradient descent, such as batch gradient descent, stochastic gradient descent, and mini-batch gradient descent [46]. Batch gradient descent computes the gradient on the entire training dataset and updates the model parameters once per epoch. While stochastic gradient descent computes the gradient on a single data point at a time, it updates the model parameters for every sample. Mini-batch gradient descent computes the gradient on a small batch of data points.

Batch gradient descent is a more efficient algorithm for small datasets and can ensure convergence to the global minimum of the loss function. However, it can slow down the training process since it needs to process the entire dataset in each iteration and is more likely to get trapped in local minima. On the other hand, SGD is more computationally efficient than batch gradient descent as it processes one example at a time, making it capable of escaping from local minima. Nevertheless, it does not converge well as batch gradient descent. Mini-batch gradient descent strikes a balance between the efficiency

of SGD and the robustness of batch gradient descent, but its implementation is more complicated [31].

2.7.2 Optimizers

To overcome the limitations of the common gradient descent algorithm (SGD), such as slow convergence and the tendency to get stuck in local minima, optimizers modify the basic algorithm by adding adaptive learning rates, momentum, and other features to improve convergence and training speed [37]. Below explains the other optimizers that commonly be implemented in training the neural network:

Momentum optimizer accelerates SGD in the relevant direction and dampens the oscillations [37]. It adds a fraction of the update vector from the previous step to the current update vector, which allows it to speed up in directions where the gradient is consistently pointing. The formula of the Momentum optimizer is:

$$v = \beta v + (1 - \beta)\nabla L(w) w = w - \alpha v \tag{2.15}$$

, where β is the momentum coefficient, typically set to 0.9.

Adagrad optimizer adapts the learning rate of each parameter based on the historical gradients for that parameter [37]. It uses a per-parameter learning rate inversely proportional to the square root of the sum of the squares of the gradients for that parameter. The formula of the Adagrad optimizer is:

$$g_t = \nabla L(w_t) s_t = s_{t-1} + g_t^2 w_{t+1} = w_t - \frac{\alpha g_t}{\sqrt{s_t} + \epsilon} \tag{2.16}$$

, where g_t is the gradient at time t, s_t is a running sum of the squares of the gradients, and ϵ is a small constant to avoid division by zero.

RMSProp optimizer is an adaptive learning rate optimizer that uses a moving average of the squared gradients to normalize the learning rate for each parameter [37]. RMSProp optimizer helps to determine an appropriate learning rate for different parameters. The formula of the RMSProp optimizer is:

$$g_t = \nabla L(w_t) s_t = \beta s_{t-1} + (1 - \beta) g_t^2 w_{t+1} = w_t - \frac{\alpha g_t}{\sqrt{s_t} + \epsilon} \tag{2.17}$$

, where g_t is the gradient at time t, s_t is a running average of the squared gradients, β is the decay rate, typically set to 0.9, ϵ is a small constant to avoid division by zero.

Adam optimizer combines the benefits of both momentum and adaptive learning rate methods [37]. It maintains a running average of both the gradients and the second moments of the gradients and uses these estimates to compute the adaptive learning rates for each parameter. It also includes a bias correction term to account for the initialization of the moving averages. The formula of the Adam optimizer is:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \hat{m}_t = \frac{m_t}{1-\beta_1^t} \hat{v}_t = \frac{v_t}{1-\beta_2^t} w_{t+1} = w_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.18}$$

, where g_t is the gradient at time t, m_t and v_t are the first and second moments of the gradients, β_1 and β_2 are the decay rates for the moments, typically set to 0.9 and 0.999, ϵ is a small constant to avoid division by zero. The terms $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates of the moments.

2.7.3 Vanishing Gradient & Exploding Gradient

Vanishing gradients and exploding gradients are two problems commonly occurring during deep neural network training [37]. Vanishing gradients occur when the gradients in the early layers of the network become very small as they propagate backward through the network during backpropagation. This situation can make it difficult for these layers to learn useful features, as the small gradients prevent significant updates for the weight. The issue of vanishing gradients could be problematic in deep neural networks, as the gradients can become exponentially smaller with each layer. Exploding gradients occur when the gradients in the early layers of the network become very large during backpropagation. This can cause the weights to update too much, leading to unstable and divergent behavior during training.

These problems could lead to poor performance and slow convergence during training. However, there are several techniques that can be used to mitigate these issues [37]. Batch normalization is a common technique used to address both issues. Additionally, residual connections and non-saturating activation functions, such as ReLU and Leaky ReLU, can be used to prevent vanishing gradients. Weight initialization and gradient clipping are also suggested for implementation to avoid exploding gradients.

2.8 Development Framework

Developing ANNs from scratch can be challenging, but fortunately, there are two widely used open-source frameworks available for building deep neural networks: TensorFlow [47] and PyTorch [48]. Both frameworks offer user-friendly interfaces and rich library packages that can ease the development process.

TensorFlow is an end-to-end, open-source, dataflow graph-based framework developed by the Google team. With the comprehensive, flexible ecosystem of tools and libraries, it provides better visualization and supports users in developing large-scale distributed models across various devices and platforms.

PyTorch, on the other hand, was released by Facebook. It implemented the popular computing framework, Torch library, to provide a dynamic computational graph. It allows for more flexibility and ease of use when building and debugging models.

The selection of the development framework fully depends on the specific use case and personal preferences, as Tensorflow is supported by a larger developer community which is more suitable for deployment on a production system; while PyTorch is more flexible in model development and ideal for research and prototyping.

2.9 Backbone CNN Models

According to study [49], VGG [50], InceptionV3 [51], ResNet-50 [52], XceptionNet [53], DenseNet [54], MobileNet [55], and Efficient-Net [56] have been widely used as backbone model when designing or developing a deepfake detector. VGG [50], which stands for Visual Geometry Group, is one of the most popular image recognition architectures that propose to replace large kernel filters with uniform 3x3 kernel filters to improve network decision functions and reduce the issue of network overfitting. The inception network [51] focuses on reducing computational power for deep networks by leveraging a dimensionality reduction with stacked 1×1 convolutions. ResNet-50 [52] is a residual network with a 50 convolutional network that exploits 1×1 convolutions as a 'bottleneck' for designing building blocks to reduce parameter numbers and matrix multiplication, thus speeding up the training process. XceptionNet [53] replaces the standard inception modules in InceptionV3 with depthwise separable convolutions to efficiently use the model

parameters for improving model performance. DenseNet [54] was developed to improve the model performance due to the gradient vanishing problem of deep neural networks. It allows each layer access to the features of all previous layers via the dense connection. MobileNet [55] is a low-latency CNN based on 3x3 kernel filters that implement depthwise separable convolutions to reduce computational costs. Similar to MobileNet, Efficient-Net [56] aims to reduce FLOPS and computational power while maintaining high performance, but it focuses on the use of the model scaling method. The idea is to scale the dimensions of the CNN models in a more structured and efficient manner to achieve a suboptimal performance. Specifically, the analysis [49] showed that Xception-Net and ResNet both take 17% while VGG takes 12% in the total frequency of model implementation. Table 2.1 outlines the total parameters of each CNN. The larger the number of parameters, the higher the computational power required for replicating and training the model, especially when working with huge image datasets. The developments of MobileNet and EfficientNet show that the research community has put effort into exploring lighter-weight yet higher-accuracy networks for future computer vision tasks.

Model	Total Parameter (Millions)
VGG [50]	138
InceptionV3 [51]	25
Resnet-50 [52]	23
Xception [53]	22
DenseNet [54]	20
MobileNet [55]	13
EffifientNet [56]	11

TABLE 2.1: Total Parameters of each backbone CNN

2.10 Training Constraint

Building a neural network model is an arduous task that may encounter numerous obstacles. The selection of network capacity, dataset size, and parameter configuration (such as batch size setting) requires careful consideration in relation to the available hardware resources, especially the computer Random-Access Memory (RAM). Although implementing a larger dataset and batch size or designing a complex neural network with multiple layers and neurons can improve the network performance, it can also increase computational costs and easily exhaust the computer RAM, leading to training failure.

Chapter 3

Overview of Deepfake

This chapter is mainly based on the following publications:

Jia Wen Seow, Mei Kuan Lim, Raphaël C.W. Phan, & Joseph K. Liu (2022). A comprehensive overview of Deepfake: Generation, detection, datasets, and opportunities. Neuro-computing, 513, 351-371.

DOI: 10.1016/j.neucom.2022.09.135

Jia Wen Seow, Mei Kuan Lim, Raphaël C.W. Phan, & Joseph K. Liu (2023). SparcoNet with Block-Switched Self-Supervised Learning: An Effective Framework for Deepfake Detection with Improved Adversarial Defense. Submitted to Information Sciences Journal. [Submitted]

3.1 Deepfake Generation

Deepfake generation refers to a new technique for tampering with media that surpasses traditional forgery methods by minimizing manipulation traces or fingerprints, which are often used for forgery detection (such as biometric or compression artifacts inconsistency) [57, 58]. This technique relies on a deep neural network that learns the segmentation map or latent representation to extract input characteristics and produce a new, hyperrealistic fake content based on the input data. Due to the minimal distinction between real and

fake data, detecting deepfakes is more difficult than detecting traditional manipulated media.

3.1.1 Deepfake Generation Techniques

There are three primary models for creating deepfakes: *i.* Autoregressive model [59], *ii.* Autoencoder [60], and *iii.* Generative Adversarial Network (GAN) [10].

The **Autoregressive model** is designed to model the conditional distribution of each pixel based on its previous pixels rather than relying on latent representation. While it can generate high-quality images, the evaluation process can be time-consuming due to its pixel-by-pixel predictions and sequential evaluation. Examples of this model include Pixel-RNN [61] and Pixel-CNN [62].

The **Autoencoder** is a type of artificial neural network used for unsupervised data representation learning. It consists of a coupled network formed by an encoder and decoder. The encoder converts the input into a hidden latent representation, which the decoder then uses to regenerate the data back to its original form. The goal is to produce output that is as close as possible to the input. Variational Autoencoder (VAE) plays a vital role in the generation of deepfakes. Figure 3.1 illustrates the basic workflow of an autoencoder. The purpose of an autoencoder is to train the network to recognize the essential features of the input while ignoring irrelevant noise. VAE differs from conventional autoencoders in that it assumes a particular probability distribution of latent variables, which enables it to better restore complex input information. However, VAE is more prone to generating blurry outputs, although it can easily produce new output after training by sampling the distribution.

The **GAN** is comprised of a pair of neural networks: a generator and a discriminator network. The generator network's objective is to produce a synthetic output that mimics the input's data distribution to deceive the discriminator. Conversely, the discriminator network's aim is to distinguish whether the output sample is real or fake. Both networks continually optimize through backpropagation until they reach an equilibrium state where fake data is indistinguishable from real data. GANs enable manipulation such as style transfer and image restoration that are challenging to achieve with traditional forgery generation methods [63].

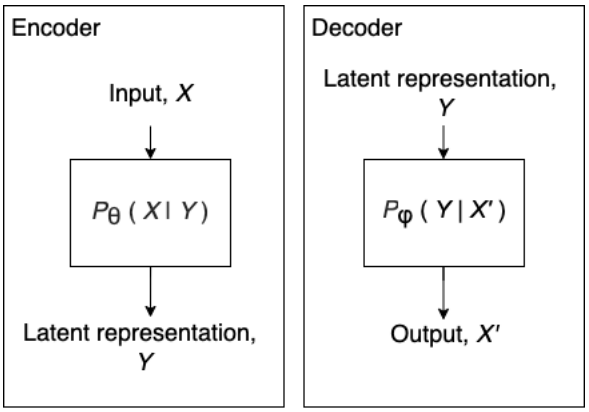


FIGURE 3.1: The encoder, pθ converts the input data, x into a latent representation, y, and the decoder pφ reconstruct the data back as output, x'.

Over the years, various GANs [12, 14, 15, 64–68] have been published to enhance software application performance. For example, ZAO [69] and FaceApp [70] demonstrate excellent performance in producing entertainment deepfake videos. The RCNN network [71] was developed for mobile cameras to improve camera resolution. Stacked-GAN [72] was used to address the common low-quality deepfake synthesis issue using super-resolution to preserve more facial details in image synthesis. However, GANs require high computational power and a vast dataset for training [15].

3.1.2 Deepfake Types

The deepfake type can be categorized into four major groups: entire face synthesis, reenactment (facial expression, body motion), facial attribute manipulation, and face-swapping. Figure 3.2 illustrates the different types of deepfakes.

3.1.2.1 Face Systhesis

Face synthesis is a technique that involves learning the latent representation of face data to generate a hyper-realistic synthetic persona. The synthetic persona does not exist in the real world as it is created without a target subject. While this technique benefits the gaming and modeling industries, it can also be used by attackers to fake someone's identity for illegal activities.

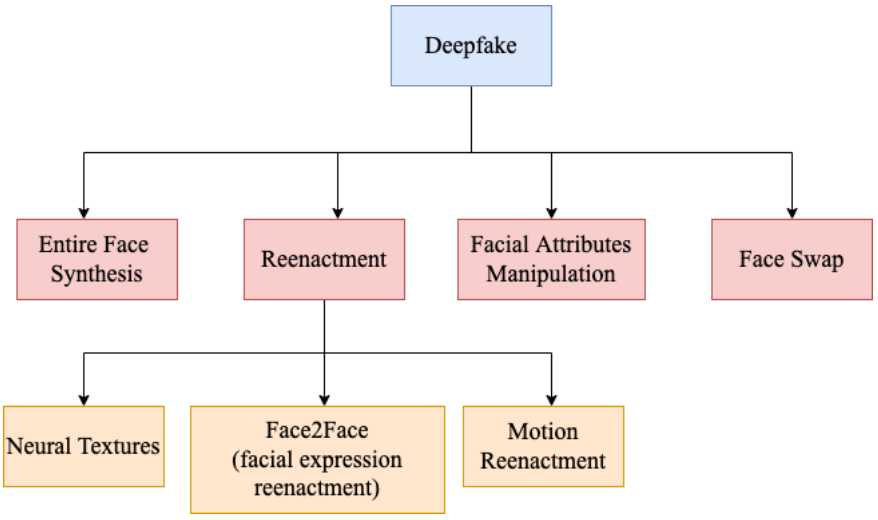


FIGURE 3.2: The four main types of deepfake.

In 2017, Radford *et al.* [73] proposed a more stable generative model architecture called DCGAN, which enhances the overall training stability. They implemented a deep convolutional concept without pooling and batch normalization to improve image synthesis performance based on an arithmetic vector. A year later, NVIDIA researchers [74] introduced another network architecture called ProGAN to further improve output quality and stability during network training. They progressively trained the input from low resolution and gradually improved fine details throughout the training process.

StyleGAN [16] inherits from ProGAN and introduces an adaptive instance normalization (AdaIN) to control the generator learning at each convolutional layer. The generator tends to synthesize a consistent style or pose based on the provided vector. The authors also introduced stochastic variation in controlling the placement of synthesized features such as hairs, stubble, freckles, or skin pores. However, they found that the instance normalization of StyleGAN produced significant water droplet-like artifacts in the synthetic image, making it easily exposed to detection. Therefore, the same research team redesigned a new normalization approach and published it as StyleGAN version 2 [75] in the same year. They successfully improved image quality and eliminated the artifacts seamlessly. ProGAN and StyleGAN are widely used to produce synthetic face databases [76, 77].

3.1.2.2 Reenactment

Unlike face synthesis, reenactment involves transferring facial expressions or body motions between people. This technique was popular even before the appearance of deepfake, and traditional approaches utilized computer graphics to achieve reenactment results. For example, Blanz *et al.* reenacted the input image with a 3D morphable model extracted from a database of 3D scans [78]. Thies *et al.* promoted real-time facial expression transfer using a commodity RGB-D sensor to capture facial performance. They altered the parameters of the target to fit the source expression with a parametric model and employed a mouth retrieval synthesis to produce a high-quality outcome [79, 80]. In this section, we will outline several reenactment techniques, such as facial expression transfer with neural textures, typical facial expression reenactment (Face2Face), and body motion reenactment (Puppet Master).

Neural Textures utilizes feature maps, texture maps, or neural textures within the parametric vectors to generate photorealistic reenactment results. This technique was first proposed in [13], where the authors developed an end-to-end deferred neural rendering network using a convolutional encoder-decoder that blends traditional computer graphics knowledge with adaptable neural textures. The authors created a UV map by sampling the target's neural textures and aligning them with the source's expression. The UV map was then inputted into the neural renderer, along with the background image, to synthesize the reenactment result. However, the output quality can be influenced by the geometry proxy. To achieve a similar concept, Fried *et al.* [81] focused on lip reenactment. They produced an altered output by modifying, eliminating, or adding dialogue to a talking-head video. The authors used a parametric face model to control the expression and pose variations across different frames based on the transcript and video sequence.

Face2Face is a widely used method for reenacting facial expressions from a source to a target, which attackers can exploit to manipulate the target's appearance and expressions. One notable instance of this technique was the creation of a deepfake video featuring former President Obama [17, 18]. Some studies have used Barack Obama's dataset for training their models. For example, in [82], the authors generated new videos based on Obama's voice and video stocks. They extracted audio features using

Mel-frequency Cepstral Coefficients (MFCC) from the source video, converted the target's mouth shape into vectors, and applied Principal Component Analysis (PCA) to represent the mouth shape over frames. They then used an LSTM network to map the MFCC audio coefficients to PCA mouth shape coefficients. Kumar *et al.* [83] introduced a fully trainable reenactment network for lip synthesis based on text input using a similar dataset. Their network consisted of three modules: *i.* A text-to-speech network, CharWav, *ii.* An LSTM network for converting audio to mouth keypoints, and *iii.* A UNet-based Pix2Pix network to synthesize the target video based on the mask and mouth keypoints. In another work, Song *et al.* [84] also used MFCC to extract audio features but proposed a conditional recurrent network to ensure temporal coherence and improved lip movement in an adversarial manner during training.

CycleGAN has been widely used in numerous reenactment studies. Originally introduced in [66], CycleGAN enables the translation of image content from source domain A to target domain B without requiring paired data. The method involves training two GANs with cycle-consistency loss to learn the domain conversion between source and target in a return way, thus ensuring that the synthetic output obtains the characteristics of the target domain while retaining its original content. However, this method may have poor performance when data have significant geometric or distribution differences. StarGAN [14], another CycleGAN-based translation network, focuses on multi-domain translation tasks with a single model, thereby simplifying the effort required to train the transfer of multiple expressions with support for mask vectors for different facial expression labels. RecycleGAN [85], introduced in 2018 in cooperation with Facebook Reality Lab, is a data-driven translation network designed for video retargeting. The model implements CycleGAN with a recycle formulation, utilizing loss functions such as recurrent loss, cycle consistency loss, recycle loss, and adversarial loss to optimize spatio-temporal constraints and obtain better local minima during style transformation. RecycleGAN successfully addresses the mode collapse issue of conditional-GAN and produces hyper-realistic output in facial reenactment tasks. To improve the mapping boundary latent space from source to target in facial reenactment, Wu *et al.* [86] leveraged CycleGAN for boundary transformation and implemented Pix2Pix encoder-decoder for reconstructing synthetic output.

In 2020, a generic face animator called Interpretable and Controllable face reenactment network (ICface) was published by [87]. The network comprises two major stages: *i.*

extracting facial attributes from the driving image, such as interpretable head pose angles and Action Unit (AU) values, and *ii.* integrating the extracted attributes with the source image using conditional-GAN. Before training, the source image is first converted into a neutral image using a neutralized generator. The results show a notable performance in pose transformation and face reenactment, with less semantic distortion than the baselines. However, there is still room for improvement in mitigating noticeable artifacts. In the same year, [88] proposed an Ordinal Ranking Adversarial Networks based on the concepts of CycleGAN and StarGAN. The generator works with a multi-scale discriminator and a one-hot label to denote the ranking of the input's age and expression intensity, ensuring that the synthesis is correctly conducted according to specific age groups or expression intensity. They have improved the precision and performance of condition-based synthesis.

Several studies have focused on identity-invariant pose and expression reenactment. One example is FaceID-GAN [89], which used a GAN with an identity classifier to preserve identity features during adversarial training. They used 3DMM to convert input images into shape, pose, and expression parameters for synthesizing the reenacted face. Another approach is DR-GAN [90], which employed an encoder-decoder structure to learn disentangled representations for face translation and transformation.

However, most synthesis neural networks require large datasets to learn the latent representation from different perspectives. To address this, Zakharov *et al.* [91] proposed a few-shot learning model using meta-learning to map face landmarks to embedding vectors. The embedder, generator, and discriminator were trained in a K-shot learning manner, and the output was fed into a Pix2Pix generator with landmarks from a different frame to produce the synthetic target output. FaceSwapNet [92] is another well-known network that addresses scalability limitations and supports many-to-many face reenactment. It uses a landmark swapper module and a landmark-guided generator to compute swapped landmarks vectors and synthesize the reenacted face, respectively.

Motion reenactment, also known as Puppet Master, is capable of achieving a high level of photorealistic motion transfer, where the body motion or position is transferred from the source to the target without altering the original appearance. However, a common issue with this technique is pixel-to-pixel misalignment due to the differences in sources and targets. To address this challenge, the authors of [93] proposed a method

that utilizes deformable skip-connections with the nearest neighbor loss. The approach involves decomposing the source's global information into a local affine transformation set based on the target pose, deforming the source's feature map, and applying a common skip-connection to transfer the transformed tensor, which is then fused with other corresponding tensors in the decoder to produce the synthetic output.

Unlike the feature mapping-based image reconstruction in [93], Neverova *et al.* [94] proposed a warping module that performs texture mapping from a source to a target, resulting in high-quality texture restoration for various viewpoints and body movements. They used a conditional generative model for target pose prediction and employed texture wrapping with a Spatial Transformer Network (STN) based on the UV coordinates of each surface area. In their work [95], the authors proposed a modular neural network that translates changes in pose into image space using four major modules. These modules include segmentation of the source image to separate the background and foreground, spatial transformation of segmented body parts, foreground synthesis to produce a hyperrealistic target appearance, and background synthesis using the foreground mask from the previous module to complete the body movement reenactment. The goal of this approach is to generate a new video by using a generative neural network to depict unseen poses. To further improve synthesization, Tulyakov *et al.* [96] decomposed the video into content and motion subspaces. They utilized a Gaussian distribution to sample the content subspace and produce motion embedding, then mapped the source content with the target motion vector to synthesize a new video. Their approach, named MoCoGAN (Motion and Content Decomposed GAN), employed a GRU network to create a vector set that formed the motion representation and an image generator to generate videos. Two discriminators were used to ensure the output quality, where the image discriminator ensured the photorealism of each frame, and the video discriminator guaranteed the temporal coherence between the frames. Aberman *et al.* [97] proposed a two-branch network to also deal with unseen poses. The first branch focused on learning pose-to-frame mapping, while the second branch aimed to achieve temporal coherence to convert the unseen poses into sequences that match the source video.

In their work [98], Kim *et al.* focused on reenacting head pose, eye gaze, and facial expression using a novel monocular face reconstruction technique that obtains a low-dimensional parametric representation of the source and target. They modified the posture, eyes, and expression parameters from source to target while preserving the

source's identity and background illumination, and then rendered the conditioning input images based on the modified parameters. The rendering-to-translation network used a space-time encoder and an image decoder to convert the conditioning input images into a synthetic video portrait. A similar model for character-to-image translation was presented in [99], where the authors reconstructed the target from a static image to a 3D character model and trained it with motion data to produce video-realistic output. However, they identified several limitations that significantly degraded the network's performance, such as non-linearity of articulated motions, performance discontinuities due to self-occlusion, quality degradation due to imperfect monocular tracking, and an inability to capture challenging poses.

NVIDIA introduced a widely used video-to-video translation network called Vid2Vid-GAN [100], which begins the video translation process by matching the source's conditional distribution to the target and synthesizing the target background and foreground using the source's segmentation mask. However, the model's performance can be unreliable and inconsistent due to insufficient semantic labels in training. To address this issue, the authors proposed the few-shot Vid2Vid framework [101], which uses a network weight generation module with an attention mechanism. The proposed network requires only several target images for the synthesis of unseen data, but its training heavily relies on the semantic estimation input, which limits its performance.

The paper [11, 102, 103] utilized a Pix2Pix network for movement transfer, which involved similar preprocessing tasks such as extracting the source's pose or motion into keypoints, landmarks, or segmentation masks, and then mapping it with the source foreground before feeding into Pix2Pix-GAN for synthesis. However, the authors implemented different enhancement approaches to support their solutions. In [11], FaceGAN was proposed to improve the realism of the face region, while Liu *et al.* [102] combined upper body keypoints (UBKP), facial action units, and pose (FAUP) to increase facial detail during training. Zhou *et al.* [103] encoded position, orientation, and body parts as a Gaussian smoothed heat map to refine foreground synthesis and alleviate incoherent body artifacts.

In [104], the authors introduced an identity invariant siamese generative adversarial network (PS-GAN) to resynthesize the input based on a pose-guided image. The network

consisted of two identical branches: a generative network with pose-attentional transfer blocks (PATBs) that encoded the input and target pose-guided image into feature representation and a pair-conditional discriminator that differentiated the generated image from the real one. This study made a significant contribution to the Person Re-Identification (ReID) task by enabling the identification of the generated image's identity.

3.1.2.3 Facial Attributes Manipulation

Facial attribute manipulation involves modifying specific facial characteristics such as eye color, hairstyles, wrinkles, skin color, gender, and age, which can alter a person's appearance based on preset conditions. One of the most popular domain-to-domain translation networks for this technique is StarGAN [14]. Unlike other networks such as [66, 105–108], which focus on style translation between two domains, StarGAN employs a mask vector methodology to facilitate multi-domain training. Another network proposed by Xiao *et al.* [109], called ELEGANT, also uses a multi-attribute CycleGAN-based translation network and emphasizes model training with adversarial loss, domain classification loss, and reconstruction loss. However, the tampered outputs from ELEGANT often contain unwanted artifacts, making it less desirable than StarGAN, despite its notable style transfer results.

AttGAN [110] differs from [14, 109] as it focuses on producing high-quality facial attribute outputs by employing attribute classification constraints instead of attribute-independent constraints in latent representation to ensure the preservation of attribute-excluding details during modification. Although the authors achieved favorable results, it is still not feasible for large area appearance modification. In contrast, Liu *et al.* introduced STGAN [111], which overcomes the blurry output issue by embedding a selective transfer unit with an encoder-decoder network. The selective transfer unit algorithm is constructed based on the Gated Recurrent Units (GRU) mathematical model, and the result shows that the synthetic image quality of STGAN is better than StarGAN and AttGAN. In [112], the authors propose a URCA-GAN network based on Upsample Residual Channel-wise Attention Module (URCAM) and StarGAN to manipulate the specific content of the input's foreground that is different from the target image. The

URCAM determines the attention map and regulates the most distinctive features without affecting the spatial dimension of the transformation. Their approach results in a higher-quality yet lesser-artifact output compared to [14, 66].

Li *et al.* presented BeautyGAN [68] to transfer makeup styles from one instance to another while preserving the face identity. They improved the translation consistency of makeup styles on different faces by translating the makeup style of a reference input to the target output in intra- and inter-domain perspectives. The authors utilized different loss functions such as makeup loss, perceptual loss, and cycle consistency loss to improve output realism and maintain face identity. Similarly, [113–115] used different loss functions to control attribute manipulation from various perspectives, including age, identity, expression, and facial attributes. Despite the various approaches, BeautyGAN achieved the highest voting rate of 61.84

In [67], Jo *et al.* proposed SC-FEGAN, a GAN-based image editing system that can edit images based on free-form masks, sketches, or colors. The algorithm can translate sketch input into a hyperrealistic texture form and fuse it with the original image to produce a high-quality, artifact-free synthetic image. The SC-FEGAN algorithm relies heavily on input processing, such as face segmentation and free-form input feature extraction. The authors utilized histogram equalization [116] and holistically-nested edge detection [117] to process sketch data and feed the output for synthesis training with the input image. This study provides insight into future opportunities to use simple drawing skills to achieve a sophisticated image editing or restoration process. Afifi *et al.* proposed HistoGAN [118], a color histogram-based generative model to naturally alter skin tone. They used StyleGAN as their backbone model and applied a color histogram instead of a fine-style vector for the last two blocks of StyleGAN to control the color of the generated image.

3.1.2.4 Face Swapping

Face-swapping is a technique that involves swapping faces from one image to another while maintaining the original facial expression. In 2017, Korshunova *et al.* introduced a rapid face-swapping methodology [119] that transforms person A's identity to person B while keeping the head position, facial expression, and lighting conditions unchanged. In contrast to common style transfer, the style in this case refers to the person's identity,

while the rest is considered the content. The authors employed a modified multiscale convolutional neural network (CNN) with content loss, style loss, and light loss functions to transform A's content into B. Using an affine transformation with 68 facial keypoints, they aligned the output face and stitched the background with a segmentation mask. As the neural network is trained to learn A's content, it can swap with multiple target Bs, resulting in a one-to-many face swap algorithm. However, training the network requires a large dataset of single-person images for fine-tuning, which may not be practical in common applications.

The authors in [120] proposed a face replacement method to obfuscate identity by utilizing a parametric Model-based Face Autoencoder (MoFA) and a GAN for synthesis and blending with the background. In contrast, Nirkin *et al.* [12] derived a recurrent neural network (RNN) approach consisting of three components to support face-swapping and facial reenactment. The *Gr* component obtained pose and expression from the target's facial landmarks and generated the source reenacted face to form a segmentation mask, while *Gs* and *Gc* were used for segmentation and inpainting, respectively. The hair and face segmentation mask of the target image were computed by *Gs*, while the missing areas or occluded parts on the output of *Gs* were inpainted by *Gc* to obtain a detailed, completed face-swapped result. To ensure facial temporal coherence and support face view interpolation, Delaunay Triangulation and barycentric coordinates were implemented. However, the output resolution tended to degrade for content with different angles.

Microsoft collaborated with researchers from Peking University to publish a two-stage framework named FaceShifter[121] that can handle occlusion during face-swapping for high-fidelity output production. They performed face shifting using an Adaptive Embedding Integration Network (AEINet) and a Heuristic Error Acknowledging Network (HEARNet). AEINet is an adaptive attentional denormalization generator that denormalizes local feature integration at different levels. Meanwhile, HEARNet leveraged the heuristic error between the input and reconstructed image to identify the occlusion position and further refined it in a self-supervised manner. This framework shows superior performance in preserving the identity and occlusive face accessories during face-swapping.

3.1.3 Available Deepfake Generation Tool

Many developers and researchers are keen to share their studies as open-source tools or applications that are user-friendly, which contributes to the proliferation of deepfake content on social media platforms. Table 3.1 presents a list of publicly available deepfake generation tools and applications, along with their features. These include Face App [70] for manipulating facial attributes, and DFaker, ZAO, Deep Face Lab, Face Swap, Deepfakes web β, Machine Tube, and Reface apps [69, 122–127] for swapping faces. The Avatarify tool [128] can transfer facial expressions, while Impersonator++ and Jiggy tools [129, 130] can transfer movement.

3.1.4 Disscusion

In the literature, face reenactment is considered the pioneer type of deepfake. However, with the exponential growth of deep learning, deepfake technology has expanded beyond just face reenactment to include body reenactment. When producing high-quality deepfakes, most researchers prefer to use a stable GAN structure network, despite its longer training time. This is because GAN can generate sharper outputs compared to Autoencoder and Autoregressive models.

Given that deepfake technology has the potential to cause psychological and physical harm to individuals and organizations, it is essential to be aware of the various types of deepfake to distinguish between truth and fake media. A comprehensive understanding of different deepfake behaviors is also necessary to create a reliable deepfake detection model. Tables 3.2 and 3.3 provide a summary of key information from the discussed studies, which can aid in better understanding deepfake types. The tables use abbreviations such as DF (Deepfake), F2F (Face2Face), FS (Face Swapping), EFS (Entire Face Synthesis), FA (Facial Attribute Manipulation), MR (Motion Reenactment), NT (Neural Texture), Au/ED (Autoencoder/Encoder-Decoder), ArM (Autoregressive Model), L (Loss Functions), SC/Att (Statistical Characteristic/Attention Mechanism), and TPF (Tertiary Preprocessing Framework) to label and distinguish between the various deepfake types.

Ref	Deepfake Type	Tools	Feature
[70]	Facial Attributes Manipulation / Face2Face	FaceApp	Support modification of facial expression and face attributes
[122]	Face Swap	DFaker	Support training of face swap model
[69]		ZAO	Allows face swap with celebrities from movie or TV show
[123]		DeepFaceLab	Allows face swap in videos
[124]		FaceSwap	Support face swap between peoples
[125]		Deepfakes web β	Support training of face swap model
[126]		MachineTube	Support face swap in image or video
[127]		Reface	Allows face swap with celebrities or movie character
[128]	Face2Face	Avatarify	Allows to transfer facial expression from one to the target avatar
[129]	Puppet Master	Impersonator++	Support motion transfer using image synthesis
[130]		Jiggy	Allows to animate the person in the static image to dance motion

TABLE 3.1: Available deepfake generation tools or applications

3.2 Deepfake Detection

There are two approaches to detect deepfakes: (i) Conventional Methods that rely on handcrafted features, and (ii) Deep Learning Approaches that emphasize learned features. Figure 3.3 provides an overview of these approaches and their sub-groups. Like detecting traditional forgeries such as copy-move, splicing, and inpainting, the deepfake detection pipeline also involves input preprocessing, feature extraction, and classification. However, the deep learning approaches perform feature learning between the feature extraction and classification stages.

No.	Ref.	Year	Pub.	DF Type	Technique								Output
					GAN	Au/ED	ArM	CNN	RNN	L	SC/ Att	TPF	
1	[73]	2015	ICLR	F2F	✓								Image
2	[82]	2017	ACM Trans. Graph	F2F					✓		✓	✓	Video
3	[83]	2017	NeurIPS	F2F			✓		✓			✓	Video
4	[66]	2017	ICCV	F2F	✓					✓			Image
5	[119]	2017	ICCV	FS				✓			✓		Image
6	[131]	2018	ACM SIG-GRAPH	FS	✓	✓							Image
7	[132]	2018	ACCV	FS		✓							Image
8	[120]	2018	ECCV	FS	✓	✓							Image
9	[85]	2018	ECCV	F2F	✓		✓			✓			Image / Video
10	[109]	2018	ECCV	FAM	✓								Image
11	[14]	2018	CVPR	F2F/ FAM	✓					✓			Image
12	[68]	2018	ACM-MM	FAM	✓					✓			Image
13	[15]	2018	ICLR	EFS	✓					✓			Image
14	[89]	2018	CVPR	F2F	✓			✓				✓	Image
15	[90]	2018	TPAMI	F2F	✓	✓							Image
16	[93]	2018	CVPR	MR	✓							✓	Image
17	[95]	2018	CVPR	MR	✓			✓			✓		Image
18	[94]	2018	ECCV	MR	✓						✓		Image
19	[84]	2018	IJCAI	F2F	✓				✓			✓	Video
20	[86]	2018	ECCV	F2F	✓		✓			✓			Video
21	[96]	2018	CVPR	MR	✓				✓		✓		Video
22	[98]	2018	ACM Trans. Graph	MR	✓							✓	Video
23	[100]	2018	NeurIPS	MR/ EFS	✓					✓			Video
24	[111]	2019	CVPR	FAM	✓	✓			✓				Image / Video
25	[110]	2019	IEEE TIP	FAM	✓	✓							Image / Video

TABLE 3.2: Summary of deepfake generation part 1

No.	Ref.	Year	Pub.	DF Type	Technique								Output
					GAN	Au/ED	ArM	CNN	RNN	L	SC/ Att	TPF	
26	[11]	2019	ICCV	MR	✓		✓					✓	Video
27	[67]	2019	ICCV	FAM	✓						✓		Image
28	[12]	2019	ICCV	FS/ F2F			✓			✓	✓		Image / Video
29	[16]	2019	CVPR	EFS	✓						✓		Image
30	[75]	2019	CVPR	EFS	✓								Image
31	[13]	2019	ACM Trans. Graph	NT		✓					✓	✓	Image / Video
32	[102]	2019	ISMAR-Adjunct	MR/ F2F			✓				✓		Video
33	[81]	2019	ACM Trans. Graph	NT		✓			✓			✓	Video
34	[99]	2019	ACM Trans. Graph	MR	✓							✓	Video
35	[97]	2019	Comput Graph	MR	✓						✓		Video
36	[103]	2019	ICCV	MR			✓			✓		✓	Video
37	[91]	2019	ICCV	F2F			✓				✓		Video
38	[101]	2019	NeurIPS	MR/ F2F	✓						✓		Video
39	[92]	2019	CVPR	F2F		✓				✓		✓	Image / Video
40	[121]	2019	CVPR	FS		✓				✓	✓		Image / Video
41	[115]	2019	Neuro-computing	FAM	✓					✓			Image
42	[87]	2020	WACV	F2F	✓			✓					Image
43	[133]	2020	WACV	F2F	✓								Image / Video
44	[88]	2020	IEEE TIFS	F2F	✓								Image
45	[104]	2020	Neuro-computing	MR	✓					✓			Image
46	[113]	2020	Neuro-computing	MR	✓					✓			Image
47	[114]	2020	Neuro-computing	MR	✓					✓			Image
48	[112]	2021	Neuro-computing	FAM	✓						✓		Image
49	[134]	2021	IEEE TIFS	F2F		✓		✓		✓			Image
50	[118]	2021	CVPR	FAM	✓					✓			Image

TABLE 3.3: Summary of deepfake generation part 2

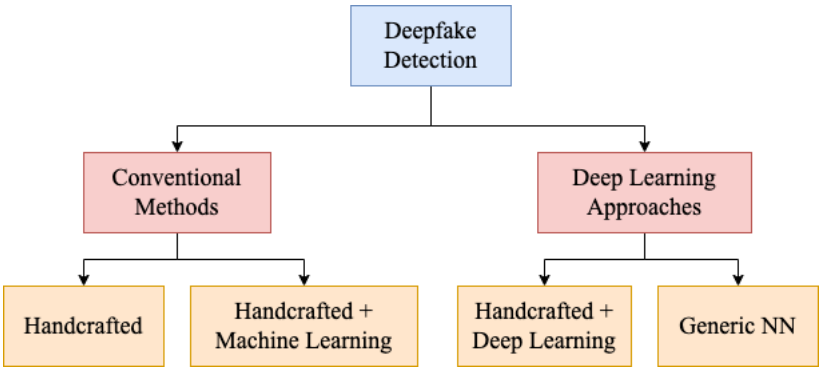


FIGURE 3.3: Deepfake detection

To illustrate the differences between the conventional and deep learning approaches in the detection process, Figure 3.4 presents the standard deepfake detection pipeline. This section reviews the studies on both conventional and deep learning-based detection methodologies.

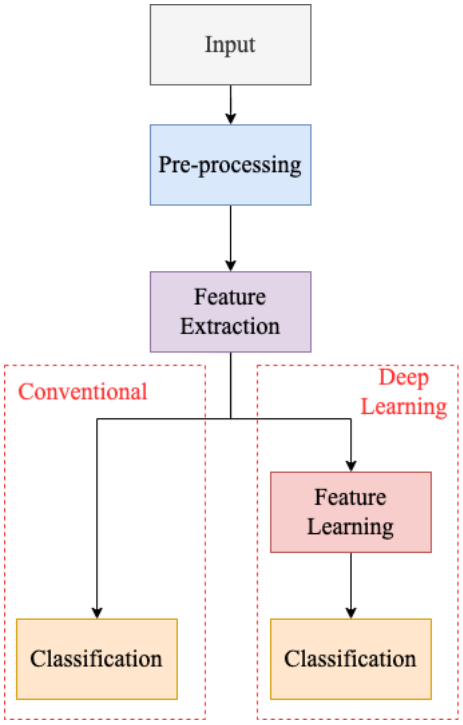


FIGURE 3.4: Common detection pipeline

3.2.1 Traditional Deepfake Detection Methods

The traditional deepfake detection methods require a series of sophisticated algorithms to extract meaningful information or features from the raw data before entering the classification stage. This is because the raw data cannot be directly input into the

machine learning algorithms for classification. The extracted features are known as handcrafted features, which could be subtle traces, pixel anomalies, edge boundary discrepancies, or abnormal artifacts presented in the counterfeit input data. Figure 3.5 shows the types of handcrafted features. These processes for feature extraction can be tedious and time-consuming.

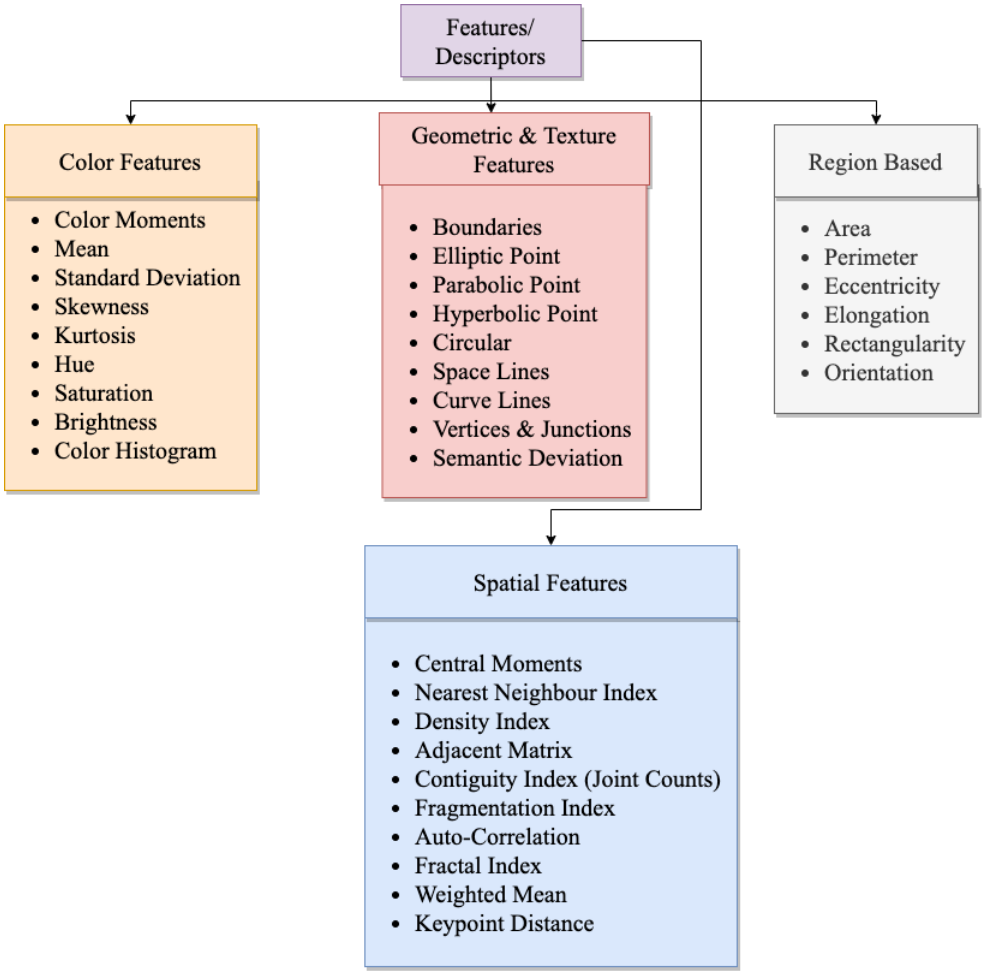


FIGURE 3.5: Example of handcrafted features

In Zhang *et al.* [135], the authors utilized a Speeded Up Robust Features (SURF) algorithm along with the Bag of Words model (BoW) for detecting face swaps. The SURF algorithm is a faster version of the Scale Invariant Feature Transform (SIFT) algorithm, which was used for localizing and detecting features. The authors employed two methods to select SURF keypoints: Grid division and Interest point detection. The selected keypoints' descriptors were then extracted, passed through a clustering system to generate features using the BoW model, and classified using Support Vector Machine (SVM). In contrast, Agarwal *et al.* [136] proposed feature extraction based on pixel anomalies. They applied weighted local magnitude patterns by assigning weight inversely

proportional to the absolute differences between the center and neighboring pixels. A histogram feature vector was constructed based on the output values and classified using SVM. Based on the extracted features, the authors discovered that tampered images retained high-frequency information but lost low-frequency details, making them susceptible to detection. For instance, although the central face appears well-blended, facial keypoints such as the eyes, nose, and mouth are ambiguous.

The Photo Response Non-Uniformity (PRNU) technique utilizes non-uniform noise patterns for forgery detection. In a study by Koopman *et al.* [137], they presented a purely handcrafted PRNU-based deepfake detection approach by comparing the final Welch's t-test evaluation scores for both original and deepfake videos. To do this, they preprocessed the face region-related video frames, evenly split them over eight groups, and computed the normalized correlation scores based on each group's noise patterns. However, this method is not entirely reliable as most deepfakes in real-life scenarios do not have a comparison source. In [138], the authors proposed an unsupervised detection methodology based on classical frequency domain analysis that addresses this issue. This approach requires no training sample amount and uses a Discrete Fourier Transform (DFT) algorithm to capture the deepfake image frequency by decomposing the discrete signals. They then convert the frequency data into a 1D-representation feature vector using an Azimuthal average for classification.

Meanwhile, Marra *et al.* [139] analyzed the possibility of identifying different GAN sources based on the fingerprints left in their deepfake output. This study suggests that GAN anomalies provide clues for deepfake detection. In this direction, papers [140, 141] introduced several methods to exploit the non-uniform distribution drawback of GANs during deepfake generation for deepfake detection. Both studies extracted color components from the deepfake image using different extraction approaches and converted them into feature vectors for classification using SVM. However, their performance could be significantly degraded with the improvement of GANs in data distribution.

Biometric artifact is another rising trend for deepfake detection. Yang *et al.* [142] proposed exploiting the mismatched facial landmarks of deepfake images as a clue for detection. They used the head orientation vector with a face detector using DLIB [143] and OpenFace2 [144]. In [145], the authors captured unusual artifacts, such as anomalies in the reflections, eyes, or teeth details, for detection. They used facial geometry to train

the SVM classifier with four feature vector sets: Eye, Teeth, 16-dimensional eye, teeth, and Full face crop feature vectors. These approaches exploit the drawback of deepfakes in output realism.

With the advancement of deepfake generation technologies, handcrafted feature extraction has become increasingly difficult [146]. As a consequence, the focus of deepfake detection research has shifted towards deep learning methods, which aim to provide more flexible and dependable detection through dynamic feature learning.

3.2.2 Deep learning-based Deepfake Detection Methods

Learned features have demonstrated their success in solving complex issues in various domains, including advanced computer vision tasks, machine translation, face recognition, object detection, and localization [147]. Deep learning approaches utilize a black box CNN feature extraction process that automatically learns and derives features from the training data using a deep neural network [57, 148]. The deep learning-based methodology comprises four main processes: *a.* Data Preprocessing, *b.* Feature Extraction, *c.* Feature Learning, and *d.* Classification. It can be divided into two primary groups: Handcrafted feature-based feature extraction and Generic NN.

i. Handcrafted feature-based feature extraction

In this approach, the authors utilized specific handcrafted features or post-processing of the feature extracted from NN as the input for further model training.

Biometric artifact In [149], the authors used a long recurrent convolutional network (LRCN) with eye sequences to detect deepfakes. Their model, based on VGG and LSTM architecture, leveraged the time-series nature of eye-blinking activity to capture the temporal features required for deepfake discrimination. However, this methodology might not be effective in people with mental illness or nerve conduction issues as these conditions can easily influence the eye blinking frequency.

Another study [150] focused on evaluating the similarities between source and target eyebrows with a cosine distance metric, assuming that high-resolution deepfakes are easier to detect through biometric comparison pipeline such as eyebrow alterations. However, this approach is limited to well-known subjects like politicians and celebrities

because it heavily relies on identity matching between the source and target, which requires a large number of training samples.

Ciftci *et al.* [151] proposed FaceCatcher, a method that emphasizes the detection of biological signals or Photoplethysmography (PPG). The authors used PPG to detect subtle changes resulting from skin color due to blood pumping or peripheral circulation through the face. The variation in the PPG signal provided valuable information for deepfake detection. The authors first preprocessed the PPG signals into a feature set using conventional signal processing methods, such as log scale, Butterworth filter, and power spectral density. Then, they applied a CNN classifier to handle the classification of the complicated feature space. By encoding the PPG feature maps with binned power spectral densities, the authors achieved an outstanding detection accuracy of 96%, but its performance might be compromised if the data is biased.

Yang *et al.* [152] proposed using lip sequence to support deepfake discrimination based on the client's talking habit. They preprocessed the raw input data by implementing a random password strategy and Dlib detector to extract the lip region, then further converted them into a lip sequence using Connectionist Temporal Classification (CTC). Next, they utilized a Dynamic Talking Habit-based Speaker Authentication network (SA-DTH-Net) to evaluate whether the extracted lip sequence conforms to the client's talking style. However, this method is not effective for broad deepfake detection as most deepfake videos do not obtain the original actor's lip sequence for talking habit evaluation.

In [153], Agarwal *et al.* hypothesized that the deepfake subject's mouth shape (Visemes) dynamics are sometimes inconsistent with the related spoken phoneme. They extracted the phoneme with Google's Speech-to-Text API, then manually aligned and synchronized the transcripts to the audio using P2FA [154]. They conducted experiments with different methods for visemes (classify if the mouth is open or closed) measurement and realized that the CNN-based approach achieved higher accuracy than the handcrafted-based algorithms. However, this approach might be time-consuming as it requires many manual operations in handling phonemes and visemes alignment.

Haliassos *et al.* [155] proposed another mouth feature-based approach. They trained the preprocessed grayscale lip-cropped frames with two pre-trained lipreading networks: a Resnet-18 model and a multi-scale temporal convolutional network (MS-TCN). The idea

was to finetune the detection model with the high-level irregularities in mouth movement. They achieved a good performance in cross-dataset evaluation with an average of 87.7% accuracy across five datasets. However, their model is susceptible to mouth features and failed to detect fake videos with mouth occlusion.

Spatio-Temporal Feature In [156], the authors proposed using optical flow, a vector field formulated on two consecutive frames *f(t)* and *f(t + 1)*, to detect unusual motion artifacts in deepfake videos. They extracted optical flow using PWC-Net and fed it into a semi-trainable network with VGG-16 and ResNet-50 as the network backbone.

On the other hand, [157] developed a motion-magnified spatial-temporal representation (MMSTR) with a dual-spatial-temporal attentional network (Dual-ST AttenNet) to capture PPG variations in both spatial and temporal aspects. They preprocessed the faces to form the MMSTR maps and followed three steps to gather the required features: *i.* Produce an adaptive spatial attention output using the MMSTR map and a spatial attention network, *ii.* Generate block-level temporal attention using LSTM, and *iii.* Feed the motion-magnified face video to the pre-trained Meso-4 network to output the frame-level temporal attention. Finally, they employed a ResNet-18 for classification by taking the features and MMSTR map as the classifier input. However, similar to [151], PPG signals could easily be affected by other texture factors, such as skin color, sunburn, or sensitive skin.

In order to capture temporal inconsistencies, such as sudden changes of brightness and facial artifacts in deepfake video, Tariq *et al.* [158] applied a simple CLRNet (Convolutional LSTM Residual Network). They used few-shot transfer learning to generalize the network by training it with several scenarios, including *i.* Single-source to Single-target, *ii.* Multi-source to Single-target, and *iii.* Single-source to Multi-target.

Pixel & Statistical Feature In [159], the authors used chrominance components for detection. They transformed the image from RGB space to a YCrCb space, extracted its edge information using a Scharr operator, and converted it into a gray level co-occurrence matrix (GLCM) for scaling. They utilized a depthwise separable convolution deep neural network for feature extraction and classification, achieving a higher average F1 score of 0.9865 compared to other methods.

Khodabakhsh *et al.* [160] proposed using a ResNet-based PixelCNN++ with a universal background model (UBM). The concept was to apply a conditional probability matrix on the log-likelihood of each pixel intensity to enhance feature extraction.

In [161], the authors introduced a self-supervised decoupling network (SDNN) for authenticity and compression feature learning. It exploited the compression ratio of given inputs as the self-supervised signals. The idea was to normalize the model with different compression rates so that the authenticity classifier could achieve better classification results without being affected by input compression. However, as the range of compression rates can be adjusted, the model's performance with an unseen compression rate might still be an issue.

Chen *et al.* [162] proposed a light-weight principal component analysis (PCA) based detection method, DefakeHop. They extracted features from different face regions using PixelHop++ and applied subspace approximation with adjusted bias (Saab) to reduce the spatial dimension of each patch. The output was then fed to an extreme gradient boosting (XGBoost) classifier for further classification.

In [163], the authors introduced a frequency-aware discriminative feature learning framework (FDFL) to solve the ambiguous feature discrimination of softmax loss and the low efficiency of handcrafted features for forgery detection. They presented a single-center loss (SCL) to bring the neutral face features to the center and push away the manipulated features. The authors found that a combined loss of SCL with softmax loss provided better results when working with the FDFL framework. However, the model has poor generalization with unseen datasets.

Luo *et al.* [164] noted that most CNN-detectors fail to generalize across different datasets due to overfitting in method-specific color texture. They found that image noise could efficiently remove color texture and expose forgery traces. Hence, they introduced a method using an Xception-based detector with SRM [165] high-frequency noise features. The entire model consists of three functional modules: *i.* A multi-scale high-frequency feature extraction module, *ii.* A residual guided spatial attention module, and *iii.* A dual cross-modality attention module. They adopted the suggested modules to extract more meaningful features and capture the correlation and interaction between the complementary modalities. The result shows that the model outperforms competing approaches by more than 15

In [166], Liu *et al.* used the Discrete Fourier Transform (DCT) to capture the phase spectrum for deepfake detection. They hypothesized that the phase spectrum is sensitive to up-sampling, which is an operation that is normally applied in deepfake generation, and assumed that the local textual information has more impact than high-level semantic information for forgery detection. However, it might be vulnerable to generation methods that do not use up-sampling.

ii. Generic NN

Generic NN is a detection methodology that comprises one or multiple neural networks conducting feature extraction and classification tasks. In contrast to other deep learning-based approaches that are integrated with handcrafted-feature, Generic NN only relies on learned-feature.

Face Recognition & Artifacts Discrepancies In their study on deepfake detection, Wang *et al.* [167] proposed a deep face recognition system called FakeSpotter, which employed a shallow layer-wise neural network architecture and introduced a new neuron coverage criterion called mean neuron coverage (MNC) to monitor the neuron behavior for synthetic face detection. By using MNC to specify the neuron activation, FakeSpotter achieved over 80% accuracy in comparison to other detectors. Yuezun *et al.* [168] focused on face artifacts discrepancies, training several neural networks such as VGG16, ResNet50, ResNet101, and ResNet152 to learn the discriminative features between deepfake face areas and their neighboring regions, exploiting the artifacts introduced by affine face warping during deepfake generation. Similarly, Nirkin *et al.* [12] presented FSGAN to improve deepfake classifier performance by utilizing multiple face identification networks to capture artifact defect between deepfake segmented faces and their neighboring contexts. They trained two XceptionNet-based recognition systems to extract differences between foreground and background, then further trained them with the source embedding for deepfake classification. In their research, Zhu *et al.* [66] broke down face textures into several physical decomposition groups to identify the best combination for forgery detection, grouping "direct light and identity texture" as face detail and "3-dimensional shape, ambient light, and common texture" as facial trend. They developed an Xception-based Forgery-Detection-with-Facial-Detail Net, which is a two-stream network that combines feature clues from both original images and facial detail. The output was fused with three approaches: score fusion (SF), feature fusion

(FF), and halfway fusion (HF), with HF providing the best performance. They also integrated a supervised-detail guided attention module to enhance forgery detection by exploring more plausible manipulated attributes.

Multi-Stream & Multi-Stack Neural Network In [169], it was suggested to integrate an InceptionNet-based face classification stream with a triplet stream network for steganalysis feature extraction. This approach exploited the low-level noise residual features with high-level tampering artifacts to enhance detection. The final detection score was computed from the output scores of both streams. Similarly, in [170], a multi-stream network was proposed, consisting of five dedicated parallel ResNet-18s that learned the respective face regions and captured local facial artifacts. The combined learning of regional face areas with full-face artifacts improved detection performance, especially with compressed input. Li *et al.* [171] used ResNet-18 to implement face regional learning as well. They employed a Patch&Pair Convolutional Neural Networks(PPCNN) that separated images or frames into face and non-face region patches to capture inconsistencies between the foreground and background. The embeddings of patch pairs from both branches were concatenated and passed to a classifier to compute a global decision for fake or real input. This approach improved neural network generalization against cross-origin deepfakes.

In [172], a multi-stack neural network called DeepfakeStack was introduced. It consisted of two major sections: *i.* Base-Learners Creation, and *ii.* Stack Generalization. Base-Learners Creation began with initializing seven deep learning models (XceptionNet, MobileNet, ResNet101, InceptionV3, DensNet121, InceptionReseNetV2, DenseNet169) that applied ImageNet weights for transfer learning. They were then connected by replacing the topmost layer with two output layers and the softmax activation function. The authors also employed the Greedy Layer-wise Pretraining (GLP) algorithms for model training. The Stack Generalization was a meta-learner formed by a CNN classifier named DeepfakeStackClassifier(DFC). It was integrated with a larger multi-head neural network to evaluate the best detection outcome according to the predictions from each base-learner.

Shallow-CNN In [27], Afchar *et al.* proposed two shallow network architectures to capture simpler and localized patterns for discriminating mesoscopic features. The

first network, Meso-4, consists of four convolution layers, followed by a pooling layer and a fully-connected layer with a hidden layer. The second network, MesoInception-4, is an improved version of Meso-4 with the integration of inception modules. Notably, MesoInception-4 outperforms Meso-4 by a significant margin. This study achieved a detection accuracy of 95% on the FaceForensic++ dataset and serves as a benchmark for other deepfake detection tasks [170, 173–175]. In [176], the authors proposed a shallow convolutional network (ShallowNet) that effectively detects subtle differences between deepfake and authentic images. They observed that a shallower network with a max-pooling layer could perform better on low-resolution images. However, while it significantly reduces the training time, it cannot maintain high detection performance when dealing with highly compressed input [23, 170].

Attention Mechanism In [177], Fernando *et al.* introduced the hierarchical attention memory network (HAMC), which incorporates an attention mechanism and a bidirectional GRU to extract facial attribute features for deepfake future semantic anticipation. The idea is to evaluate an unseen deepfake based on previously seen deepfake samples. To achieve this, they extracted local patches of feature embeddings using a pre-trained ResNet, passed them to the bidirectional GRU, and applied different weights and biases to foster learning of patch features at different attention levels. They used an adversarial training approach to train the output encodings and ground truth with a discriminator for final deepfake classification. Similarly, in [77], Dang *et al.* proposed inserting an attention map to the backbone network to enhance the feature map for deepfake classification. Their approach involved using the idea of camera model identification and using the 'fingerprint' in the source image to discriminate between deepfake and source data. The attention map contained various receptive fields and encoded the high-frequency fingerprint for classification. In contrast to [77, 177], Zhao *et al.* in [178] introduced a multiple-attentional framework that employs EfficientNet-b4 as the network backbone to extract and aggregate low-level texture and high-level semantic features of multiple attention maps for forgery detection. They applied a regional independence loss function, the bilinear attention pooling loss (BAP), and an attention-guided data augmentation mechanism to regularize each attention map in learning different semantic regions and non-overlap discriminative feature information. They hypothesized that the subtle differences of low-level texture mostly disappear in the deeper layer and

demonstrated that enhancing the textural feature from shallow layers helps stimulate the learning of discriminative features in the forged region.

Contrastive Loss & Triplet Loss To improve the generalization of the detector, Hsu *et al.* [146] proposed the use of contrastive loss in their deep forgery discriminator (DeepFD) to learn discriminative features across different GANs. The network architecture was similar to a siamese network, and the contrastive loss was computed based on pairwise learning. This study was further enhanced with improved algorithms in [179, 180]. In [179], the authors introduced a Common Fake Feature Network (CFFN) that implemented DenseNet with cross-layer features, significantly improving performance in both feature extraction and fake image recognition. They applied and trained the contrastive loss with the CFFN in a novel two-step learning policy. Similarly, Zhuang *et al.* [180] also adopted a two-step learning policy for model training, but they utilized a triplet loss for optimization instead of a contrastive loss. This alteration was due to the poor performance of contrastive loss when dealing with data in the same category, such as Fake-Fake or Real-Real. The model can prominently differentiate the positive and negative samples with triplet loss. They presented a new siamese network structure called Coupled Deep Neural Network (CDNN) to capture local and global features and achieved high precision of 98.6% in detection. However, the result of [146, 179, 180] might drop if handling test data with distorted spatial information or different resolutions.

Mittal *et al.* [181] applied facial and speech embedding vectors with a triplet loss function, aiming to maximize the similarity between modalities and source video and minimize the similarity between modalities and deepfake video. They integrated the memory fusion network (MSN) with CNN for deepfake prediction.

CNN Rearchitecture Reconstructing CNN architecture is a common technique to improve deep learning classification efficiency, involving the alteration and enhancement of designing the kernel, filter, convolutional layer, and hyper-parameters. In [182], the authors hypothesized that the residual domain could reflect the discriminative feature. They restructured the CNN by integrating a high pass filter, transforming the input image into residuals, and fed them to a three-layer convolutional network for training. They experimented with three sets of high pass filters, starting from the low to high filter dimensions. The result showed that the highest dimension high pass filter obtained the

lowest detection accuracy, while the rest attained similar results. Adequate pairing of a high pass filter is required to enhance the performance of the CNN in achieving high detection accuracy.

Guo *et al.* [183] suggested using the adaptive convolutional layer to improve the detection accuracy. They proposed an adaptive manipulation traces extraction network (AMTEN) that extracted the feature map from the input image and used it to subtract the source image to obtain low-level manipulation traces. The hierarchical feature extraction is formed by repeating the procedure to acquire higher-level discriminative features with the subsequent convolutional layers.

In contrast to papers [182, 183], Do *et al.* in [184] devised a VGGNet-based face detection network (VGGFace), emphasizing feature extraction and hyper-parameter fine-tuning based on face recognition.

Capsule Network The major limitation of CNN, according to Hinton *et al.* in 2017, was the lack of consideration for relative spatial and orientation relationships during network training. To address this, they proposed a more robust network architecture based on the capsule concept. In [174], the authors developed a capsule-forensic network for deepfake detection, utilizing VGG-19 for latent feature extraction and comprising three primary and two output capsules. They improved upon the algorithm from [185] by introducing Gaussian random noise to the 3D weight tensor and implementing an extra squash function before routing by iterating. The agreement between the low-level and high-level capsules will predict the probability of the input being fake or real. They subsequently published a more detailed capsule architecture in Nguyen19, which yielded better performance.

Recurrent Neural Network (RNN) The Recurrent Neural Network (RNN) is widely utilized to explore temporal features. In [186], the authors identified that DenseNet outperformed ResNet in feature extraction and face alignment in preprocessing can enhance the training performance. They also discovered that evaluation on a sequence of images provided better results than a single frame input, and the bidirectional recurrent network outperformed the uni-directional recurrent network. They concluded that the best approach is to preprocess face alignment using the facial landmark method. The

authors of [187] proposed leveraging temporal inconsistencies across frames by using the capsule network as a feature extractor and feeding the output sequence to the LSTM to capture the temporal features. They found that equal interval frame selection provides better performance. However, the performance can be degraded if there are consistent discrepancies across frames. Amerini *et al.* [188] preprocessed and transformed the input to compute a set of correlated inter-frame prediction errors, which they utilized to capture the temporal correlation among the consecutive frames via sequence learning with the CNN and LSTM. Masi *et al.* [189] implemented a two-branch structure to encode the color domain and frequency with the Laplacian of Gaussian layers (LoG) to amplify the deepfake artifacts. They fed the combined feature maps to the bi-directional LSTM for further time-series training and classification. Similarly, Sun *et al.* [190] adopted a two-stream network to mine geometric features from the extracted facial landmarks. They preprocessed the video into frames and extracted the facial landmarks using Dlib. Two different feature vectors were generated from the facial landmarks and input separately to each branch of the two-stream RNN. However, their performance dropped when tested with the CelebDF dataset, showing their incapability in assuring model generalization. Moreover, the complicated calibration process in mining geometric features might make it difficult for duplication.

Autoencoder In *Locality-aware AutoEncoder (LAE)*, introduced by Du *et al.* [175], they proposed a solution to prevent overfitting in the detection model. They achieved this by utilizing the latent space loss and reconstruction loss to enforce the semi-supervised learning of the data's intrinsic representation. In a similar vein, Khalid *et al.* [191] developed a one-class variational autoencoder named OC-FakeDect to detect deepfakes based on image reconstruction. They utilized an anomaly score computed by Root Mean Squared Error (RMSE) between the source and reconstructed images of VAE to formulate a threshold to distinguish deepfake data.

Multi-Person Forgery In 2021, Zhou *et al.* [192] published a novel wild dataset consisting of an average of three people per scene to simulate real-case scenarios. They addressed multi-person forgery with a multi-temporal instance learning methodology, which consists of three significant modules: *i.* A multi-temporal scale instance feature aggregation module, *ii.* An attention-based bag feature aggregation module, and *iii.*

A sparse attention regulation loss. As most approaches focus on single-face forgery detection, this study provides important insight into effectively detecting deepfakes with multiple persons per scene using lesser label cost.

Self-supervised learning Self-supervised learning has become a popular method for improving the ability of image classification and object detection models to generalize to unseen datasets [193–195]. In the field of deepfake detection, this approach has also been adopted in recent studies [161, 178, 196]. For example, Zhao *et al.* [178] proposed pairwise self-consistency learning, which uses source features to detect deepfakes based on data inconsistency. Zhang *et al.* [161] introduced a self-supervised decoupling network to enhance feature representations under varying compression factors. They enforced similarity feature learning between authentic and compression features during the similarity decoupling stage and performed adversarial decoupling to train the model to learn more robust and sophisticated facial features. Meanwhile, Chen *et al.* [196] used adversarial data augmentations to improve the generalizability of the detector. Similar to [161], training was conducted in an adversarial manner.

Table 3.6, Table 3.7, Table 3.8, and Table 3.9 provide a comprehensive overview of various deepfake detection approaches. The performance of each approach is evaluated using four different metrics, namely Accuracy (ACC), Area under the ROC Curve (AUC), Equal Error Rate (EER), and False Rejection Rate (FRR). In each table, the detection results presented for each paper are prioritized based on the most commonly used evaluation metric. These tables provide valuable insights into the performance of various deepfake detection methods and can help researchers and practitioners in selecting an appropriate approach for their specific application.

3.2.3 Deepfake Datasets

In the past two years, diverse datasets have been released to facilitate research and experimentation related to deepfakes. Understanding the characteristics of the datasets, such as quality, quantity, and manipulation techniques, is crucial to avoid overfitting or underfitting during practical training. Therefore, this section will discuss the popular deepfake datasets that are publicly available. Table 3.10 summarizes the discussed deepfake datasets.

The UADFV dataset, which was created by Yang *et al.* [142, 149] for their deepfake detection experiments, comprises 49 real videos collected from YouTube and 49 fake videos generated using the FakeApp application. In the real videos, the faces of the original individuals were replaced with those of an American actor named Nicolas Cage, resulting in him being the sole identity depicted in all of the fake videos. The dataset includes an equal number of authentic and synthetic videos, with each video having a resolution of 294 x 500 pixels.

DeepfakeTIMIT is a video dataset [197] that has been modified from the VidTIMIT dataset [198]. To create this dataset, the authors used an open-source FaceSwap-GAN approach for face-swapping. They manually selected 16 pairs of individuals to create 620 face-swapped deepfake videos from 32 subjects. The original audio channel of the videos was not manipulated. The fake videos have two different quality standards: *i.* low quality (LQ) images of 64×64 pixels, and *ii.* high-quality (HQ) images of 128×128 pixels.

The **Fake Faces in the Wild (FFW)** dataset was proposed by Khodabakhsh *et al.* [199] to serve as a benchmark for evaluating the generalizability of fake face detection. The dataset consists of 150 source videos collected from YouTube, which were manipulated using various techniques such as deepfake, computer graphics image (CGI), and splicing.

FaceForensics++ (FF++) [23] is a well-known deepfake dataset that includes various types of deepfakes. The authors manipulated 1,000 YouTube source videos with four advanced face manipulation algorithms, such as Deepfakes, Neural Texture, Face2Face, and FaceSwap, resulting in 4,000 fake video sequences. To ensure a more comprehensive evaluation, the authors compressed the H.264 format videos using compression rate factors of 0, 23, and 40. Later, Google and Jigsaw upgraded the dataset as the DeepfakeDetection dataset (DDD/DFD) by adding another 363 original videos. Additionally, 28 actors were hired to create 3,086 high-quality deepfake videos in 16 different scenarios.

Liu *et al.* [25] presented a large-scale dataset called **Celeb-DF**, which includes 590 real and 5,693 high-quality fake videos. The authors collected source videos from YouTube featuring subjects of diverse ages, ethnic groups, and genders to create this dataset that aims to simulate real-life scenarios with various video qualities.

In late 2019, Facebook collaborated with numerous technology giants and industry and academic experts to organize the **Deepfake Detection Challenge (DFDC)** and launched a dataset collection campaign. The organizers hired several actors to record videos for the dataset, and the challenge released the dataset in two phases: *i.* The preview dataset [22], and *ii.* The final version [26]. The preview dataset includes 1,131 real videos and 4,113 fake videos, while the final version comprises 19,154 real videos and 100,000 deepfake videos produced by 3,426 paid actors. Both versions contain a rich variety of gender, skin tones, lighting conditions, head poses, ages, and backgrounds.

Jiang *et al.* published **DeeperForensices-1.0** [28], which comprises 50,000 real and 10,000 high-quality fake videos originating from 100 paid actors. To simulate real-world scenarios, they employed various perturbations, including compression, blurriness, and transmission errors, ensuring dataset diversity.

He *et al.* [200] introduced the **ForgeryNet Dataset**, which includes over 36 mix-perturbations using 15 manipulation techniques on more than 54k subjects. The authors used CREMA-D [201], RAVDESS [202], VoxCeleb2 [203], and AVSpeech [204] as source data to increase diversity from different aspects, such as facial expression, face identity, subject angle, and case scenarios. They used face swapping, face reenactment, deep-fake, and identity transfer as the primary manipulation techniques. The entire dataset comprises two main groups: *i.* Image-forgery subset with 2.9M still images, and *ii.* Video-forgery subset with over 220k video clips. It defines four significant tasks (image and video classification, spatial and temporal localization) with 9.4M annotations.

Zhou *et al.* introduced **Face Forensics in the Wild** to address multi-person face forgery detection [192]. The authors gathered 4k raw source videos from YouTube, with a minimum resolution of 480p, and divided each video into four equal clips. From each clip, they randomly selected a 12s sequence for generating forgeries. They used DeepFaceLab, FS-GAN, and a FaceSwap graphic method to perform face-swapping by selecting two videos randomly from a filtered sequence collection of 12k videos. The dataset comprises an average of three human faces in each frame and underwent an automated manipulation process using a domain-adversarial quality assessment network to reduce costs.

3.2.4 Evaluation Metrics

There are four possible classification scenarios for deepfake classification, *i.* True Positives (TP), where the data (deepfake) is correctly classified as positive (deepfake); *ii.* True Negatives (TN), where the data (real) is correctly classified as negative (real); *iii.* False Positive (FP), where the data (real) is incorrectly classified as positive (deepfake); and *iv.* False Positive (FN), where the data (deepfake) is incorrectly classified as negative (real). Figure 3.4 shows the example of a deepfake detection confusion matrix.

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

TABLE 3.4: Confusion matrix

The evaluation metrics applied these four scenarios for performance measurement and the common evaluation metrics used for deepfake detection experiments are Accuracy (Acc.), Area Under the Curve (AUC), Receiver Operating Characteristic (ROC), Recall, Precision, F1 Score, Equal Error Rate (ERR), False Acceptance Rate (FAR), and False Rejection Rate (FRR). Table 3.5 outlines the formula for each evaluation metric.

3.2.5 Discussion

Figure 3.6 illustrates the frequency of deep learning and non-deep learning-based deepfake detection models used in the studies mentioned earlier. The analysis reveals that more researchers are opting for deep learning-based approaches than traditional handcrafted feature-based techniques. This trend is mainly due to the high quality of deepfakes, which leaves minimal traces and anomalies in the intrinsic features of deepfake images or videos, making it challenging to extract handcrafted features. Moreover, with the introduction of more efficient CNN architectures in recent years, many researchers have shifted their focus to learned feature extraction. Figure 3.7 shows the frequency of popular CNNs used for the deepfake detector in the discussed studies.

Recent research has shifted the focus from a model-centric approach to a data-centric methodology. In [205], the authors introduced a representative forgery mining (RFM)

Evaluation Metric	Formula	Explanation
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$ (3.1)	The proportion of correct predictions among the total number of predictions made by the model.
True Positive Rate (TPR)	$\frac{TP}{TP+FN}$ (3.2)	The rate of positive samples classified as negative.
False Positive Rate (FPR)	$\frac{TN}{TP+FN}$ (3.3)	The rate of positive samples classified as negative.
Recall	$\frac{TP}{TP+FN}$ (3.4)	The proportion of true positives among the total number of actual positive samples in the dataset.
Precision	$\frac{TP}{TP+FP}$ (3.5)	The proportion of true positives among the total number of positive predictions made by the model.
F1-Score	$2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$ (3.6)	The harmonic mean of precision and recall, which provides a balance between the two metrics. A model with high precision and high recall will have a higher F1-Score than a model with either high precision or high recall alone.
False Positive Rate/ FRR	$\frac{FN}{FN+TP}$ (3.7)	The proportion of actual positives that are incorrectly predicted as negatives by the model.
False Negative Rate/ FAR	$\frac{FP}{FP+TN}$ (3.8)	The proportion of actual negatives that are incorrectly predicted as positives by the model.
ERR	$\frac{FP+FN}{TP+TN+FP+FN}$ (3.9)	The point at which the False Acceptance Rate (FAR) and the FRR are equal, which indicates the point of the optimal trade-off between the two.

Table 3.5: Formula and explanation for each evaluation metric

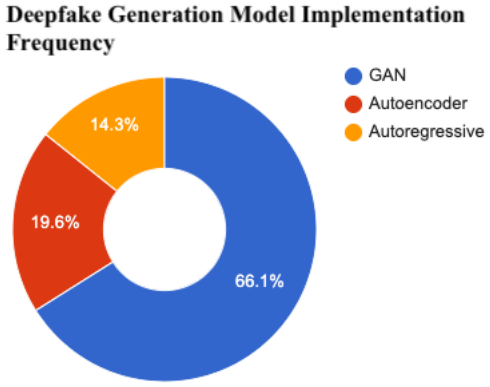


FIGURE 3.6: The implementation frequency of deep learning & non-deep learning-based deepfake detection model structure based on discussed studies

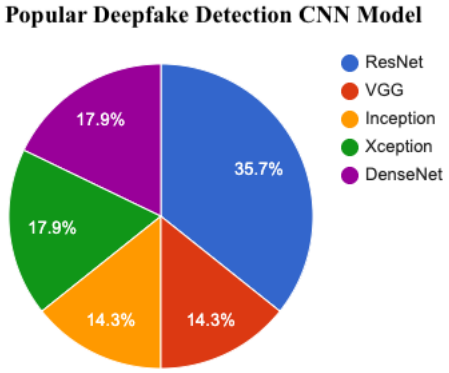


FIGURE 3.7: The implementation frequency of the popular CNN architecture for deepfake detection model based on discussed studies

framework that refines the training data to enhance the performance of the vanilla CNN-detector. The proposed RFM framework is applicable to any CNN-based detector and provides a significant visualization result for exploring the forgery region of different manipulation techniques.

3.3 Challenges

The deepfake generation and detection can be compared to a cat-and-mouse game where improving the generator leads to the advancement of the detector. Conventional methods show that designing a detector based on a particular generator's weaknesses, such as traces or anomalies, is not a sustainable, reliable, and flexible solution. As deepfake generators aim to produce artifact-less results, the trend in detector research has shifted towards discrimination based on learned features instead of handcrafted ones. However,

No.	Ref.	Pub.	Year	Handcrafted Features			Classifer	Dataset	Performances
				Color	Texture	Spatial			
1	[135]	ICSIP	2017			✓	SVM	LFW Face Database	0.929 (ACC)
2	[136]	IJCB	2017	✓		✓	SVM	SWAPPED digital attack video face database	24.50 (EER)
3	[137]	IMVIP	2018			✓	-	Self-Created Videos	-
4	[140]	Signal Process.	2018	✓			Binary, SVM	CelebA, CelebA-HQ, Labeled Faces in the Wild	1.0 (ACC)
5	[141]	ArXiv	2018	✓		✓	SVM	LSUN	0.700 (AUC)
6	[138]	ArXiv	2019			✓	SVM	Faces-HQ , CelebA , FF++	1.0 (ACC)
7	[142]	ICASSP	2019		✓		SVM	UADFV , DARPA MediFor GAN Image,Video Challenge	0.890 (AUC)
8	[145]	WACVW	2019		✓		Logistic Regression	CelebA , ProGAN, Glow, FaceForensics	0.866 (AUC)
9	[162]	ICME	2021			✓	XGBoost	UADFV, Celeb-DF V1, Celeb-DF V2, Face Forensics ++	0.959 (AUC)

TABLE 3.6: Summary of pure handcrafted feature & handcrafted-feature with machine learning based deepfake detection approaches

No.	Ref.	Pub.	Year	Techniques							Model	Dataset	Performances
				Color	Texture	Spatial	Att.	Loss	NN Rearchi-tecture	Multi Stream/ Net-work			
1	[168]	CVPR	2018		✓						ResNet, VGG	UADFV , Deepfake-TIMIT	0.990 (AUC)
2	[149]	WIFS	2018		✓						CNN, LSTM	CEW, Eye Blink-ing Video (EBV)	0.990 (AUC)
3	[169]	CVPRW	2018							✓	Inception	SwapMe and FaceSwap dataset	0.928 (AUC)
4	[27]	WIFS	2018						✓		Inception	FF++	0.984 (ACC)
5	[146]	IS3C	2018					✓		✓	CNN	GAN-Generated Images Based On CelebA	0.947 (ACC)
6	[182]	IH and MMSec	2018						✓		CNN	CelebA-HQ	0.980 (ACC)
7	[184]	ISITC	2018						✓		CNN, VGG	CelebA , DC-GAN and PG-GAN generated images	0.800 (ACC)
8	[177]	IEEE TIFS	2019				✓				ResNet, GRU	FaceForensics, FF++, FakeFace in the Wild	0.999 (ACC)
9	[176]	MPS	2019						✓		CNN	CelebA , PG-GAN generated images	0.999 (ACC)
10	[156]	ICCVW	2019	✓		✓					ResNet, VGG	FF++	0.816 (ACC)
11	[174]	ICASSP	2019						✓		Capsule	FF++	0.994 (ACC)
12	[173]	Voice-Personae Project	2019						✓		Capsule	FF++	0.931 (ACC)
13	[186]	CVPR	2019						✓		DenseNet, GRU	FF++	0.969 (ACC)
14	[159]	IEEE Access	2019	✓							CNN	CASIA, GPIR, COV-ERAGE, BigGANs, LSUN Bedroom, PGGAN, SNGAN, StyleGAN	0.975 (ACC)
15	[180]	ICIP	2019					✓		✓	DenseNet	GAN-Generated Images Based On CelebA	0.986 (ACC)

TABLE 3.7: Summary of deep Learning-based deepfake detection part 1

No.	Ref.	Pub.	Year	Techniques							Model	Dataset	Performances
				Color	Texture	Spatial	Att.	Loss	NN Rearchitecture	Multi Stream/ Network			
16	[179]	Applied Sciences	2020					✓		✓	DenseNet	CelebA , ILSVRC12	0.988 (ACC)
17	[170]	WACV	2020							✓	ResNet	FF++	0.999 (ACC)
18	[158]	ArXiv	2020	✓							RNN	FF++, DDD	0.990 (ACC)
19	[157]	ACM-MM	2020			✓					ResNet, LSTM	FF++, DFDC Preview	0.997 (ACC)
20	[150]	BIOSIG	2020		✓						LightCNN, ResNet, DenseNet, SquezeNet	Celeb-DF	0.879 (AUC)
21	[187]	Master Thesis	2020			✓					Capsule, VGG, LSTM	DFDC	0.834 (ACC)
22	[160]	BIOSIG	2020	✓		✓					ResNet	FF++	0.993 (ACC)
23	[175]	CIKM	2020					✓			Autoencode, UNet	FF++	0.968 (ACC)
24	[183]	ArXiv	2020	✓		✓					CNN	CelebA , CelebA-HQ, GANs-generated dataset	0.985 (ACC)
25	[77]	CVPR	2020			✓	✓				CNN	CelebA, Flicker-Faces-HQ (FFHQ), Face Forensics++, GANs-generated dataset	0.997 (ACC)
26	[167]	IJCAI	2020		✓		✓				CNN	CelebA , FFHQ, Face Forensics++, GANs-generated dataset, DFDC, CelebDF	0.986 (ACC)
27	[171]	TheWebConf	2020							✓	ResNet	Face Forensics++, Deepfake-TIMIT, Mesonet data	0.994 (ACC)
28	[206]	ArXiv	2020		✓	✓					Inception	FF++, Celeb-DF-v2, DFDC	0.997 (AUC)
29	[189]	ECCV	2020			✓					CNN, LSTM	FF++, Celeb-DF, DFDC	0.943 (ACC)

TABLE 3.8: Summary of deep Learning-based deepfake detection part 2

No.	Ref.	Pub.	Year	Techniques							Model	Dataset	Performances
				Color	Texture	Spatial	Att.	Loss	NN Rearchitecture	Multi Stream/ Network			
31	[153]	CVPR	2020		✓						CNN	A2V, T2V-S, T2V-L, FakeFace in-the-wild	0.997 (ACC)
32	[188]	IH and MMSec	2020			✓					LSTM	FF++	0.943 (ACC)
33	[172]	Edge-Com	2020							✓	XceptionNet, InceptionV3, Inception-ResNetV2, MobileNet, ResNet, Densenet	Self-generated dataset	0.997 (ACC)
34	[191]	CVPR	2020			✓					Autoencoder	FF++	0.982 (ACC)
35	[152]	IEEE TIFS	2021		✓						CNN	MOBIO, GRID	2.80 (FRR)
36	[161]	ICME	2021			✓					MTCNN, EfficientNet-B2	FF++	0.918 (ACC)
37	[207]	CVPR	2021	✓	✓	✓					Xception	Face Forensics ++, DFD, DFDC	0.995 (AUC)
38	[163]	CVPR	2021	✓				✓			Xception	FF++	0.967 (ACC)
39	[164]	CVPR	2021			✓					Xception	FF++, DeepfakeDetection(DFD), CelebDF, DeeperForensics-1.0	0.994 (AUC)
40	[190]	CVPR	2021		✓					✓	RNN	Face Forensics++, UADFV, CelebDF	0.999 (AUC)
41	[155]	CVPR	2021		✓						ResNet, MS-TCN	Face Forensics++, DFDC, CelebDF, DF1.0, FaceShifter	0.988 (ACC)
42	[178]	CVPR	2021		✓		✓	✓			EfficientNet-B4	Face Forensics++, DFDC, CelebDF, DF1.0	0.976 (ACC)
43	[166]	CVPR	2021			✓					Xception	Face Forensics++, DFDC, CelebDF	0.816 (ACC)
44	[192]	CVPR	2021			✓				✓	ResNet	Face Forensics++, DFDC Preview, CelebDF, FFW	0.993 (AUC)
45	[196]	CVPR	2022		✓					✓	Xception	FF++, DFDC, CelebDF, DF1.0	0.984 (ACC)

TABLE 3.9: Summary of deep Learning-based deepfake detection part 3

Dataset	Ref.	Year	Manipulation	Ratio (Real: Fake)	Total Videos	Resolution	Format	Source	Participant Consent
UADFV	[142]	2018	FakeApp	1.0: 1.0	49 real videos, 49 fake videos	294p x 500p	YouTube	YouTube	N
Deepfake-TIMIT	[197]	2018	FaceSwap-GAN	Only Fake	0 real videos, 620 fake videos	64p x 64p - 128p x 128p	JPG	Actor	Y
FFW	[199]	2018	Splicing, CGI, Deepfake	Only Fake	0 real videos, 150 fake videos	480p, 720p, 1080p	H.264, YouTube	YouTube	N
FF++	[23]	2019	FaceSwap, Deepfake	1.0: 4.0	1,000 real videos, 4,000 fake videos	480p, 720p, 1080p	H.264, CRF=0, 23, 40	YouTube	N
DFD/DDD	[24]	2019	Deepfake	1.0: 8.5	363 real videos, 3,086 fake videos	1080p	H.264, CRF=0, 23, 40	Actor	Y
Celeb-DF	[25]	2019	Deepfake	1.0: 11.3	590 real videos, 5,639 fake videos	Various	MPEG4	YouTube	N
DFDC-preview	[22]	2019	Deepfake	1.0: 3.6	1,131 real videos, 4,113 fake videos	180p - 2160p	H.264	Actor	Y
DFDC	[26]	2019	Deepfake	1.0: 5.2	19,154 real videos, 100,000 fake videos	240p - 2160p	H.264	Actor	Y
Deeper-Forensics-1.0	[28]	2020	Deepfake	5.0: 1.0	50,000 real videos, 10,000 fake videos	1080p	MP4	Actor	Y
ForgeryNet Dataset	[200]	2021	FaceSwap	1.0: 1.2	99630 real and 121617 fake videos, 1,438,201 real and 1,457,861 fake images	240p - 1080p	YouTube	Actor/ YouTube	Y/N
Face Forensics in the Wild	[192]	2021	FaceSwap	1.0: 1.0	10,000 real videos, 100,000 fake videos	¿480p	YouTube	YouTube	N

TABLE 3.10: List of deepfake dataset

pre-trained CNN models may not perform well with different deepfake scenarios and can be vulnerable to malicious attacks. Addressing these drawbacks may bring the deepfake detector's performance to a higher level.

The creation of state-of-the-art deepfakes heavily relies on GAN technology. Researchers have improved deepfake network training by integrating tertiary concepts such as style transfer, motion transfer, biometric artifacts, and semantic segmentation to achieve more hyperrealistic and natural results with high confidence [208–210]. However, current deepfakes are still imperfect and leave room for improvement. GAN training is time-consuming, resource-intensive, and susceptible to overfitting, and the output is not flawless enough to evade detection.

3.3.1 Opportunities in Deepfake Generation

Few-shot Learning The availability of a large training dataset has been a significant challenge in the deepfake area because most deepfake generation techniques require a vast amount of genuine data to support the training in creating more convincing fake content, resulting in increased computational resources. To address this issue, the research community has focused on training with as little dataset as possible, specifically training on few-shot learning. Few-shot learning is a domain field of meta-learning that requires only a relatively small dataset and low data labeling costs for training, unlike previous deepfake generation methods. One of the popular few-shot learning methods is *N-way K-shot* classification, which utilizes transfer learning and knowledge sharing to match the training likelihood distribution to the few-shot support set, showing the potential of creating deepfakes. In [91, 211], the authors successfully implemented few-shot learning to reduce the required computational training resources. The research trend of reducing computational power and training datasets for deepfakes has consistently driven the development of deepfake research.

Deepfake Quality Due to the instability of GAN training, most deepfake outputs contain subtle traces or fingerprints, such as unusual texture artifacts or pixel inconsistencies, making them vulnerable to detection. Additionally, the current research has mostly focused on non-occlusive frontal face training data for deepfake generation, which may not maintain output quality when occlusions occur in the input data. To address

this issue, Li *et al.* [121, 212] proposed a mask-guided detection approach. Future studies could aim to ensure deepfake quality through artifact elimination, high output resolution, and the ability to generate deepfakes that defend against attacks.

Real-time Deepfake The use of deepfakes to achieve real-time face transformation effects online has presented a new opportunity for the technology. One team of researchers published open-source software [128] based on [213] to promote real-time deepfake usage in video conferencing. This software utilizes image animation techniques based on keypoint learning and affine transformation, then applies the training results directly to the desired input image to achieve real-time reenactment. However, since it is an image animation technique, the results may contain biometric artifacts and low fidelity for specific facial expressions and head movements. Despite its imperfections, this technology offers insight into the development of future real-time deepfake applications, which could have both positive and negative impacts on industries such as medicine, education, and entertainment.

3.3.2 Opportunities in Deepfake Detection

Adversarial Attack The current focus on deepfake detection performance has resulted in a neglect of the importance of robustness. A small perturbation to the input can significantly impact the performance of a trained neural network detection model, causing it to deviate from the expected results. This is where adversarial samples come into play, as they can be used to help deepfake data evade detection. Two standard threat models for adversarial attack are black-box and white-box models, which are classified based on their knowledge and access to the target detector [214]. Wang *et al.* [215] have proposed a stochastic-based defense mechanism that involves switching a model's block layers to parallel channels and randomly allocating active channels during runtime. As deepfake detection has shifted towards neural network training, studying adversarial attack defense mechanisms in this field could provide new opportunities for exploration and development.

Model Generalization Despite extensive efforts devoted to distinguishing deepfakes, detecting different types of deepfakes, data diversity, and data resolution remains a

significant issue for detector generalization. While achieving outstanding results on planned cases, detector performance drastically drops when facing these scenarios. For instance, a detector designed to detect Face2Face might not perform well when detecting entire face synthesis, a detector trained on a particular dataset might not handle other unseen datasets, or a detector trained with specific data resolution becomes vulnerable when tested with different input compression. Although few studies [160, 175] have examined generalizable detectors, none of them have achieved a consistent detection accuracy with an acceptable 5% deviation when considering all three factors in their works. Detector generalization is undoubtedly an important and rising trend in deepfake detection research.

Application & Platform-friendly To ensure protection against deception from deepfake data, it is essential to convert the deepfake detector into a dependable feature or API. Additionally, there is an opportunity to develop a user-friendly tool that can be integrated with social media platforms or applications, enabling people to make accurate judgments and protect themselves from false information.

3.3.3 Summary

The training of a deepfake detector model incurs a potentially high computational cost, thereby presenting a challenging hurdle for replication [49]. Simultaneously, the research community remains deeply concerned about the reliability and generalizability of deepfake detection models. The production of deepfakes employs diverse methods, resulting in distinct intrinsic data representations that could potentially affect the generalization capabilities of the detector. Additionally, the detector might excessively specialize in specific feature representations, leading to diminished reliability when applied to previously unseen datasets lacking diversity [160, 175]. Adversarial attacks pose a significant threat to neural network-based approaches, where even minor perturbations can mislead the classifier, causing unforeseen behaviors [216]. Numerous studies [217–220] have explored various strategies, such as manipulating victim and threat models and employing adversarial learning for detecting cues, in order to exploit adversarial attacks and evade deepfake detection. These studies reveal that the performance of most neural network-based deepfake detectors can be substantially influenced by incorporating a minimal

number of adversarial samples. However, this particular aspect has scarcely been taken into account during the design of neural network-based deepfake detectors. By delving into these existing research gaps, this thesis aims to address the aforementioned issues comprehensively.

Chapter 4

A shallower and spatially cost-efficient network structure (SparcoNet)

This chapter consists of part of the following publication:
Jia Wen Seow, Mei Kuan Lim, Raphaël C.W. Phan, & Joseph K. Liu (2023). SparcoNet with Block-Switched Self-Supervised Learning: An Effective Framework for Deepfake Detection with Improved Adversarial Defense. Submitted to Information Sciences Journal. [**Submitted**]

4.1 Motivation

The trend in designing deepfake detectors has shifted from traditional handcrafted methodologies to deep learning approaches. This concern is due to the vulnerability of traditional detectors against advanced deepfake data. Most current deep learning-based deepfake detection models use deeper network structures to emphasize local feature learning. However, these deeper network models are often more complex, require

greater computational power for training, and are harder to reproduce or implement with limited computing resources [221].

In 2018, Mesonet [27] demonstrated the potential of an intermediate network structure by achieving a promising and competitive result at a lower computational cost. This inspired us to develop SparcoNet, a shallower and spatially cost-efficient network structure. We monitored the behavior and limitations of Mesonet during training and utilized this information to design a more effective network architecture through fine-tuning. Therefore, our proposed model can be considered a monitoring-based enhanced model [222].

4.2 SparcoNet Development

The development of SparcoNet involves two primary network structures: the Inception module and the Residual Connection.

The convolutional neural network (CNN) architecture proposed by Simonyan *et al.* [50] demonstrated that using a small 3x3 filter size improves classification accuracy when the network depth is around 16-19 weight layers. Camg"ozl"u *et al.* [223] suggested that a smaller filter size could reduce data loss during dimension reduction and provide better performance in accuracy and processing time. However, the basic idea behind inception modules is that different convolution filter sizes can capture different features at different scales. For instance, small filters are good at capturing local, fine-grained details, while larger filters are better at capturing coarse, global features. The network can capture a wider range of features across different scales by using filters of various sizes in parallel, forming several parallel convolutional paths. The outputs of these paths are then concatenated and passed on to the next layer. The model can achieve high accuracy across various tasks by increasing the network width. This network structure was proposed in the Inception network [4]. Figure 4.1 shows the structure of an inception module.

On the other hand, the Residual Connection, implemented in ResNet, helps prevent overfitting and improve performance on deeper networks [5]. In a typical neural network, the output of one layer is passed as input to the next layer, and as the network gets deeper, the gradients can become very small. This makes it difficult to update

the weights of the earlier layers during backpropagation, resulting in a degradation of performance compared to shallower networks.

Residual connections address this issue by allowing the network to learn residual mappings, which are the differences between the input and output of a given layer. These mappings are added to the original input, creating a "shortcut" connection that bypasses subsequent layers. This shortcut enables the gradients to flow directly from the output of a layer to the input of an earlier layer during backpropagation, reducing the vanishing gradient problem and enabling the network to learn deeper and more complex representations. In essence, the residual connection provides a way for the network to learn the underlying structure of the input data more efficiently and effectively. Figure 4.2 shows the structure of the residual connection network. The Inception-ResNet [6] implemented both structures; however, it is computationally expensive and requires more training times as the network structure is more complex. Figure 4.3 shows the structure of an inception module with the residual connection.

Inspired by Mesonet [27], which proposed an intermediate network structure, we created SparcoNet, which enhanced a shallower deep neural network by exploiting the structure advantages from both the inception module and residual connection.

In our network, we embraced the foundational concept of inception modules and took a unique approach. Rather than integrating various filter sizes that could potentially deepen the network size, we opted for a consistent filter size of 3 in each separable convolutional layer within the inception modules. This strategic choice serves to not only enhanced the consistency of spatial relationships but also significantly reduce the computational burden. Moreover, we harnessed the power of dilated convolution within the inception block, resulting in a parallel increase in the scale of feature learning. The dilated convolution within the inception block helps to expand receptive field exponentially without increasing the complexity of the network, and at the same time helps overcome the issue of overfitting. As a result, we achieved outcomes comparable to those attainable with a deeper network layer while maintaining a more budget-friendly computational overhead.

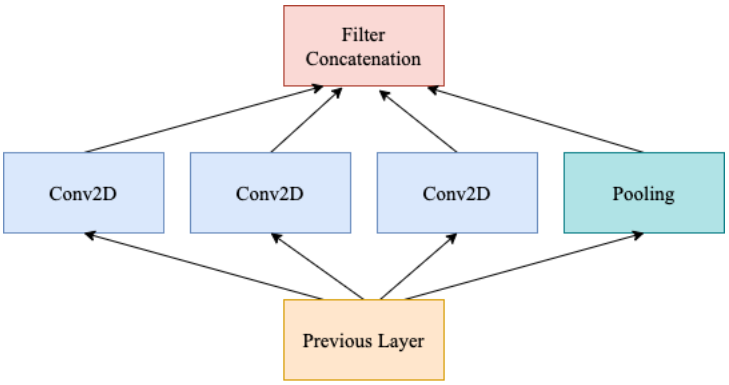


FIGURE 4.1: The structure of an Inception module [4]

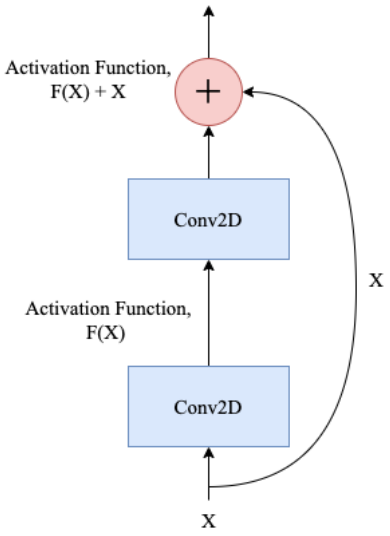


FIGURE 4.2: The structure of a residual connection [5]

4.2.1 Model Architecture

Figure 4.4 shows the workflow architecture of SparcoNet. The entire pipeline begins by taking a dataset of videos or images as input. The dataset undergoes data preprocessing, which includes slicing the data into image frames if it is a video dataset. Next, the MTCNN detector extracts the facial region of each data. Subsequently, all image data are resized to 256 x 256 x 3 pixels before being fed into the feature extractor module. After feature extraction, the resulting embeddings are passed to a sigmoid classifier for the final deepfake classification.

The convolutional neural network (CNN) architecture proposed by Simonyan *et al.* [50] demonstrated that using a small 3x3 filter size improves classification accuracy when the network depth is around 16-19 weight layers. Camg"ozl"u *et al.* [223] suggested that a smaller filter size could reduce data loss during dimension reduction and provide

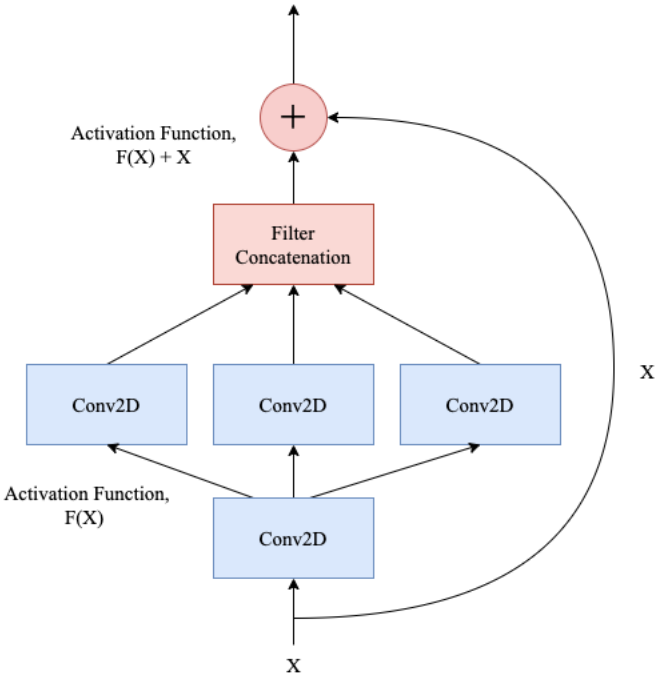


FIGURE 4.3: The structure of an Inception module with residual connection [6]

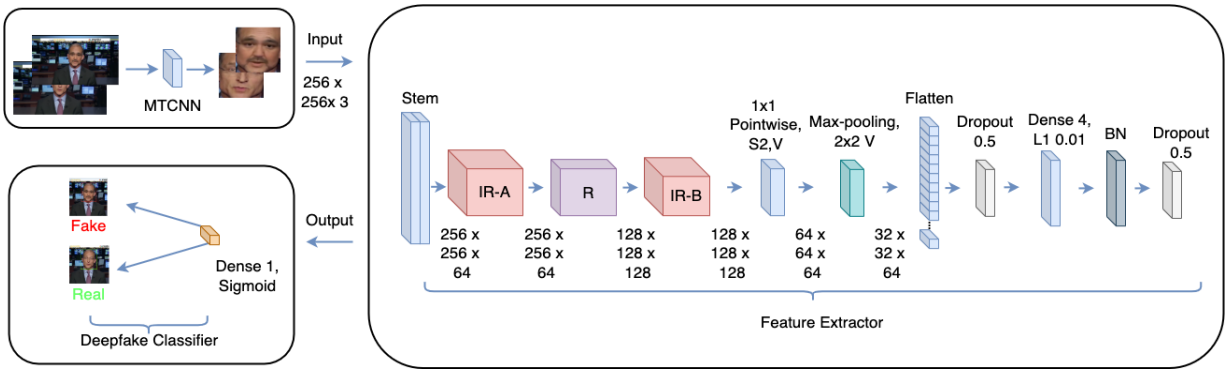


FIGURE 4.4: The architecture of the proposed SparcoNet.

better performance in accuracy and processing time. In this work, we aimed to improve the stability of spatial resolution throughout the network by implementing a consistent filter size of 3 for each convolutional layer in our proposed network.

SparcoNet consists of an initial stem, two middle blocks, a reduction block, and a projection head. We built the feature extractor using a combination of spatial separable convolution, dilated convolution, and pointwise convolution layers to capture fine-grained spatial features while reducing the computational cost. The initial stem extracts and prepares features from the input data before passing them to the middle blocks. The middle inception blocks consist of a pair of separable convolution layers, dilated convolution layer, and a pointwise convolution layer. Although we maintain the filter size as

3, the two dilated convolutions in the inception-res middle blocks helps capture the long-range spatial relationships in multiscale and avoid sub-optimal results. Instead of using a pooling layer for downsampling in the middle blocks, we implemented a consistent two-strided convolution layer-based reduction block, incorporated with the bottleneck structure, to optimize the computational cost.

We empirically increased the number of neurons in both the middle and reduction blocks to 16, 32, and 64 to increase the receptive field of our model. The selection of the number of neurons was carefully considered based on the results of our ablation study, where the model did not encounter issues of overfitting and was able to operate within our hardware's capacity. We used the inception modules structure for both blocks and added residual connections to address potential vanishing gradient issues due to the implementation of a larger number of neurons. We used the 1x1 pointwise convolution as a conjunction block for concatenation in the residual-inception modules. Details of the inner block structure are listed in Table 4.1.

Part	Structure
Stem	1x3 Conv2D (64), 3x1 Conv2D (64)
Inception-Res Middle Block A	1x3 Conv2D (16), 3x1 Conv2D (16) 3x3 Conv2D 2 Dilation Rate (16) 1x1 Conv2D (32)
Reduction Block	1x3 Conv2D (32), 3x1 Conv2D S2(32) 1x3 Conv2D (32), 3x1 Conv2D S2(32) 1x1 Conv2D S2 (64)
Inception-Res Middle Block B	1x3 Conv2D (32), 3x1 Conv2D (32) 3x3 Conv2D 2 Dilation Rate (32) 1x1 Conv2D (64)
Projection Head	1x1 Conv2D S2(64) Max-Pooling (2,2) V Flatten, Dropout(0.5) Dense(4) L2(0.01) Batch Normalization, Dropout(0.5) Dense(1), Sigmoid

TABLE 4.1: The Inner block details of the proposed SparcoNet, the (16,32,64) indicated the neuron numbers.

4.2.2 Network parameters

Floating point per second (FLOPS) is a metric used to measure the computational performance and processing speed when the model computes arithmetic operations involving floating points. This metric is often used for measuring model complexity or computational performance. We compared the network parameters, depth, and FLOPS of SparcoNet with Mesonet [27], Inception-Resnet-V2 [6], and Xception [23] in Table 4.2. Our proposed model has only 0.7% and 1.7% of the parameters of Inception-Resnet-V2 and Xception, respectively, which are often used as the backbone for deeper detection networks. Moreover, SparcoNet's depth is only six layers weight deeper than Mesonet. Our model performs better regarding accuracy and computational cost, where the accuracy is stated between [0,1].

Model	Parameters	Depth	FLOPS
MesoInception [27]	28,615	10	$1.17e^{8}$
Inception-Resnet-V2 [6]	54,339,810	36	$2.63e^{10}$
Xception [23]	22,855,952	71	$1.67e^{10}$
SparcoNet	383,297	16	$7.68e^{8}$

TABLE 4.2: Comparison of model parameters, network depth, and model complexity(FLOPS) between Mesonet, Inception-Resnet-V2, Xception network, and the proposed SparcoNet

4.2.3 Ablation study

We have conducted ablation studies to examine the effect of various structural implementation factors on a spatial-emphasized network's performance. The factors under consideration include the number of neurons, kernel size, multiscale 1x1 filter, batch normalization (BN), and middle block number. Table 4.3 presents the results of several key experiments in the ablation study, demonstrating that BN plays a crucial role in enhancing the performance of a shallower network. In fact, its inclusion resulted in an approximately 14% improvement in network performance. Of all the tested variants, the full model without multiscale 1x1 filter, using consistent 3-only kernel size with BN, and 128 neurons exhibited the best performance.

Variant	Accuracy	AUC
Full model	0.986	0.998
Without 1x1 filter, 3 only kernel size, 128 neurons, Additional BN, 1 middle block	0.941	0.990
Without 1x1 filter, 3-7 kernel size, 128 neurons, Additional BN	0.933	0.980
Multiscale 1x1 filter, 3-7 kernel size, 128 neurons, Additional BN	0.912	0.976
Multiscale 1x1 filter, 3-7 kernel size, 128 neurons	0.778	0.872
Multiscale 1x1 filter, 3-7 kernel size, 32 neurons	0.721	0.780

TABLE 4.3: Performances comparison of ablated SparcoNet structure on FF+ dataset

4.2.4 Data Preprocessing

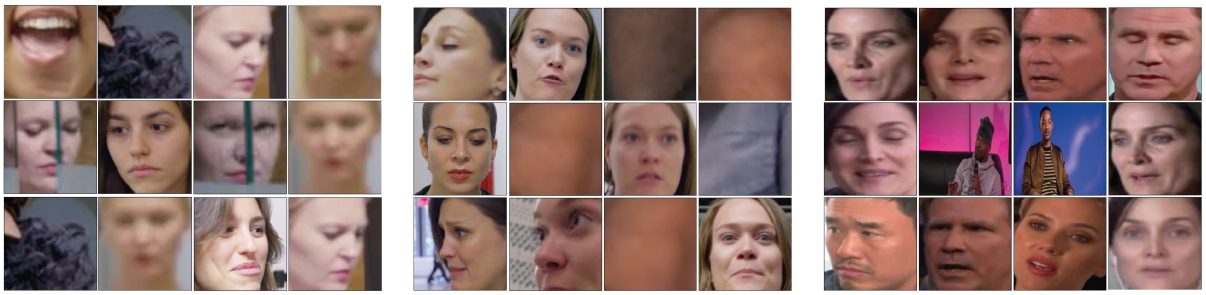
Most deepfake techniques target the human face region, but this area can be influenced by physical and environmental factors, such as head angle and distance between the target and the camera. To improve the model's detection performance, the input data undergoes pre-processing for normalization. In our case, face detection and extraction are vital operations in pre-processing, as the detection model mainly focuses on learning features between the deepfake and real face region. Therefore, we evaluated the three popular facial detectors: *i.* Dlib frontal face detector [224], *ii.* Cascade classifier [225], and *iii.* MTCNN[226]. We used these detectors to detect and extract the face region of the input data and resize it to a standard 256x256x3 before passing it to the detection model.

During our study, we observed a minor finding that the bias of different face detectors has a significant impact on the detection model's performance. Table 4.4 shows that the validation loss of MTCNN is 40 times lower, and its validation accuracy is 1.0% higher than the other two detectors. Although the accuracy difference is only 1.0%, the 40 times lower loss could have a more substantial impact on larger datasets, resulting in a higher error rate and making the detection model more vulnerable. Additionally, we discovered that using different face detectors for pre-processing training and testing datasets leads to a drop in detection accuracy of up to 10%. The biases of the different detectors are evident in Figure 4.5. The Dlib frontal face detector captures certain areas, such as the mouth or hair, the Cascade classifier occasionally returns a skin-like portion,

and the MTCNN detector can fail to extract the face and retain the original image. It is noteworthy that we selected MTCNN as the primary face detector for our experiment.

Face Detector	AUC	Val. Accuracy	Val. Loss
Dlib	0.996	0.986	0.0418
Cascade Classifier	0.999	0.988	0.0475
MTCNN	1.0	1.0	0.0059

TABLE 4.4: Detection performances of each face detector for Face2Face dataset



(a) (b) (c)

FIGURE 4.5: Sample inaccurate detection (where real images are detected as deepfake) for the different face detectors to demonstrate bias caused by incorrect face region, (a) Dlib (b) Cascade Classifier (c) MTCNN.

4.3 Experimental Setup

We trained SparcoNet using the TensorFlow framework and Python version 3.8 with a GeForce RTX 2080 GPU. To maximize computer utilization, we implemented 100 epochs with a batch size of 128. The configuration of the hyperparameters is as follows: Adam optimizer, a learning rate of $1e$-03 with a decay rate of 0.5, and patience of 10 epochs, $1e$-01 L2 kernel regularization, ReLU activation, and a Sigmoid classifier. We performed early stopping with patience of 12 epochs to prevent overfitting.

4.3.1 Datasets

In this experiment, we will focus on examining the performance of the proposed SparcoNet on six primary datasets, including the four major datasets (Face2Face, Deepfake, FaceSwap, Neural Textures) of FaceForensics++ [23], Deepfake Detection Dataset (DDD) [24], Celeb-DF [25], Deeper-Forensics-1.0 [28], the deepfake dataset created by Mesonet's authors (MesoDF) [27], and the complicated Deepfake Detection Challenger

(DFDC) dataset [26]. To be precise, the total number of datasets will be nine datasets if we assume the four datasets from FaceForensics++ as individual datasets. We implemented the dataset following the ratio mentioned in FaceForensics++ [23], which is a 7.2:1.4:1.4 split ratio for training, validation, and testing sets. We pre-processed the datasets using the MTCNN face detector to extract facial region and sliced the video into frames with a consistent 20 frames per video at a 0.5 frame rate.

4.3.2 Evaluation Metric

We evaluated the overall model classification using the AUC (Area under the ROC Curve, where ROC stands for Receiver Operating Characteristic curve) and Accuracy (Acc), both of which were measured at the frame level. We have also applied recall, F1-score, and precision to support a more thorough analysis. Additionally, we introduced G_i, a generalization gap index, to measure the deviation of model accuracy across different datasets. The idea of this metric is to analyze the model's generalization capability across different datasets. Given T_x is the final classification score of training set, and T'_x is the final classification score of each testing set, T'_{x1}, T'_{x2}, T'_{x3}...T'_{xn}. G_i presents the generalization gap between the training and testing datasets. Equation 4.1 and 4.2 show the formula of the computation.

$$T'_{avg} = \frac{\sum_{i=1}^{n} T'_{xi}}{n} \tag{4.1}$$

$$G_i = T'_{avg} - T_x \tag{4.2}$$

4.3.3 Experiments & Results

Comparison to popular state-of-the-art benchmark datasets. We first examined the model's ability using popular state-of-the-art benchmark datasets, including Faceforensics++ [23], CelebDF [25], MesoDF [27], DF1.0 [28], DDD [24], and DFDC [26]. The results, as shown in Table 4.5, demonstrate that SparcoNet achieves outstanding performance, with an average accuracy of 98.1% and 0.985 AUC across these datasets. The F1-Score, Recall, and Precision are all maintained above 90% for all datasets. Notably, our experiments reveal that the model's performance remains quite stable across various datasets, with less than 4% accuracy deviation, even with the complex DFDC

dataset. This observation suggests that our model has the capacity to handle datasets with different configurations and perturbation techniques effectively. Figure 4.6 presents the AUC graph of Sparconet evaluated with each dataset.

Dataset	F1	Recall	Precision	Acc	AUC
FF++ [23]	0.981	0.982	0.982	0.986	**0.996**
Celebdf [25]	0.978	0.946	0.989	0.980	**0.999**
DF1.0 [28]	0.997	0.999	0.996	0.999	**1.000**
DDD [24]	0.962	0.961	0.980	0.965	**0.991**
DFDC [26]	0.955	0.939	0.974	0.960	**0.921**
Mesodf [27]	0.994	0.990	0.997	0.998	**1.000**

TABLE 4.5: SparcoNet performance evaluation on state-of-the-art datasets

Comparison to state-of-the-art models. We have also compared our performances with state-of-the-art models [23, 27, 161, 164, 166, 174, 196, 227, 228] on raw, c23, and c40 compression factor rate of the F2F dataset. As shown in Table 4.6, the best performance of each compression factor rate is presented in bold. Although there are models that surpass the accuracy of SparcoNet for certain compression rate, for example, Zhang *et al.* [161] works better for raw data and Luo *et al.* [164] works better for c23 data, but their performances is not consistent across all the compression factor. Whereas, SparcoNet performs more consistently on different compression rate data. Despite utilizing lower computational resources, our model achieved a higher average accuracy than all stateof-the-art models. This finding is particularly significant, as our model focuses more on spatial feature learning at a mesoscopic level using the separable convolution layers with a consistent kernel size and high receptive field. It learns to capture the general spatial discriminative features regardless of the data compression rate. This result highlights the potential practical benefits of our model over other state-of-the-art alternatives, particularly when computational resources are limited or restricted. Figure 4.7 shows the example of raw, c23, and c40 compression factor rate data. We can see the image has a greater distortion with the increment of compression rate.

4.3.3.1 Cross-dataset Evaluation

We computed cross-dataset evaluation to measure model generalizability. The evaluation can be categorized into two main groups: *i.* Intra-dataset evaluation, which involves the

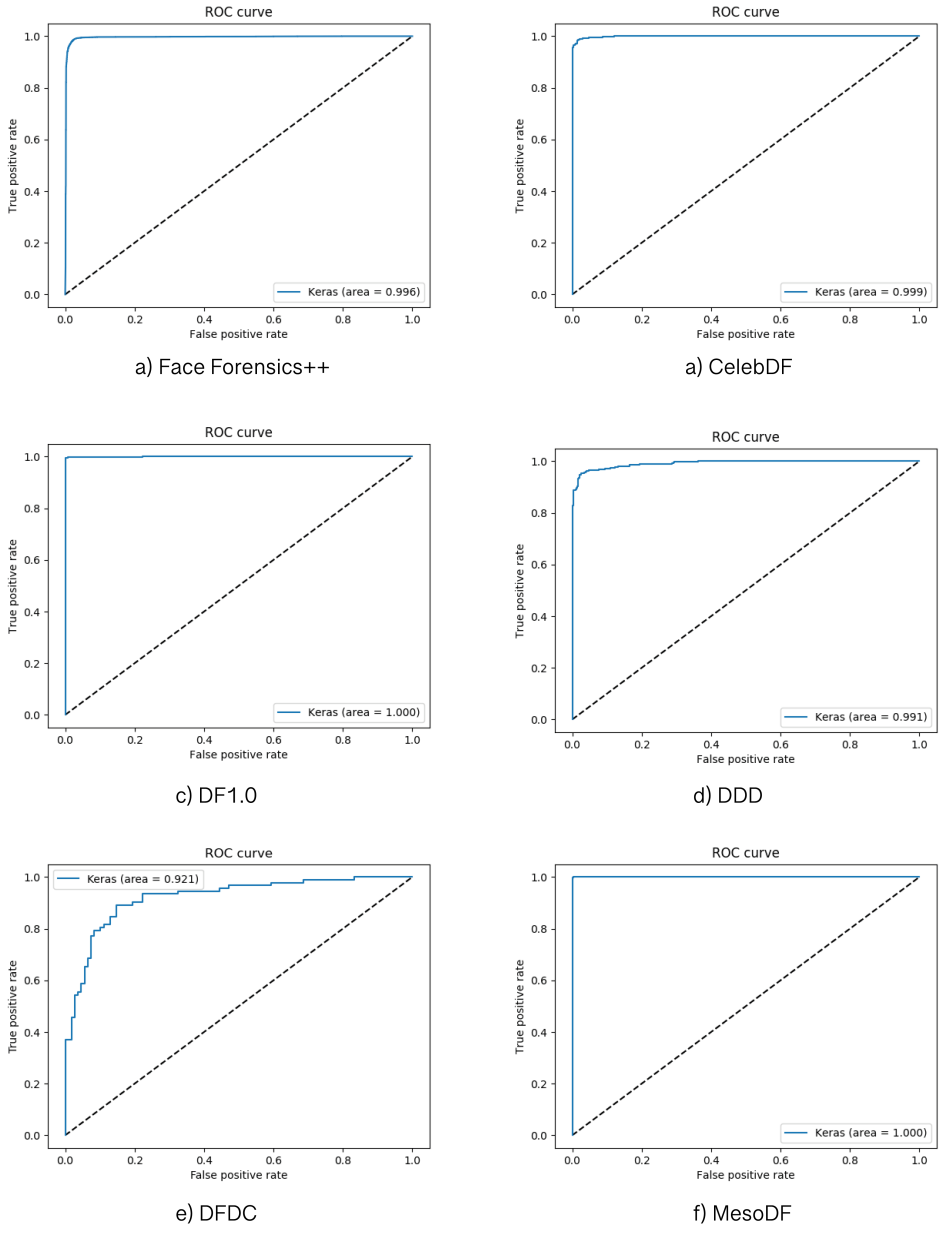


FIGURE 4.6: The AUC Graph for SparcoNet performance evaluation on state-of-the-art datasets

FIGURE 4.7: Example of Raw, C23, and C40 data

Model	Accuracy (Acc)			Avg
	Raw	C23	C40	
Cozzolino *et al.* [227]	0.996	0.798	0.558	0.784
Meso-4 [27]	0.946	0.924	0.832	0.900
MesoInception-4 [27]	0.968	0.934	0.813	0.905
Capsule-Forensics [174]	0.994	0.965	0.810	0.923
Xception [23]	0.993	0.957	0.810	0.920
Luo *et al.* [164]	-	**0.992**	0.957	0.975
Zhang *et al.* [161]	**0.999**	0.980	0.918	0.966
DeepfakeHop [228]	-	0.960	0.930	0.945
SPSL [166]	-	0.860	-	0.860
Chen *et al.* [196]	-	0.960	-	0.960
Ours	0.992	0.986	**0.982**	**0.987**

TABLE 4.6: Comparison with state-of-the-art models on Raw, C23, and C40 compression factors of FF++ F2F dataset

training and testing the dataset generated from same source of data, and *ii.* Inter-dataset evaluation, , which involves the training and testing the dataset generated from different sources of data.

Intra-dataset Evaluation. We applied DF and F2F from the FF++ dataset to evaluate intra-datasets evaluation with Luo *et al.* [164], Xception [23], Face X-ray [171], and MesoInception-4 [27], where the first three models have been widely applied as baseline models for cross-dataset evaluation. The DF and F2F are datasets generated using different manipulation techniques from the same source data. Table 4.7 shows that our intra-datasets generalization performance outperforms that of MesoInception-4 [27] and Xception [23], and Face X-ray [171] and our G_i is close to Luo *et al.* [164] although our model size is only 1.7% of theirs as they implemented Xception network as their backbone. Upon reviewing the table of results, we can observe that both our model and Luo's model demonstrate superior performance on the F2F dataset as compared to the DF dataset. This observation is attributed to the fact that the discriminative features from the DF dataset have less general impact than those from the F2F dataset. This is because the F2F discriminative feature is more likely to detect unusual artifact features on any facial landmarks, whereas the DF dataset emphasizes facial features thoroughly. Given that SparcoNet places a greater emphasis on preserving spatial relationships between features, while Luo *et al.* focus on exploring residual features with an attention

mechanism, it is reasonable to conclude that our model performed better in the cross-evaluation of the F2F dataset, where spatial features play a crucial role in identifying subtle traces between landmarks. On the other hand, Luo *et al.* achieved better results in the cross-evaluation of the DF dataset, where residual features contribute more to the thorough detection of facial artifact traces.

Model	Training	Testing (AUC)		G_i (%)
		DF	F2F	
Luo *et al.* [164]	DF	0.992	0.764	**25.97**
MesoIn-4 [27]		0.917	0.657	33.04
Xception [23]		0.963	0.684	28.97
Face X-ray [171]		0.987	0.633	35.87
Ours		0.993	0.729	**26.49**
Luo *et al.* [164]	F2F	0.837	0.994	**17.14**
MesoIn-4 [27]		0.640	0.934	37.36
Xception [23]		0.803	0.994	21.25
Face X-ray [171]		0.630	0.984	35.98
Ours		0.847	0.998	**15.13**

TABLE 4.7: Intra-datasets Evaluation with state-of-the-art baseline models on C23 compression factor between FF++ F2F & DF datasets

Inter-dataset Evaluation. Evaluating across datasets is significantly more complex than within a single dataset due to the presence of different perturbations and environmental factors. As shown in Table 4.8, larger models such as Luo *et al.* [164], Xception [23], and Face X-ray [171] have achieved superior performance compared to shallower models such as Mesonet [27] and our proposed model. This difference is particularly significant when comparing results on complex datasets such as DDD and DFDC dataset. This statement is because these datasets employ videos featuring actors who are not necessarily facing the camera and standing at a distance, leading to lower-quality extracted facial regions. Consequently, a shallower network may not have sufficient capacity to handle such scenarios. It is important to note that although our overall results are not as favorable as those of larger models, our results on the CelebDF and DF1.0 datasets are comparable and even surpass those of the Xception [23] and Face X-ray [171] models. Furthermore, we have achieved an overall 7.40% lower G_i than the benchmark Mesonet, which is a noteworthy accomplishment.

Compression rate factors. Additionally, we conducted a cross-dataset evaluation of

Models	Testing				G_i (%)
	CelebDF	DDD	DF1.0	DFDC	
Luo *et al.* [164]	0.794	0.919	0.738	0.673	21.10
MesoIn-4 [27]	0.477	0.586	0.553	0.414	40.70
Xception [23]	0.594	0.831	0.698	0.679	26.30
Face X-ray [171]	0.660	0.856	0.723	0.625	27.10
Ours	0.612	0.602	0.728	0.556	33.30

TABLE 4.8: Inter-dataset evaluation Comparison with state-of-the-art baseline models trained with FF++ C23 dataset

the model's performance, specifically investigating its ability to handle different compression rate factors (CRF) using the F2F dataset. The dataset included the dataset with varying levels of compression rate factor, ranging from raw to C23 and C40. Our findings, as presented in Table 4.9, indicate that the generalization gap index is inversely proportional to the degree of compression rate. Additionally, cross-evaluation performance proves to be more stable with the C40 CRF. This suggests that the model is more likely to perform consistently across data with different resolutions when trained with the hardest compression rate. Interestingly, we observed that the model trained on highly compressed data demonstrated a greater ability to learn more detailed and meaningful features, ultimately enabling it to cope better with datasets featuring different levels of compression.

CRF		Test (AUC)			G_i (%)
		Raw	C23	C40	
Train	Raw	0.998	0.920	0.683	13.1
	C23	0.997	0.998	0.850	5.27
	C40	0.951	0.952	0.956	**0.31**

TABLE 4.9: SparcoNet robustness evaluation against Raw, C23, and C40 compression factor of F2F dataset

4.4 Visualization

Visualization is a powerful technique that enables us to gain insight into how a neural network model makes its final prediction decisions. In order to better understand the behavior of our SparcoNet model, we have generated both feature maps and gradient maps for further analysis.

4.4.1 Feature Map

Feature maps provide a powerful visualization tool that allows us to examine the activation patterns of individual neurons within a neural network in response to specific input stimuli. By analyzing these activation patterns across different network layers, we can gain valuable insights into the types of features that the model is learning and how it processes input data. In the context of deepfake detection, we have observed that the facial alignment of the subject is typically invariant, while the generation of the face organ shape in deepfakes is becoming increasingly realistic. As a result, we have found that focusing on global feature learning is more effective than capturing low-level features. We aim to train the model to place greater emphasis on global feature learning so that it can more effectively capture features as a whole rather than relying solely on certain local features.

Figures 4.8 and 4.9 provide detailed visualizations of the feature maps generated by the four major blocks of Mesonet and SparcoNet, respectively. However, due to the larger receptive field of SparcoNet, the feature map representations can sometimes be too small to observe in detail. To address this issue, we have selected 16 filters from each major block of SparcoNet and presented them in Figure 4.10. These filters were carefully chosen to highlight key features and provide a more detailed view of the activation patterns within the network.

By carefully examining these feature maps and selecting the most relevant filters, we can gain valuable insights into the types of features that the network is learning at different layers. Figure 4.8 shows that Mesonet tends to capture more local features at the earlier layers, such as the face edges shown in Block 1. However, as the network processes downsampling after each block, the resulting smaller spatial dimensions, higher activation thresholds, and more complex features in the subsequent blocks can sometimes make the feature maps too small to be effectively visualized.

On the other hand, Figure 4.10 provides a different picture of the feature maps generated by SparcoNet. Despite also being a shallower deep neural network like Mesonet, SparcoNet places a greater emphasis on capturing global features in the first two blocks. Even after dimension reduction, the feature maps are still able to effectively present a visualization of some complex global features.

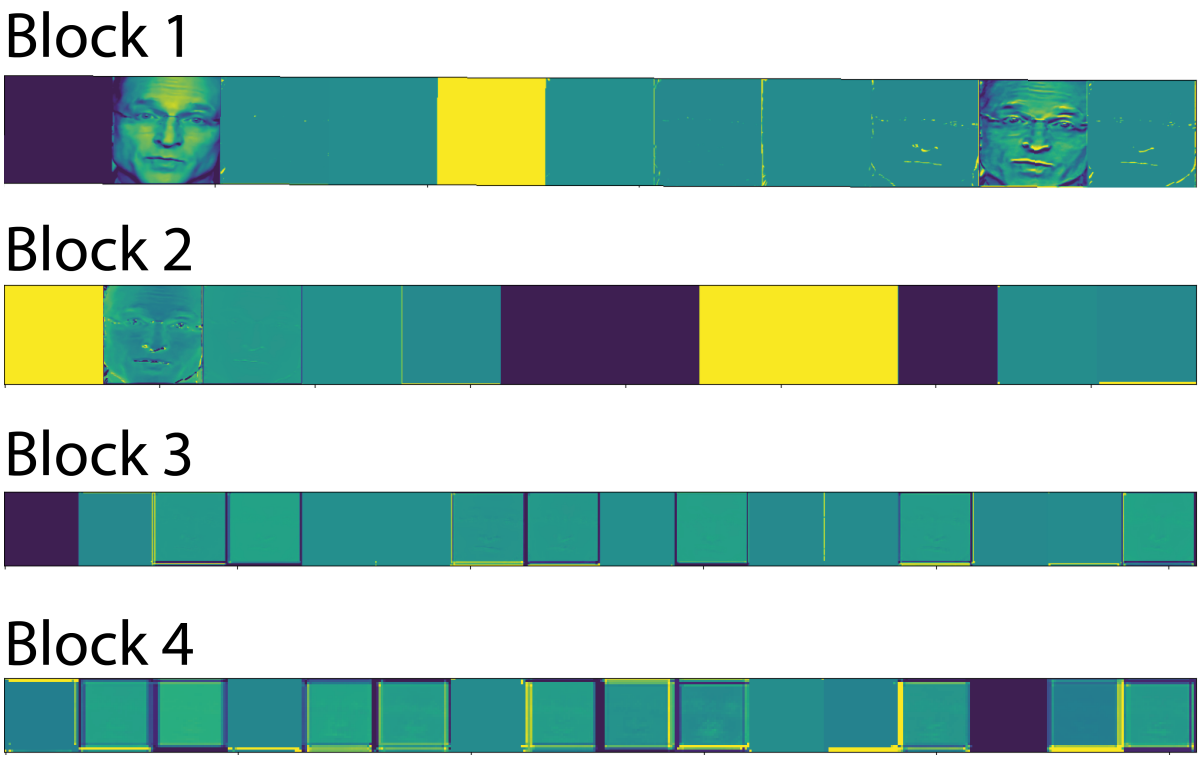


FIGURE 4.8: The feature map of the four major blocks of Mesonet

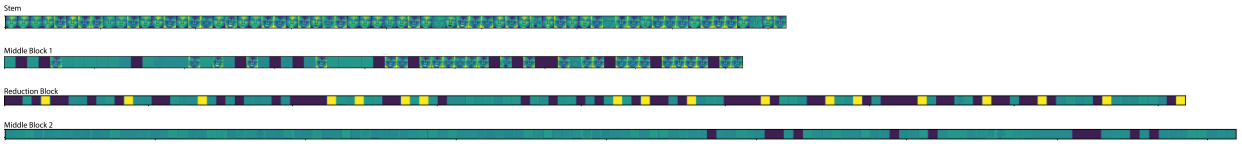


FIGURE 4.9: The feature map of the four major blocks of SparcoNet

4.4.2 Grad-CAM

Gradient maps allow us to visualize how changes in the input data affect the model's output. Specifically, gradient maps show the magnitude and direction of the gradient of the model's output concerning the input data. By analyzing these maps, we can identify which parts of the input data are most important for the model's decision-making process and how changes to these input features affect the final prediction. Figure 4.11 depicts the gradient map for real, FFF2F, and FFDF data. The figure reveals that deepfake images have a more centralized confidence level in specific regions, such as the eye and mouth regions for FFF2F and the forehead, chin, and jaw regions for FFDF. By evaluating the gradient maps, we can see that as SparcoNet leverages both local and global features with focus on spatial relationship preservation instead of focusing on specific local facial landmarks, the temperature intensity difference throughout the whole image is more consistent in real images compared to fake images.

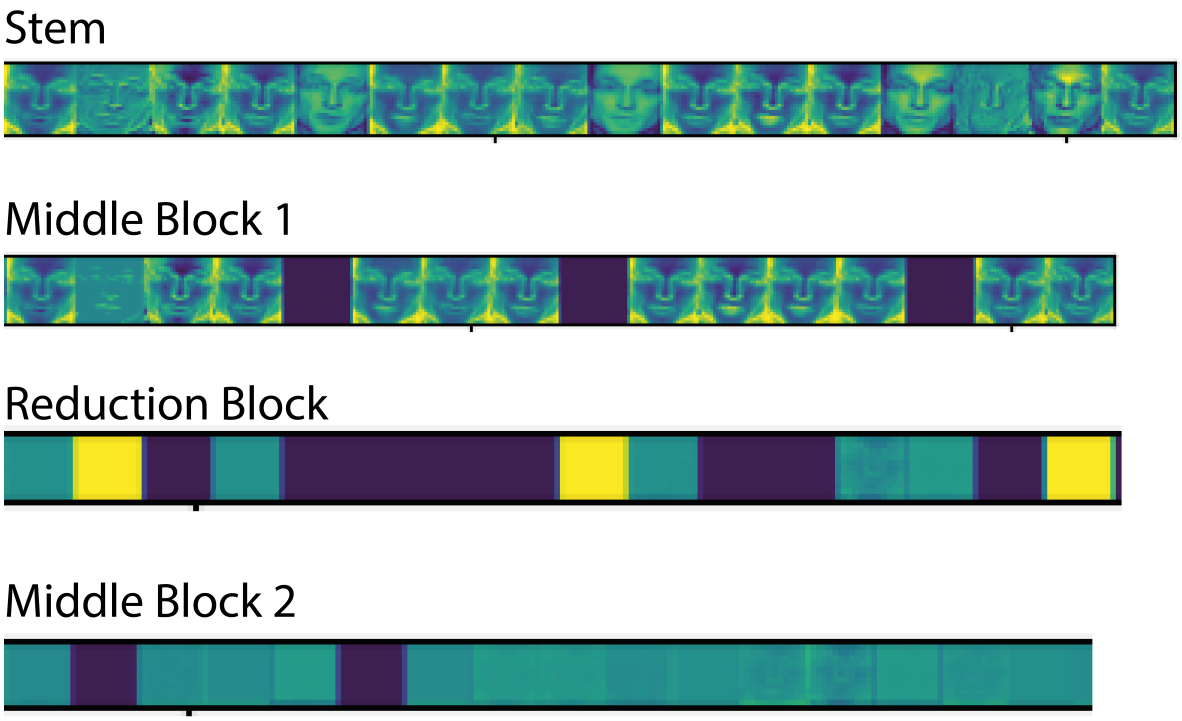


FIGURE 4.10: The feature map with 16 filters of the four major blocks of SparcoNet

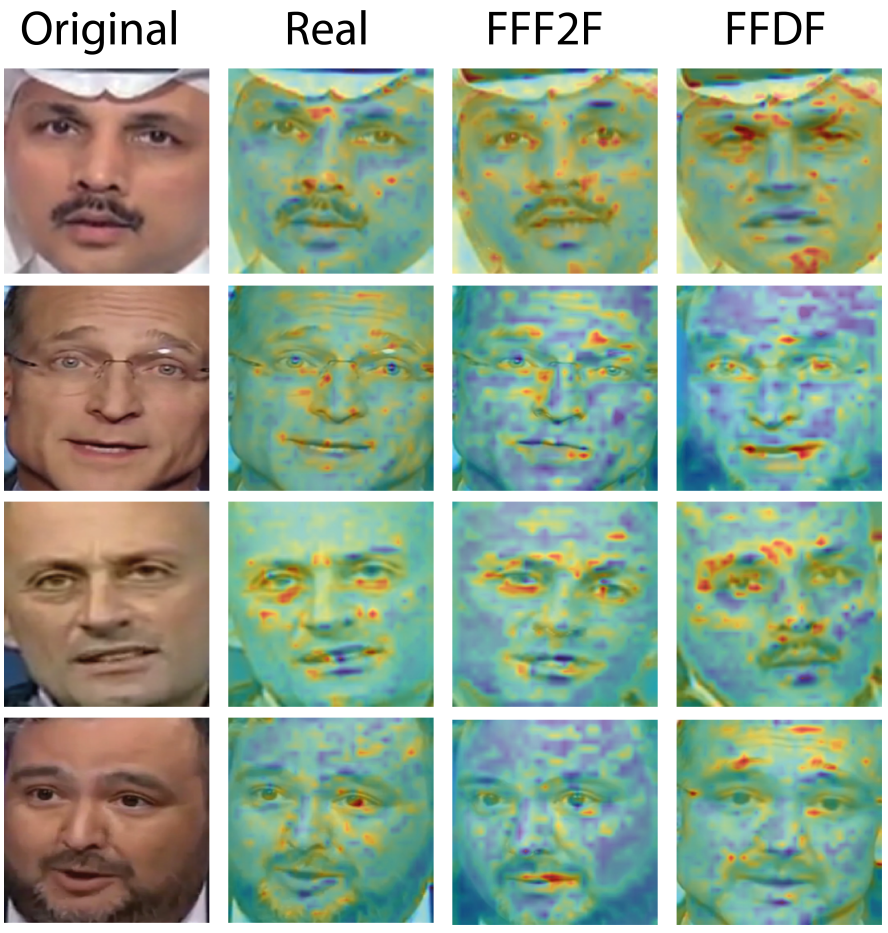


FIGURE 4.11: The gradient map comparison between real, FFF2F, and FFDF output

4.5 Summary

We have developed a shallower and spatially cost-efficient network (SparcoNet) that exploited the concept of Mesonet's mesoscopic feature learning and further enhanced its robustness and feature learning by introducing a new network structure architecture. Our findings indicate that the model can capture high-level features better while reducing computational costs by increasing the receptive field, emphasizing spatial feature preservation, and optimizing the network through residual connections. The experiment results show that our model outperforms most state-of-the-art models and achieves promising intra-dataset evaluation performance comparable to deeper networks. Specifically, SIRNet achieved an average AUC of 0.985 across six state-of-the-art datasets, with a similar intra-dataset performance as the deeper deepfake detector model, with less than a 1% difference in the generalization gap index (G_i) and at a lower computational cost. However, inter-dataset evaluation results show that SIRNet still has room for improvement. It is harder to capture global feature learning across different datasets due to distinctive spatial information resulting from varying perturbations in the testing datasets compared to the training dataset. Hence, our second and third contributions aim to improve the model's reliability against various perturbations and adversarial attacks.

Chapter 5

SparcoNet with Self-Supervised Learning (SSL-SparcoNet)

This chapter consists of part of the following publication:
Jia Wen Seow, Mei Kuan Lim, Raphaël C.W. Phan, & Joseph K. Liu (2023). SparcoNet with Block-Switched Self-Supervised Learning: An Effective Framework for Deepfake Detection with Improved Adversarial Defense. Submitted to Information Sciences Journal. [**Submitted**]

5.1 Motivation

To enhance the reliability and generalization of the proposed model in dealing with unseen perturbations and white-box adversarial attacks, we have constructed SparcoNet using a contrastive approach for self-supervised learning. Neural networks (NNs) have been prone to issues where even small input perturbations can cause the trained model to deviate from its expected behavior. Therefore, ensuring that the proposed NN-based detection model can withstand malicious adversarial attacks is crucial. Two major types of adversarial attacks are known by their access level to the target model: white-box attacks and black-box attacks. White-box attacks have access to information about the

target model, including weight, hyperparameter, and model architecture. In contrast, black-box attacks have limited knowledge about the model and can only produce adversarial samples through query access [214]. This chapter will focus on white-box attacks, and we will discuss defense against black-box attacks in the following chapter. Figure 5.1 illustrates an example of an adversarial attack where the object detection model classified a panda as a gibbon by simply adding some Gaussian noise to the source image [216]. To enhance the generalization and reliability of the SparcoNet model against perturbations and unseen datasets, we have modified and adapted SimCLR self-supervised learning, which has shown outstanding performance in helping the object detection model deal with unseen datasets [193].

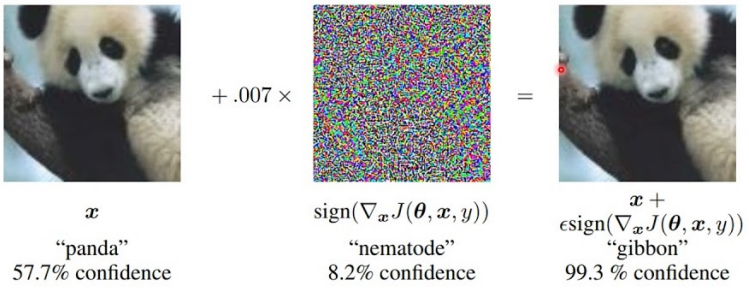


FIGURE 5.1: Example of Adversarial Attack

5.2 SSL SparcoNet Development

SimCLR self-supervised learning is a powerful technique that allows the model to learn generic feature representations of an unlabeled dataset in an unsupervised manner. This is done by maximizing the agreement between distinct augmented views of the same input image while minimizing the agreement between different input images [193].

The main idea behind contrastive learning is to learn a mapping function that maps similar inputs closer to each other in the feature space, while pushing dissimilar inputs further apart. In the context of SimCLR, this is achieved by training the model on pairs of augmented images, where the model must determine whether the two images are different views of the same image or two separate images.

Specifically, during training, the model takes two randomly augmented views of the same image and passes them through a shared encoder network. The encoder network

maps the two images to a high-dimensional feature space. Then, a contrastive loss is applied to the embeddings of these images, where the goal is to maximize the similarity between embeddings of similar images and minimize the similarity between embeddings of dissimilar images.

This approach allows the model to learn generic feature representations of the unlabeled dataset in a contrastive, unsupervised manner. These representations are then fine-tuned with a labeled dataset in the downstream classification task. During unsupervised learning, the model learns generic features by maximizing the agreement between distinct augmented views of the same input image and minimizing the agreement between different input images. This learning approach enables the features of related data to "attract" each other while features of unrelated data "repel". Figure 5.2 illustrates the concept of contrastive learning, where the model learns to map similar inputs closer together in the feature space while pushing dissimilar inputs further apart.

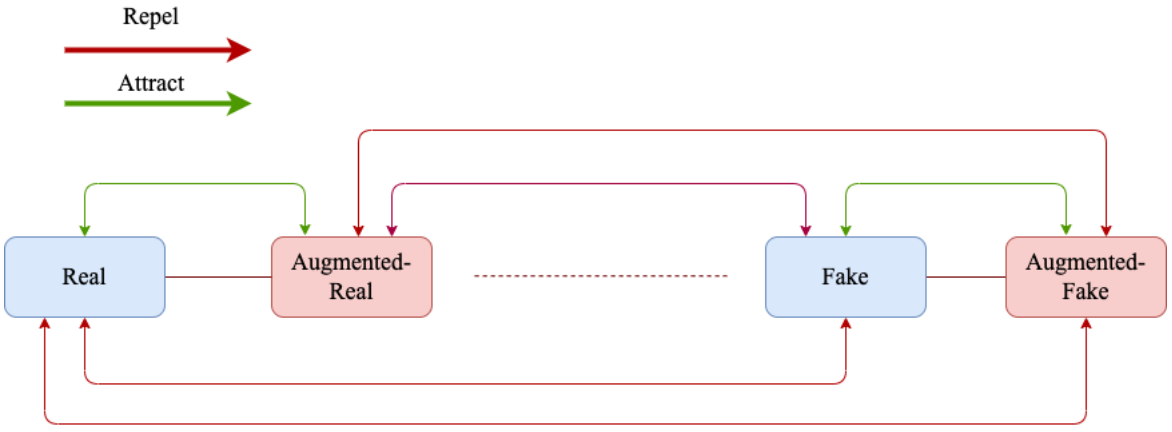


FIGURE 5.2: The concept of Self-Supervised SparcoNet Model

Drawing inspiration from the success of the SimCLR self-supervised learning framework in enhancing the generalization capability of object detection models under different transformations, the present thesis proposes a novel approach, SSL-SparcoNet, which adapts the training methodology of SimCLR SSL to our specific use case.

Figure 5.3 shows the architecture of the SparcoNet model with SimCLR self-supervised learning, where the unsupervised pre-training step is followed by supervised downstream fine-tuning using labeled data. In contrast to the conventional SimCLR SSL approach, our proposed methodology employs a more balanced data-splitting ratio for unsupervised and supervised downstream tasks. This is motivated by the distinctive nature of deepfake data, which consists solely of the human facial region instead of object data that features

diverse subjects. While the model can effectively learn and differentiate between various subjects in object detection, the same may not hold true for deepfake subjects. Thus, the model requires adequate data for downstream task learning. By integrating the features learned from unsupervised learning on both the original and augmented data, SparcoNet could enhance its generalization capability against datasets with different perturbations. The training pipeline is improved by enhancing the preprocessing of the input data, denoted as Input X in the figure, which generates a pair of data comprising the original input and its augmented view. Subsequently, this pair of unlabeled data is fed to the contrastive SparcoNet for unsupervised learning. The embedding Hi and Zi represent the output representation from the backbone and projection head, respectively, and the parameters are updated concurrently to acquire generic feature representations between the pair data. A normalized temperature-scaled contrastive loss will be computed by passing the embedding Zi from both pair data to the contrastive loss function. The downstream task is then performed to fine-tune the model with labeled data for the final deepfake classification.

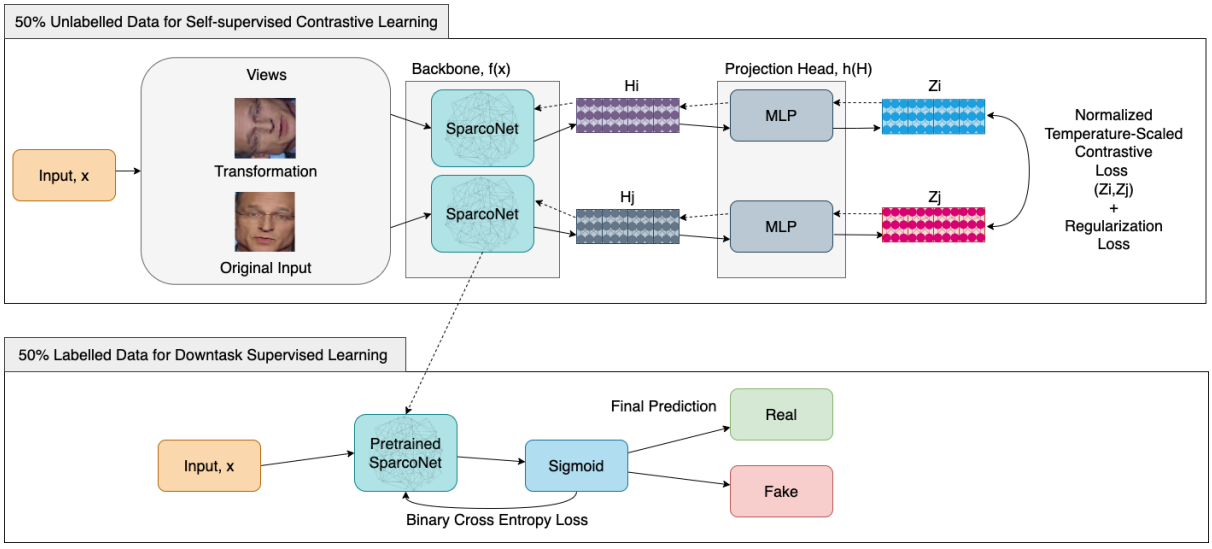


FIGURE 5.3: The architecture of Self-Supervised SparcoNet Model.

5.2.1 Model Enhancement

For SSL training, we implemented a projection head formed by three dense layers with 128, 32, and 4 neurons, respectively, where an L2 kernel regularization of $1e$-01 was applied to the last dense layer. A dropout of 0.5, batch normalization, and ReLU activation were applied after each dense layer. The hyperparameter settings were chosen through

a combination of empirical experimentation and careful consideration. The network was then connected to a sigmoid classifier. The equation 5.1 outlines the formula used in computing the normalized cross entropy loss between $\mathcal{X}$-λ during the SSL training. Different from object detection, the model tends to overfit easier due to data nature. Hence, the loss function is scaling down by multiplying a 0.5 factor to simplify the computation of the gradient of the loss function and make the loss scale more consistent with other metrics or objectives in the model. It also helps to balance the influence with other regularization terms to reduce overfitting.

$$\frac{\exp(sim(z_{i,1} \cdot z_{i,2,j})/\tau)}{\sum_{k=1}^{2B} 1(k \neq j) \exp(sim(z_{i,1} \cdot z_{i,k})/\tau)} \cdot 0.5 + \alpha \tag{5.1}$$

where $1(k \neq j)$ is the indicator function evaluating to 1 if $k! = j$, and j indexes over the batch. We reduced the overall loss by half and added an extra margin, α, to improve feature confidence.

5.2.2 Data Preprocessing

Given the superior cross-dataset evaluation performance of SparcoNet trained with the Face Forensics++ Face2Face dataset observed in the previous section, we opted to employ this dataset in our SSL experiments.

Following SimCLR's theory [193], we transformed the original dataset, which contained both real and fake data, by applying different augmentations and forming a tuple of data. In our experiment, one view of the tuple comprised the original data, while the other view comprised the augmented data. We used several augmentations in the experiments, including makeup, Gaussian blur, Sobel, color distortion, Gaussian noise, occlusion, and rotation.

We employed the makeup methodology from [229, 230] by applying the makeup with a medium intensity that covered eyeliner, eyeshadow, blush, and lipstick areas. For Gaussian blur, Gaussian noise, Sobel, color distortion, occlusion, and rotation, we applied the filter using the Python library. Gaussian blur is a type of image-blurring technique that uses a Gaussian function to calculate the intensity values of each pixel in the blurred image. This technique applies a weighted average of neighboring pixels to the central pixel, with the weights determined by the Gaussian function. The output result is a

smoothed image that reduces noise and sharp edges while preserving overall image features. Gaussian noise is a random noise that follows a Gaussian distribution and is often added to digital images to simulate various types of real-world noise, such as electronic interference fingerprints or film grain. The noise intensity is determined by the mean and standard deviation of the distribution, with higher standard deviations producing more intense noise. The Sobel filter is a spatial filter that is used for edge detection in image processing. It is a discrete differentiation operator that computes an approximation of the gradient of the image intensity at each pixel. It is typically implemented as a 3x3 convolution kernel that convolves with the image to compute the horizontal and vertical derivatives of the image. The resulting gradient magnitude and direction can be used to detect edges in the image. For color distortion, we implemented the color jitter technique. We have applied a variety of distortions, including hue, saturation, brightness, and contrast adjustments. Each distortion is applied with a random magnitude within a specified range to create a new, slightly altered image. Regarding occlusion, a black square mask was generated to randomly cover a specific image area, such as the eyes, cheeks, or mouth regions. As for rotation, a 90-degree rotation was applied to the original image.

Examples of these augmentations are shown in Figure 5.4, while Figure 5.5 displays examples of the tuple data for occlusion transformation. We selected these augmentations because they are often exploited as attacks or perturbations to evade deepfake detectors by disrupting their intrinsic feature spaces [217, 231, 232]. It is worth mentioning that, unlike the original implementation of SimCLR SSL for image classification, the object of deepfake data is more invariant to human facial content. Hence, retaining the original input of deepfake data for model training is crucial to avoid confusion during feature learning.

5.3 Experimental Setup

We used the same 8:1:1 data split ratio among the training, validation, and testing datasets. Still, unlike [193], which used an 8:2 splitting ratio for unsupervised and supervised data training, we used a 5:5 equal ratio. This application was made because the deepfake datasets have relatively similar feature spaces between classes compared to those in the object detection dataset, making the model easier to converge and stop

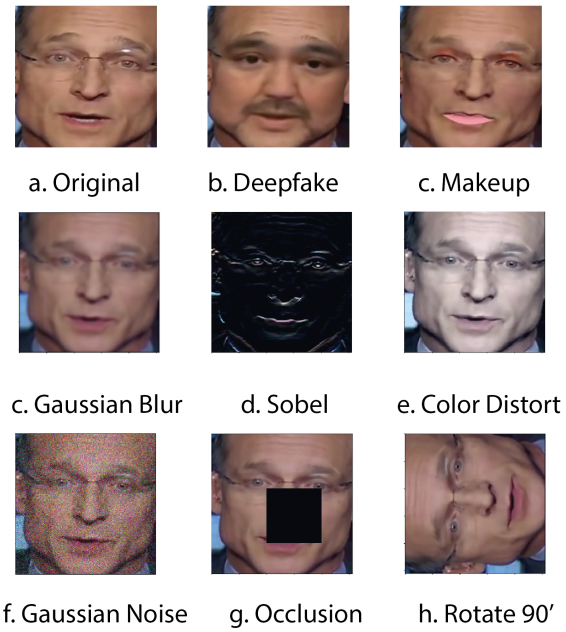


FIGURE 5.4: Examples of self-supervised augmented training data that applied to the original fake and real data

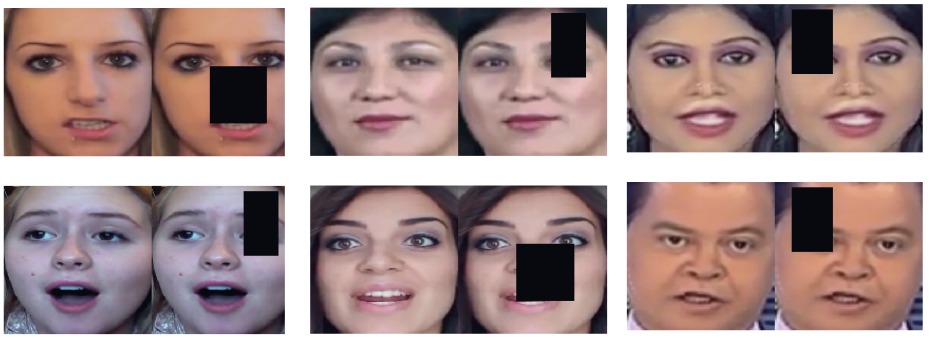


FIGURE 5.5: Examples of the tuple data with random occlusion transformation

learning if the dataset is too small in downstream supervised learning. To investigate the model's robustness against adversarial attacks, we examine the attack success rate (ASR) of the white-box adversarial attack for tuple data. The transformations were applied to emphasize the consistency of generic global feature learning of the data under different data perturbations or attacks.

There are many types of white-box adversarial attacks [233]. Among them, Fast Gradient Sign Method (FGSM)[234], which is widely adopted as a fundamental baseline due to its efficiency and practicality in generating adversarial samples [220]. Given the

consideration of training time and computational constraints in this thesis, we have performed adversarial attacks with the Fast Gradient Sign Method for our experiments. FSGM is a white-box adversarial attack, which computes the signed gradient to create a new image by backpropagating the pre-trained model. Using the signed gradient, FGSM generates an output image that, according to the human eye, looks identical to the original but tends to maximize loss and confuse most neural networks into making incorrect prediction. The epsilon value in Table 5.3 controls the size of adversarial examples; the higher the number, the stronger the attack on the target data. We also conducted both intra and inter cross-dataset experiments to examine the SSL-SparcoNet's reliability on unseen datasets. The results are presented in Tables 5.1 and 5.2.

5.3.1 Evaluation Metric

We employed similar metrics used in the previous section and included an additional Attack Success Rate (ASR) metric to evaluate the model behavior against the white box adversarial attack. The equation used to generate FGSM attack samples is presented below in Equation 5.2, where X is the original input image, Xadv is the adversarial output, C is the loss function, and Y$true$ is the true label.

$$\mathrm{X}adv = X + \epsilon \cdot sign(\nabla x C(X, \mathrm{Y}_{true})) \tag{5.2}$$

The Attack Success Rate is computed as shown in Equation 5.3, where ASR is the attack success rate, Pba is the model's performance before the attack, and Paa is the model's performance after the attack. The model performance metric used to compute ASR was AUC.

$$\mathrm{ASR} = (\frac{\mathrm{P}ba - \mathrm{P}aa}{\mathrm{P}ba}) \cdot 100\% \tag{5.3}$$

5.3.2 Experiment & Results

From Table 5.1, we can see that SSL contrastive learning with one augmentation significantly helps to reduce the generalization gap among the other three datasets, especially the augmentation with rotation. This result shows that the spatial relationship contributes more to global feature learning across datasets. By understanding the data from the different orientations, the model tends to capture more invariant deepfake

features regardless of the data source. At the same time, augmentation with color distortion and makeup presented superior performances in DF1.0 dataset compared to the original SparcoNet without SSL augmentation. Both augmentations emphasize pixel color modification which shows that DF1.0 dataset is more sensitive to color texture; hence the self-supervised learning with color-sensitive augmentations help in lowering its generalization gap.

Based on the findings reported in the study by Tsipras *et al.* [235], it was suggested that there exists a trade-off between the standard accuracy and adversarial robustness of a model, which arises from the inherent differences in the feature representations learned by standard and robust models. In light of the results presented in Table 5.2, it can be observed that the SparcoNet SSL intra-datasets evaluation tends to degrade while the inter-datasets evaluation improves. This phenomenon can be attributed to learning feature representations from pair data, which fundamentally differ from those obtained from a single original image. By evaluating with the best SSL transformation result, SSL trained with R90, SparcoNet is able to improve the inter-datasets accuracy by 11.2%, more significantly than its degradation of 5.9% on intra-datasets performance. Thus, the performance achieved can be considered reasonable, as improving global feature learning comes at the cost of a lower focus on local feature learning. It is noteworthy that SparcoNet SSL-Rotation successfully preserves the generalization gaps with a difference of no more than 6% when compared to the original SparcoNet without SSL. The focus on spatial relationship learning provided the most promising performance in both intra and inter datasets evaluation.

Augmentations	Test (AUC)				G_i (%)
	CelebDF	DF1.0	DDD	DFDC	
SparcoNet w/t SSL	0.670	0.477	0.537	0.556	40.90
SSL-SparcoNet GB	0.564	0.597	0.572	0.587	35.70
SSL-SparcoNet GN	0.558	0.591	0.478	0.644	34.50
SSL-SparcoNet R90	0.563	0.602	0.666	0.624	29.70
SSL-SparcoNet Sobel	0.525	0.651	0.477	0.697	36.20
SSL-SparcoNet CD	0.533	0.806	0.544	0.633	33.90
SSL-SparcoNet RO	0.553	0.583	0.568	0.644	35.20
SSL-SparcoNet Makeup	0.518	0.704	0.556	0.566	35.10

TABLE 5.1: Comparison of inter cross-datasets evaluation (Trained with Face Forensics++ F2F) between the original SparcoNet with SSL-SparcoNet trained with 1 transformation on different augmentaions

Augmentations	Test (AUC)				G_i (%)
	Face Forensics ++				
	F2F	DF	FS	NT	
SparcoNet w/t SSL	0.969	0.839	0.670	0.794	20.8
SSL-SparcoNet GB	0.937	0.617	0.589	0.643	32.1
SSL-SparcoNet GN	0.913	0.582	0.626	0.667	31.2
SSL-SparcoNet R90	0.910	0.653	0.620	0.681	25.9
SSL-SparcoNet Sobel	0.950	0.620	0.509	0.672	36.8
SSL-SparcoNet CD	0.968	0.651	0.592	0.721	31.3
SSL-SparcoNet RO	0.939	0.622	0.547	0.616	34.4
SSL-SparcoNet Makeup	0.937	0.564	0.605	0.683	32

TABLE 5.2: Comparison of intra cross-datasets evaluation (Trained with Face Forensics++ F2F) between the original SparcoNet with SSL-SparcoNet trained with 1 transformation on different augmentaions

The results presented in Table 5.3 demonstrate the effect of increasing epsilon on the FGSM Attack Success Rate (ASR). A higher epsilon value indicates a greater attack generated on the target data. However, the table also highlights that not all SSL training strategies contribute equally to defending against adversarial attacks. Among the various transformations tested, only those trained using Gaussian Blur (GB) and Gaussian Noise (GN) yielded superior performances compared to SparcoNet without SSL. GB and GN have perturbation characteristics that are more proximate to those of white-box adversarial attacks. It is worth noting that the ASR is almost 100% for SparcoNet without SSL, while only approximately 13% for SparcoNet-GB when epsilon equals 0.10. Figure 5.6 shows that the ASR of SparcoNet without SSL grew significantly across the epsilon increment compared to SparcoNet SSL with GB, GN, and R90.

	FGSM Attack Success Rate, ASR (%)					
Augmentations	e=0.10	e=0.15	e=0.20	e=0.25	e=0.30	e=0.35
SparcoNet w/t SSL	98.89	100	100	100	100	100
Gaussian Blur (GB)	13.58	14.90	29.68	36.68	45.78	53.12
Gaussian Noise (GN)	13.00	20.48	27.71	36.47	44.58	51.37
Rotate 90 Degrees (R90)	15.16	26.04	36.48	45.93	54.51	62.09
Sobel	81.16	91.79	95.47	97.16	98.00	93.63
Color Distortion (CD)	83.47	95.66	98.24	99.07	99.59	99.79
Random Occlusion (RO)	62.50	71.57	91.16	94.87	96.70	97.55
Makeup	31.27	47.81	64.24	69.80	79.09	81.11

TABLE 5.3: Comparison of FGSM Attack Success Rate between the original SparcoNet with SSL-SparcoNet trained on different augmentaions

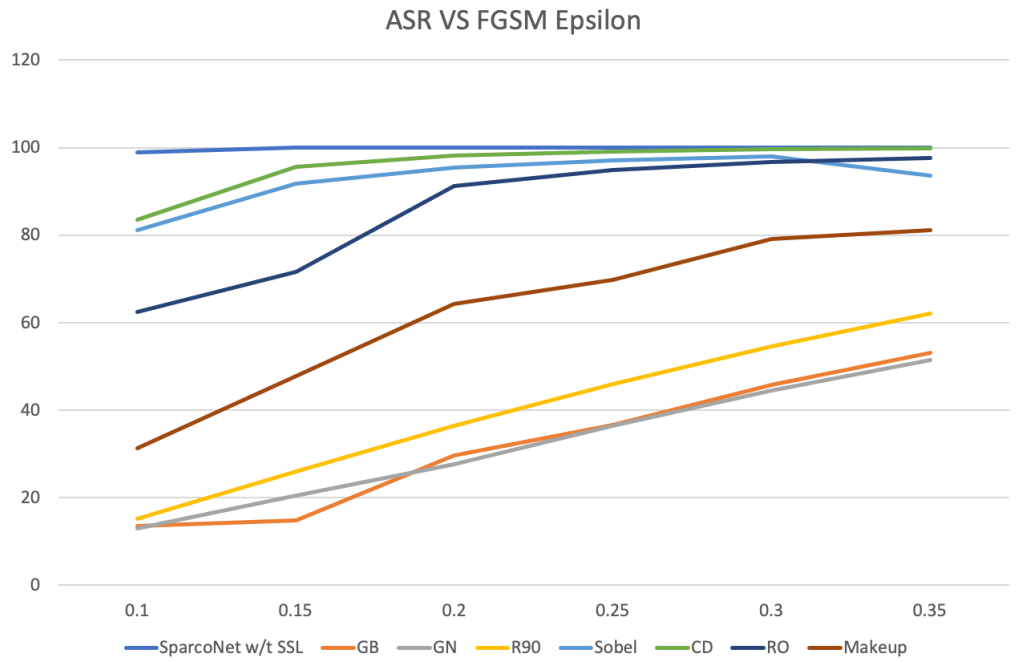


FIGURE 5.6: Attack Success Rate Against FGSM Epsilon

5.4 Summary

We have significantly improved the cross-dataset and white-box adversarial defense ability of SparcoNet by restructuring and training it in a self-supervised contrastive manner with various augmentations. The performances of SparcoNet SSL trained with diverse transformations demonstrate that the SSL training has effectively reduced the inter-dataset evaluation gap by up to 10% in G_i, while showing a slight degradation in the intra-dataset evaluation. The results also showed that SSL-SparcoNet could defend against adversarial attacks, reducing the attack success rate by up to 62.53% compared to the original SparcoNet without SSL. The following section will further enhance the model's ability to defend against black-box adversarial attacks.

Chapter 6

Block-Switching SSL SparcoNet (BS-SSL SparcoNet)

This chapter consists of part of the following publication:
Jia Wen Seow, Mei Kuan Lim, Raphaël C.W. Phan, & Joseph K. Liu (2023). SparcoNet with Block-Switched Self-Supervised Learning: An Effective Framework for Deepfake Detection with Improved Adversarial Defense. Submitted to Information Sciences Journal. [**Submitted**]

6.1 Motivation

While white-box adversarial attacks require knowledge of the model's architecture and hyperparameters, black-box adversarial attacks only rely on information about the model's input. Attackers can generate adversarial samples by creating a surrogate model or directly querying the input data. The former method is commonly referred to as attack transferability, which involves training an input dataset until it confuses the model under a white-box attack and then transferring the attack to the target model using active learning techniques [214].

The most crucial criterion in designing the defense mechanism is the model's flexibility to acquire the defense ability while simultaneously upholding performance accuracy and computational costs. Considering these concerns, we have incorporated the block-switching mechanism [215] into training the SSL-SparcoNet backbone model. The stochastic manner of the block-switching mechanism allows the model to maintain its computational cost. It offers flexibility for training and ease of implementation across various neural network structures, all without inducing significant performance degradation. Therefore, unlike other defense mechanisms that implement extra distillation training [236], or modify network structure [237], which will increase the overall computational cost, block-switching emerges as a more fitting approach within our context. By integrating this mechanism, we reduce the possibility of attackers predicting the model's weight and hyperparameters, which enhances the model's robustness against black-box attacks. It does not require any additional data or labeling efforts, making it a cost-effective solution for enhancing model security. The approach provides an effective defense strategy against adversarial attacks and strengthens the security of machine learning models.

6.2 BS-SSL SparcoNet Development

Figure 6.1 depicts the workflow of the block-switching mechanism for SSL-SparcoNet. Following a similar methodology as presented in [215], training a Block Switching model is a two-stage process. In the initial stage, several backbone SparcoNets are trained independently with random initialization in an unsupervised manner. While trained on the same data, these models exhibit similar classification accuracy and robustness, but their model parameters differ due to stochasticity in the training process and random initialization. In the subsequent stage, the pre-trained unsupervised models form parallel channels of block switching. During runtime, a random channel is selected to be active to process the input while all other channels remain inactive, resulting in a stochastic model that behaves differently at various times. The selected channel serves as the backbone model and is linked to the common projection head for further supervised downstream task learning.

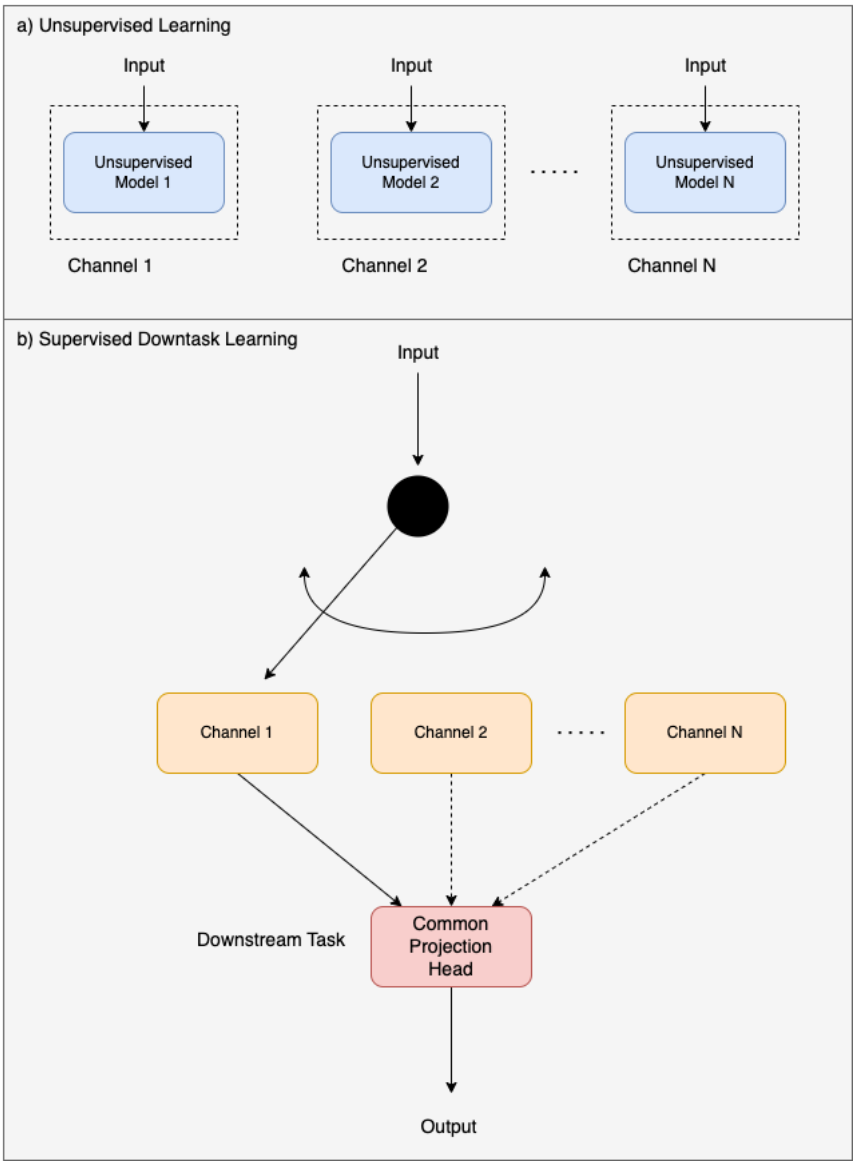


FIGURE 6.1: The workflow of Block Switching

6.2.1 Training Algorithm

By utilizing BS [215] for SSL-SparcoNet, we trained a λ channel of the encoder block to form a parallel encoder block. Each encoder block implements the same training hyper-parameter configuration but with different weight initialization. During the downstream task run time, only one encoder block will be randomly activated and connected to the projection head for further training. Therefore, the model can be less predictable by the attacker. Algorithm 1 outlines the details of implementation for BsSSL-SparcoNet. Given both labeled data $\mathcal{Y}$ and unlabeled data $\mathcal{X}$, the algorithm initially employs a chosen data transformation technique to produce a transformed version $\tilde{x}i, 2$ of each input

data $\tilde{x}i, 1$. Subsequently, the SparcoNet encoder block $f_\theta(\cdot)$ and a learned projection head $h_\phi(\cdot)$ are used to compute the original and transformed data representations $z_{i,1}$ and $z_{i,2}$, respectively. The normalized cross-entropy loss is then computed to reduce the distances between the representations of the original and transformed data. This training process is repeated λ times to enhance the diversity of learned representations. The obtained encoder blocks $f_\theta(\cdot)$ are then utilized to create a parallel encoder block β, which is employed for supervised fine-tuning on labeled data $\mathcal{Y}$. During this stage, a randomly chosen encoder block from β is linked to the projection head for further supervised learning using a binary cross-entropy loss function. This algorithm can learn meaningful representations from raw data without requiring labels and can be subsequently fine-tuned for specific supervised learning tasks.

Algorithm 1 Block-Switched Self-Supervised SparcoNet

Require: Unlabeled data $\mathcal{X}$, Labeled data $\mathcal{Y}$, block channel λ

Ensure: Learned representations $z_i = h_\phi(f_\theta(x_i))$, where $f_\theta(\cdot)$ is SparcoNet encoder block, normalized cross entropy loss $l_{zi,1,2}$, parallel encoder blocks β

1: **for** $x_i \in \mathcal{X}$ **do**
2: Sample an transformed version as $\tilde{x}_{i,2}$ of input data $\tilde{x}_{i,1}$ using the selected data transformation technique
3: Compute the original data representations $z_{i,1} = h_\phi(f_\theta(\tilde{x}_{i,1}))$ and the transformed data representations $z_{i,2} = h_\phi(f_\theta(\tilde{x}_{i,2}))$ with SparcoNet encoder block $f_\theta(\cdot)$
4: Compute the normalized cross entropy loss $l_{zi,1,2}$ using equation 5.1
5: **end for**
6: Repeat the training for SparcoNet encoder block $f_\theta(\cdot)$ for a λ block channel times
7: Discard the classifier for all SparcoNet encoder block $f_\theta(\cdot)$ and form a parallel encoder block β
8: **for** $y_i \in \mathcal{Y}$ **do**
9: Randomly activate an encoder $f_\theta(\cdot)$ from the parallel encoder block β and connect to the projection head for further supervised learning using a binary cross-entropy loss function
10: **end for**

6.2.2 Channel Selection

The plot depicted in Figure 6.3 showcases the attack success rate against FGSM at an epsilon value of 0.35 across various block-switching channel numbers. The results highlight a substantial decrease in the attack success rate from 1 to 3 channels, followed by a deceleration in the dropping rate upon reaching channel 4. Beyond channel 5, the defense mechanism exhibits a gradual saturation of its effectiveness. These findings suggest that the model's defense capability is significantly enhanced with a larger channel number.

Additionally, the overall accuracy loss demonstrated a 43% decline from channel 1 to channel 10. Such insights signify the relevance of considering block-switching channel numbers as a vital factor in bolstering model defenses against adversarial attacks.

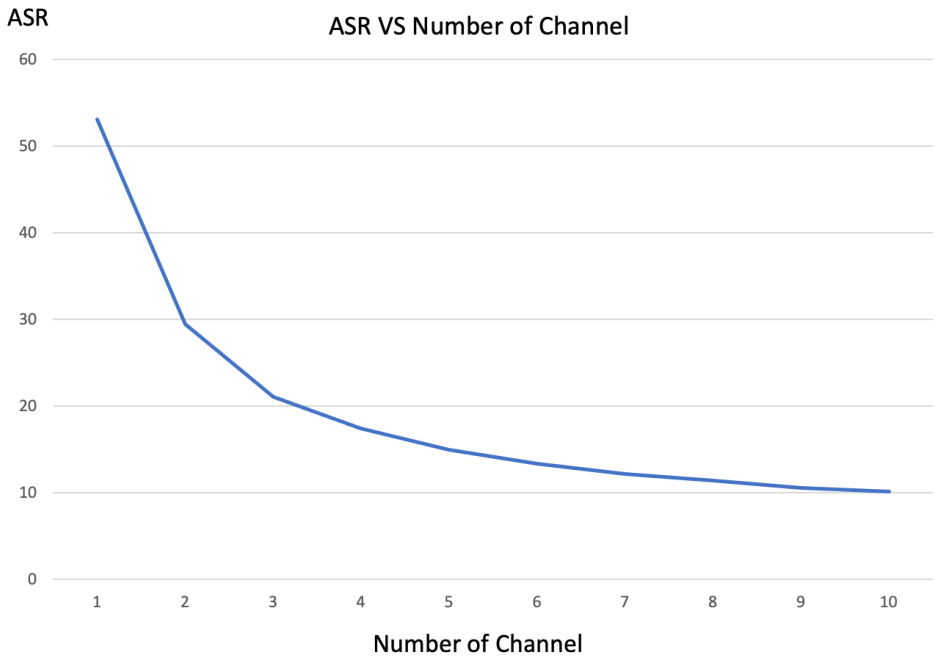


FIGURE 6.2: The attack success rate against number of channel when dealing with FGSM at 0.35 epsilson

6.3 Experiments & Result

By evaluating the average performance compared to other transformations, as shown in Table 5.3, GB was selected for BsSSL training. We evaluate the model defense ability not only over the channel number but also the adversarial epsilon value. The adversarial epsilon determines the magnitude of perturbation applied to the input image to create an adversarial example. More specifically, the FGSM algorithm works by taking the gradient of the loss function with respect to the input image and then perturbing the image by adding or subtracting epsilon times the sign of the gradient. The resulting perturbed image is the adversarial example, which is visually similar to the original image but can cause the model to make incorrect predictions.

Upon analyzing the results in Table 6.1, we observed a steady increase in the attack success rate with the increment of FGSM epsilon, which was consistent across all channel

numbers. However, we also noted that the average deviation of the attack success rate across different epsilon values was found to be smaller, at below 2%, when the channel number was increased to 7. This suggests that a larger channel number enhances the model's ability to handle distorted data more effectively and stably.

Channels	Overall Probability of FGSM ASR (%)					
	e = 0.1	e = 0.15	e = 0.2	e = 0.25	e = 0.3	e = 0.35
1	13.58	14.90	29.68	36.68	45.78	53.12
2	7.29	10.45	16.16	20.69	27.09	29.51
3	4.88	6.78	11.34	14.61	17.90	21.06
4	3.80	5.83	9.04	11.97	14.77	17.44
5	3.16	4.86	7.86	10.18	12.89	14.94
6	2.76	4.26	6.98	9.07	11.28	13.38
7	2.32	3.85	6.3	8.23	10.27	12.22
8	2.13	3.57	5.84	7.63	9.49	11.4
9	1.89	3.22	5.36	7.01	8.83	10.58
10	1.74	3.07	5.08	6.68	8.49	10.11

TABLE 6.1: Comparison of FGSM Attack Success Rate at different Block Switching channels

To further visualize the impact of the channel number on the attack success rate, we plotted the average ASR deviation over the channel number in Figure 6.3. As shown in the figure, the ASR deviation gradually decreases as the channel number increases. This implies that a larger channel number can provide better protection against adversarial attacks, resulting in a more stable and reliable performance of the model. The equation below shows the calculation of the average ASR deviation, where e is the epsilon value, $e + 1$ is the next increased epsilon value, and n is the number of channels.

$$\mathrm{ASR}_{dev} = \mathrm{mean}(\sum_{n=0}^{n} \mathrm{ASR}_{e+1} - \mathrm{ASR}_{e}) \tag{6.1}$$

6.4 Summary

Adversarial attacks are carefully crafted to utilize mathematical optimization techniques to make subtle changes to the original input, aiming to confuse the behavior of the

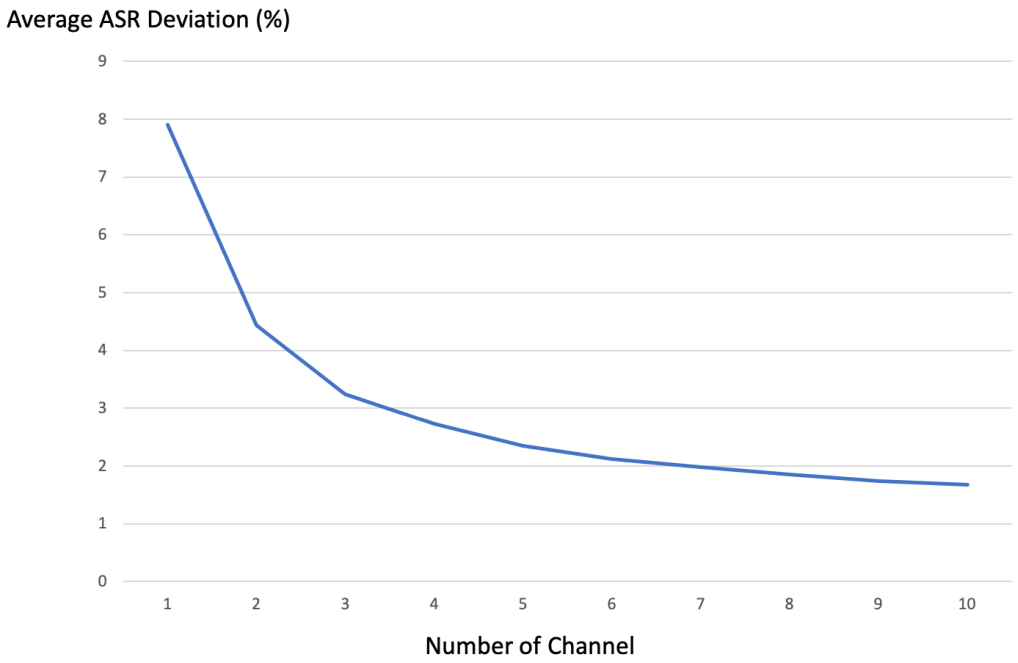


FIGURE 6.3: The average ASR deviation against number of channel

deepfake detection model. As these changes are often imperceptible to the human eye, the attacked image may still appear identical to the original input.

Despite the significant impact that adversarial attacks can have on the security of deepfake detection models, they are often overlooked during the model design process. Our current experiments, however, demonstrate that integrating the block-switching mechanism into SSL-SparcoNet can effectively mitigate such attacks. Specifically, our results show that our BS SSL-SparcoNet achieved a maximum 43% reduction in attack success rate at 0.35 FGSM epsilon when attacked by a surrogate model. Furthermore, we discovered that using a channel number of 7 or more can lead to a more stable and lower attack success rate, particularly when dealing with greater levels of perturbation.

By demonstrating the effectiveness of our proposed model, we hope to encourage further research into the development of more secure deepfake detection models that can withstand adversarial attacks. As deepfake technology continues to evolve, it is essential that we prioritize the development of robust and resilient detection models to maintain the integrity of media content and protect against malicious exploitation.

Chapter 7

Conclusions

This thesis proposes a shallower, spatially cost-efficient network structure (SparcoNet) for deepfake detection. It attempts to resolve the drawbacks of deep learning-based deepfake detection models by emphasizing achieving high detection performance at a lower training and computational cost required. It also enhances the model's defense ability against different perturbations and adversarial attacks. This chapter will conclude this study by summarizing the research objectives, contributions, limitations, and future directions.

7.1 Contributions

Three primary contributions of this thesis align with the research objectives:

1. **To devise a deepfake detection model with a low computational cost yet capable of achieving high detection accuracy**
   Inspired by Mesonet, it exploits the mesoscopic discriminative features but focuses more on local spatial feature learning and global spatial feature preservation. The thesis leverages the advantages of the network structure architecture by implementing the separable convolution, the bottleneck structure of the ResNet, and strided-convolution to create SparcoNet, a cost-efficient deep neural network for deepfake detection. Experiments results show that the proposed SparcoNet achieves an average of 0.985 AUC among six state-of-the-art deepfake datasets

and obtains a similar result for intra-datasets evaluation as the complex detection network with less than 1% differences in the generalization gap index.

2. **To embed a tertiary algorithm with the proposed detection model to enhance its robustness in dealing with unseen datasets**
   To further improve the network performance in inter-dataset evaluation and adversarial attack defense ability, the thesis proposes training the SparcoNet in a Self-Supervised Learning (SSL) manner with different data transformations. The SSL-SparcoNet incorporates a normalized temperature-scaled cross-entropy loss function to facilitate the learning of discriminative features in various data augmentations. Experiment results have shown that the SSL-SparcoNet has reduced the success rate of the Fast Gradient Sign Method (FGSM) adversarial attack by up to 85% and improved the inter-cross data evaluation performance by an average of 12%.

3. **To integrate a defense mechanism in the proposed detection model to defend against adversarial attacks**
   The thesis implements Block Switching (BS) as a framework protector to enhance the defense ability of SSL-SparcoNet against black-box adversarial attacks. The BS-SSL SparcoNet switches the runtime channel randomly to confuse the attacker in predicting the network information. Compared to other protection techniques, the proposed BS-SSL SparcoNet obtains higher flexibility in retaining network configurations. Experiment results have shown that the proposed model reduced the attack success rate of black-box adversarial attacks from a surrogate model by more than 90%.

7.2 Limitations

Although this study has made significant contributions, there are still limitations to be addressed. Firstly, the proposed network's generalization ability needs further improvement in handling diverse deepfake types. Additionally, it should be noted that the SSL method might slightly decrease the overall performance by 5-10%, requiring more training to ensure the model's ability to maintain both detection performance and defense ability.

7.3 Future Directions

In terms of future directions, there is scope for enhancing the proposed SparcoNet to tackle more complex deepfakes and improving its generalization by exploring other SSL methods, such as adversarial training. This can aid in better capturing the subtle differences between fake and real images through the encoding and decoding process.

Moreover, the development of tools for deepfake detxection is of utmost importance to counteract the rising threats of deepfakes. These tools can serve to verify the authenticity of media, protect against misinformation, and enhance the trustworthiness of digital content. It is worth considering converting SparcoNet into a user-friendly tool that can integrate with media platforms to provide value to the public.

There is also the potential applicability of SparcoNet can be extended beyond deepfake detection to other computer vision tasks, including but not limited to facial detection recognition and object detection, which would serve as a testament to its versatility and effectiveness in real-world scenarios.

Bibliography

[1] Christopher Colah. Understanding LSTM networks, 2015. URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[2] Hooman Rashidi, Nam Tran, Elham Betts, Lydia Howell, and Ralph Green. Artificial intelligence and machine learning in pathology: The present landscape of supervised methods. *Academic Pathology*, 6:237428951987308, 09 2019. doi: 10.1177/2374289519873088.

[3] V7 Labs. Neural networks activation functions. https://www.v7labs.com/blog/neural-networks-activation-functions, 2021. Accessed on April 2, 2023.

[4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, June 2016. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459.

[6] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4278–4284. AAAI Press, 2017.

[7] Robert M. Chesney and Danielle Keats Citron. Deep fakes: A looming challenge for privacy, democracy, and national security. *California Law Review*, 107:1753, 2018.

[8] Marwan Albahar and Jameel Almalki. Deepfakes: Threats and countermeasures systematic review. *Journal of Theoretical and Applied Information Technology*, 97, 2019.

[9] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv e-prints*, page arXiv:1602.07360, 2016.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[11] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5933–5942, 2019.

[12] Yuval Nirkin, Yosi Keller, and Tal Hassner. Fsgan: Subject agnostic face swapping and reenactment. pages 7184–7193, 2019.

[13] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.*, 38(4), 2019. doi: 10.1145/3306346.3323035.

[14] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. pages 8789–8797. IEEE, 2018.

[15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[16] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019.

[17] Cristian Vaccari and Andrew Chadwick. Deepfakes and disinformation: Exploring the impact of synthetic political video on deception, uncertainty, and trust in news. *Social Media + Society*, 6(1):2056305120903408, 2020. doi: 10.1177/2056305120903408.

[18] James Vincent. Watch jordan peele use ai to make barack obama deliver a psa about fake news, 2018. Retrieved Oct 19, 2019 from https://www.theverge.com/tldr/2018/4/17/17247334/ai-fake-news-video-barack-obama-jordan-peele-buzzfeed.

[19] Dave Itzkoff. How 'rogue one' brought back familiar faces, 2016. Retrieved Dec 18, 2020 from https://www.nytimes.com/2016/12/27/movies/how-rogue-one-brought-back-grand-moff-tarkin.html%20[https://perma.cc/F53C-TDYV.

[20] Medium. Ai-powered digital people, 2020. Retrieved Dec 18, 2020 from https://medium.com/syncedreview/ai-powered-digital-people-c0a94b7f0e8b.

[21] Nicholas Caporusso. Deepfakes for the good: A beneficial application of contentious artificial intelligence technology. In Tareq Ahram, editor, *Advances in Artificial Intelligence, Software and Systems Engineering*, pages 235–241. Springer International Publishing, 2020.

[22] Brian Dolhansky, Russ Howes, Ben Pflaum, Nicole Baram, and Cristian Canton Ferrer. The Deepfake Detection Challenge (DFDC) Preview Dataset. *arXiv e-prints*, page arXiv:1910.08854, 2019.

[23] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. pages 1–11, 2019.

[24] Google Research Nick Dufour and Jigsaw Andrew Gully. Contributing data to deepfake detection research, 2019. Retrieved Jul 28, 2020 from https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html.

[25] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[26] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The DeepFake Detection Challenge Dataset. *arXiv e-prints*, page arXiv:2006.07397, 2020.

[27] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Mesonet: a compact facial video forgery detection network. *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7, 2018.

[28] Liming Jiang, Ren Li, Wayne Wu, Chen Qian, and Chen Change Loy. Deeperforensics-1.0: A large-scale dataset for real-world face forgery detection. pages 2889–2898, 2020.

[29] Jia Wen Seow, Mei Kuan Lim, Raphaël C.W. Phan, and Joseph K. Liu. A comprehensive overview of deepfake: Generation, detection, datasets, and opportunities. *Neurocomputing*, 513:351–371, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.09.135. URL https://www.sciencedirect.com/science/article/pii/S0925231222012334.

[30] Jia Wen Seow, Mei Kuan Lim, Raphaël C.W. Phan, and Joseph K. Liu. Sparconet with block-switched self-supervised learning: An effective framework for deepfake detection with improved adversarial defense. 2023.

[31] Ke-Lin Du and Madisetti NS Swamy. *Neural networks and statistical learning*. Springer Science & Business Media, 2013.

[32] Anthony L Caterini and Dong Eui Chang. *Deep neural networks in a mathematical framework*. Springer, 2018.

[33] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[34] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[35] George Bebis and Michael Georgiopoulos. Feed-forward neural networks. 1994.

[36] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang. Extreme learning machine for multilayer perceptron. *IEEE Trans. Neural Networks Learn. Syst.*, 27 (4):809–821, 2016. URL http://dblp.uni-trier.de/db/journals/tnn/tnn27.html#TangDH16.

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

[38] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network, 2018. URL http://arxiv.org/abs/1808.03314. cite arxiv:1808.03314Comment: 39 pages, 10 figures, 66 references.

[39] Michael W Berry, Azlinah Mohamed, and Bee Wah Yap. *Supervised and unsupervised learning for data science*. Springer, 2019.

[40] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

[41] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[42] H Jabbar and Rafiqul Zaman Khan. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). *Computer Science, Communication and Instrumentation Devices*, 70:163–172, 2015.

[43] Tanay Agrawal. *Hyperparameter optimization in machine learning: make your machine learning and deep learning models more efficient*. Springer, 2021.

[44] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

[45] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[46] Bin Shi and Sundararaja S Iyengar. *Mathematical theories of machine learning-Theory and applications*. Springer, 2020.

[47] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris

Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[49] Md Shohel Rana, Mohammad Nur Nobi, Beddhu Murali, and Andrew H Sung. Deepfake detection: A systematic literature review. *IEEE access*, 10:25494–25513, 2022.

[50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[53] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[54] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[55] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[56] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[57] Luisa Verdoliva. Media forensics and deepfakes: an overview. *IEEE Journal of Selected Topics in Signal Processing*, 14(5):910–932, 2020.

[58] N. Kanwal, A. Girdhar, L. Kaur, and J. S. Bhullar. Detection of digital image forgery using fast fourier transform and local features. In *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pages 262–267, 2019.

[59] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv e-prints*, page arXiv:1807.03748, 2018.

[60] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. doi: 10.1561/2200000056.

[61] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. pages 1747–1756, 2016.

[62] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. page 4797–4805. Curran Associates Inc., 2016.

[63] Yuan Chen, Yang Zhao, Wei Jia, Li Cao, and Xiaoping Liu. Adversarial-learning-based image-to-image transformation: A survey. *Neurocomputing*, 411:468–486, 2020. doi: https://doi.org/10.1016/j.neucom.2020.06.067.

[64] M. Mehralian and B. Karasfi. Rdcgan: Unsupervised representation learning with regularized deep convolutional generative adversarial networks. In *2018 9th Conference on Artificial Intelligence and Robotics and 2nd Asia-Pacific International Symposium*, pages 31–38, 2018.

[65] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. *International conference on machine learning*, pages 214–223, 2017.

[66] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

[67] Y. Jo and J. Park. Sc-fegan: Face editing generative adversarial network with user's sketch and color. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1745–1753, 2019.

[68] Tingting Li, Ruihe Qian, Chao Dong, Si Liu, Qiong Yan, Wenwu Zhu, and Liang Lin. Beautygan: Instance-level facial makeup transfer with deep generative adversarial network. In *Proceedings of the 26th ACM International Conference on Multimedia*, page 645–653. Association for Computing Machinery, 2018. doi: 10.1145/3240508.3240618.

[69] Changsha Shenguronghe Network Technology Co.,Ltd. Zao, 2019. URL https://zaodownload.com/.

[70] FaceApp Inc. Faceapp, 2016. URL https://www.faceapp.com/.

[71] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[72] Jijun He, Jinjin Zheng, Yuan Shen, Yutang Guo, and Hongjun Zhou. Facial image synthesis and super-resolution with stacked generative adversarial network. *Neurocomputing*, 402:359–365, 2020. doi: https://doi.org/10.1016/j.neucom.2020.03.107.

[73] Yang Yu, Zhiqiang Gong, Ping Zhong, and Jiaxin Shan. Unsupervised representation learning with deep convolutional neural network for remote sensing images. In *International Conference on Image and Graphics*, pages 97–108. Springer, 2017.

[74] Nvidia Corporation. Nvidia, 2020. Retrieved Jan 2, 2021 from https://www.nvidia.com/en-us/.

[75] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. pages 8110–8119, 2020.

[76] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proença, and J. Fierrez. Ganprintr: Improved fakes and evaluation of the state of the art in face manipulation detection. *IEEE Journal of Selected Topics in Signal Processing*, 14(5): 1038–1048, 2020. doi: 10.1109/JSTSP.2020.3007250.

[77] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. K. Jain. On the detection of digital face manipulation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5780–5789, 2020. doi: 10.1109/CVPR42600.2020.00582.

[78] Volker Blanz, Kristina Scherbaum, Thomas Vetter, and Hans-Peter Seidel. Exchanging faces in images. In *Computer Graphics Forum*, volume 23, pages 669–676. Wiley Online Library, 2004.

[79] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. *ACM Trans. Graph.*, 34(6):183–1, 2015.

[80] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nieundefinedner. Demo of face2face: Real-time face capture and reenactment of rgb videos. In *ACM SIGGRAPH 2016 Emerging Technologies*. Association for Computing Machinery, 2016. doi: 10.1145/2929464.2929475.

[81] Ohad Fried, Ayush Tewari, Michael Zollhöfer, Adam Finkelstein, Eli Shechtman, Dan B Goldman, Kyle Genova, Zeyu Jin, Christian Theobalt, and Maneesh Agrawala. Text-based editing of talking-head video. *ACM Trans. Graph.*, 38(4), 2019. doi: 10.1145/3306346.3323028.

[82] Supasorn Suwajanakorn, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. Synthesizing obama: learning lip sync from audio. *ACM Trans. Graph.*, 36:95:1–95:13, 2017.

[83] Rithesh Kumar, Jose Sotelo, Kundan Kumar, Alexandre de Brebisson, and Yoshua Bengio. ObamaNet: Photo-realistic lip-sync from text. *arXiv e-prints*, page arXiv:1801.01442, 2017.

[84] Yang Song, Jingwen Zhu, Dawei Li, Andy Wang, and Hairong Qi. Talking face generation by conditional recurrent adversarial network. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 919–925. International Joint Conferences on Artificial Intelligence Organization, 2019. doi: 10.24963/ijcai.2019/129.

[85] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. pages 119–135. ECCV, 2018.

[86] Wayne Wu, Yunxuan Zhang, Cheng Li, Chen Qian, and Chen Change Loy. Reenactgan: Learning to reenact faces via boundary transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[87] Soumya Tripathy, Juho Kannala, and Esa Rahtu. Icface: Interpretable and controllable face reenactment using gans. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[88] Y. Sun, J. Tang, Z. Sun, and M. Tistarelli. Facial age and expression synthesis using ordinal ranking adversarial networks. *IEEE Transactions on Information Forensics and Security*, 15:2960–2972, 2020. doi: 10.1109/TIFS.2020.2980792.

[89] Yujun Shen, Ping Luo, Junjie Yan, Xiaogang Wang, and Xiaoou Tang. Faceid-gan: Learning a symmetry three-player gan for identity-preserving face synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[90] L. Tran, X. Yin, and X. Liu. Representation learning by rotating your faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(12):3007–3021, 2019. doi: 10.1109/TPAMI.2018.2868350.

[91] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9459–9468, 2019.

[92] Jiangning Zhang, Xianfang Zeng, Yusu Pan, Yong Liu, Yu Ding, and Changjie Fan. Faceswapnet: Landmark guided many-to-many face reenactment. *arXiv preprint arXiv:1905.11805*, 2, 2019.

[93] Aliaksandr Siarohin, Enver Sangineto, Stéphane Lathuilière, and Nicu Sebe. Deformable gans for pose-based human image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[94] Natalia Neverova, Riza Alp Guler, and Iasonas Kokkinos. Dense pose transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[95] Guha Balakrishnan, Amy Zhao, Adrian V. Dalca, Frédo Durand, and John Guttag. Synthesizing images of humans in unseen poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[96] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. pages 1526–1535, 2018.

[97] Kfir Aberman, Mingyi Shi, Jing Liao, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Deep video-based performance cloning. In *Computer Graphics Forum*, volume 38, pages 219–233. Wiley Online Library, 2019.

[98] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Trans. Graph.*, 37(4), 2018. doi: 10.1145/3197517.3201283.

[99] Lingjie Liu, Weipeng Xu, Michael Zollhöfer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. Neural rendering and reenactment of human actor videos. *ACM Trans. Graph.*, 38(5), 2019. doi: 10.1145/3333002.

[100] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. page 1152–1164. Curran Associates Inc., 2018.

[101] Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[102] Zhaoxiang Liu, Huan Hu, Zipeng Wang, Kai Wang, Jinqiang Bai, and Shiguo Lian. Video synthesis of human upper body with realistic face. In *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 200–202. IEEE, 2019.

[103] Yipin Zhou, Zhaowen Wang, Chen Fang, Trung Bui, and Tamara Berg. Dance dance generation: Motion transfer for internet videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2019.

[104] Ying Chen, Shixiong Xia, Jiaqi Zhao, Meng Jian, Yong Zhou, Qiang Niu, Rui Yao, and Dongjun Zhu. Person image synthesis through siamese generative adversarial network. *Neurocomputing*, 417:490–500, 2020. doi: https://doi.org/10.1016/j.neucom.2020.09.004.

[105] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. pages 1125–1134, 2017.

[106] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible Conditional GANs for image editing. *arXiv e-prints*, page arXiv:1611.06355, 2016.

[107] Mu Li, Wangmeng Zuo, and David Zhang. Deep identity-aware transfer of facial attributes. *arXiv e-prints*, page arXiv:1610.05586, 2016.

[108] Wei Shen and Rujie Liu. Learning residual images for face attribute manipulation. pages 4030–4038, 2017.

[109] Taihong Xiao, Jiapeng Hong, and Jinwen Ma. Elegant: Exchanging latent encodings with gan for transferring multiple face attributes. pages 168–184, 2018.

[110] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE Transactions on Image Processing*, 28(11): 5464–5478, 2019. doi: 10.1109/TIP.2019.2916751.

[111] M. Liu, Y. Ding, M. Xia, X. Liu, E. Ding, W. Zuo, and S. Wen. Stgan: A unified selective transfer network for arbitrary image attribute editing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3677, 2019. doi: 10.1109/CVPR.2019.00379.

[112] Xuan Nie, Haoxuan Ding, Manhua Qi, Yifei Wang, and Edward K. Wong. Urcagan: Upsample residual channel-wise attention generative adversarial network for image-to-image translation. *Neurocomputing*, 443:75–84, 2021. doi: https://doi.org/10.1016/j.neucom.2021.02.054.

[113] Jingtao Guo and Yi Liu. Attributes guided facial image completion. *Neurocomputing*, 392:60–69, 2020. doi: https://doi.org/10.1016/j.neucom.2020.02.013.

[114] Dan Ma, Bin Liu, Zhao Kang, Jiayu Zhou, Jianke Zhu, and Zenglin Xu. Two birds with one stone: Transforming and generating facial images with iterative gan. *Neurocomputing*, 396:278–290, 2020.

[115] Jiangfeng Zeng, Xiao Ma, and Ke Zhou. Photo-realistic face age progression/regression using a single generative adversarial network. *Neurocomputing*, 366:295–304, 2019. doi: https://doi.org/10.1016/j.neucom.2019.07.085.

[116] Y. Li, S. Liu, J. Yang, and M. Yang. Generative face completion. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5892–5900, 2017. doi: 10.1109/CVPR.2017.624.

[117] S. Xie and Z. Tu. Holistically-nested edge detection. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1395–1403, 2015. doi: 10.1109/ICCV.2015.164.

[118] Mahmoud Afifi, Marcus A. Brubaker, and Michael S. Brown. Histogan: Controlling colors of gan-generated and real images via color histograms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7941–7950, June 2021.

[119] Iryna Korshunova, Wenzhe Shi, Joni Dambre, and Lucas Theis. Fast face-swap using convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[120] Qianru Sun, Ayush Tewari, Weipeng Xu, Mario Fritz, Christian Theobalt, and Bernt Schiele. A hybrid model for identity obfuscation by face replacement. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[121] Lingzhi Li, Jianmin Bao, Hao Yang, Dong Chen, and Fang Wen. Advancing high fidelity identity swapping for forgery detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[122] Czfhhh Dfaker DepFA. DFaker. https://github.com/dfaker/df, 2018.

[123] Ivan Perov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Mr. Dpfks, Carl Shift Facenheim, Luis RP, Jian Jiang, Sheng Zhang, Pingyu Wu, Bo Zhou, and Weiming Zhang. DeepFaceLab: A simple, flexible and extensible face swapping framework. https://github.com/iperov/DeepFaceLab, 2020.

[124] Kvrooman Torzdf, Andenixa. Face swap. https://github.com/deepfakes/faceswap, 2020.

[125] Faceswap Web. Deepfakes web . https://faceswapweb.com/?locale=en, 2020.

[126] Mahinetube. Mahinetube. https://www.machine.tube/, 2020.

[127] NEOCORTEXT, INC. Reface app, 2020. URL https://get.reface.app/.

[128] INC. Avatarify. Avatarify: Ai face animator, 2020. URL https://apps.apple.com/us/app/avatarify-ai-face-animator/id1512669147.

[129] Wen Liu, Wenhan Luo Lin Ma Zhixin Piao, Min Jie, and Shenghua Gao. Liquid warping gan: A unified framework for human motion imitation, appearance transfer and novel view synthesis. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[130] Yarin Didi. Jiggy: Magic dance gif maker, 2020. URL https://apps.apple.com/us/app/jiggy-magic-dance-gif-maker/id1482608709.

[131] Ryota Natsume, Tatsuya Yatagawa, and Shigeo Morishima. Rsgan: Face swapping and editing using face and hair representation in latent spaces. In *ACM SIGGRAPH 2018 Posters*. Association for Computing Machinery, 2018. doi: 10.1145/3230744.3230818.

[132] Ryota Natsume, Tatsuya Yatagawa, and Shigeo Morishima. Fsnet: An identity-aware generative model for image-based face swapping. In *Asian Conference on Computer Vision*, pages 117–132. Springer, 2018.

[133] Yaohui WANG, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. Imaginator: Conditional spatio-temporal gan for video generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[134] C. Fu, Y. Hu, X. Wu, G. Wang, Q. Zhang, and R. He. High-fidelity face manipulation with extreme poses and expressions. *IEEE Transactions on Information Forensics and Security*, 16:2218–2231, 2021. doi: 10.1109/TIFS.2021.3050065.

[135] Y. Zhang, L. Zheng, and V. L. L. Thing. Automated face swapping and its detection. In *2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP)*, pages 15–19, 2017.

[136] Akshay Agarwal, Richa Singh, Mayank Vatsa, and Afzel Noore. Swapped! digital face presentation attack detection via weighted local magnitude pattern. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 659–665. IEEE, 2017.

[137] Marissa Koopman, Andrea Macarulla Rodriguez, and Zeno Geradts. Detection of deepfake video manipulation. pages 133–136, 2018.

[138] Ricard Durall, Margret Keuper, Franz-Josef Pfreundt, and Janis Keuper. Unmasking deepfakes with simple features. *arXiv-eprints*, page arXiv:1911.00686, 2019.

[139] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do gans leave artificial fingerprints? pages 506–511, 2019.

[140] Haodong Li, Bin Li, Shunquan Tan, and Jiwu Huang. Identification of deep network generated images using disparities in color components. *Signal Process.*, 174:107616, 2020.

[141] Scott McCloskey and Michael Albright. Detecting GAN-generated Imagery using Color Cues. *arXiv e-prints*, page arXiv:1812.08247, 2018.

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

[143] Davis E King. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10:1755–1758, 2009.

[144] Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 59–66. IEEE, 2018.

[145] F. Matern, C. Riess, and M. Stamminger. Exploiting visual artifacts to expose deepfakes and face manipulations. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pages 83–92, 2019.

[146] Chih-Chung Hsu, Chia-Yen Lee, and Yi-Xiu Zhuang. Learning to detect fake face images in the wild. pages 388–391, 2018.

[147] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 2015. doi: 10.1038/nature14539.

[148] W. Quan, K. Wang, D. Yan, and X. Zhang. Distinguishing between natural and computer-generated images using convolutional neural networks. *IEEE Transactions on Information Forensics and Security*, 13(11):2772–2787, 2018. doi: 10.1109/TIFS.2018.2834147.

[149] Y. Li, M. Chang, and S. Lyu. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7, 2018.

[150] H. M. Nguyen and R. Derakhshani. Eyebrow recognition for identifying deepfake videos. In *2020 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–5, 2020.

[151] Umur Aybars Ciftci, Ilke Demir, and Lijun Yin. Fakecatcher: Detection of synthetic portrait videos using biological signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):1–1, 2020. doi: 10.1109/TPAMI.2020.3009287.

[152] C. Z. Yang, J. Ma, S. Wang, and A. W. C. Liew. Preventing deepfake attacks on speaker authentication by dynamic lip movement analysis. *IEEE Transactions on Information Forensics and Security*, 16:1841–1854, 2021. doi: 10.1109/TIFS.2020.3045937.

[153] Shruti Agarwal, Hany Farid, Ohad Fried, and Maneesh Agrawala. Detecting deepfake videos from phoneme-viseme mismatches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 660–661, 2020.

[154] Steve Rubin, Floraine Berthouzoz, Gautham J Mysore, Wilmot Li, and Maneesh Agrawala. Content-based tools for editing audio stories. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 113–122, 2013.

[155] Alexandros Haliassos, Konstantinos Vougioukas, Stavros Petridis, and Maja Pantic. Lips don't lie: A generalisable and robust approach to face forgery detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5039–5049, June 2021.

[156] I. Amerini, L. Galteri, R. Caldelli, and A. Del Bimbo. Deepfake video detection through optical flow based cnn. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1205–1207, 2019.

[157] Hua Qi, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, Wei Feng, Yang Liu, and Jianjun Zhao. Deeprhythm: Exposing deepfakes with attentional visual heartbeat rhythms. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4318–4327, 2020.

[158] Shahroz Tariq, Sangyup Lee, and Simon S. Woo. A Convolutional LSTM based Residual Network for Deepfake Video Detection. *arXiv e-prints*, page arXiv:2009.07480, 2020.

[159] K. Zhang, Y. Liang, J. Zhang, Z. Wang, and X. Li. No one can escape: A general approach to detect tampered and generated image. *IEEE Access*, 7:129494–129503, 2019.

[160] A. Khodabakhsh and C. Busch. A generalizable deepfake detector based on neural conditional distribution modelling. In *2020 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–5, 2020.

[161] Jian Zhang, Jiangqun Ni, and Hao Xie. Deepfake videos detection using self-supervised decoupling network. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2021. doi: 10.1109/ICME51207.2021.9428368.

[162] Hong-Shuo Chen, Mozhdeh Rouhsedaghat, Hamza Ghani, Shuowen Hu, Suya You, and C.-C. Jay Kuo. Defakehop: A light-weight high-performance deepfake detector. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2021. doi: 10.1109/ICME51207.2021.9428361.

[163] Jiaming Li, Hongtao Xie, Jiahong Li, Zhongyuan Wang, and Yongdong Zhang. Frequency-aware discriminative feature learning supervised by single-center loss for face forgery detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6458–6467, June 2021.

[164] Yuchen Luo, Yong Zhang, Junchi Yan, and Wei Liu. Generalizing face forgery detection with high-frequency features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16317–16326, June 2021.

[165] Jessica Fridrich and Jan Kodovsky. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, 2012. doi: 10.1109/TIFS.2012.2190402.

[166] Honggu Liu, Xiaodan Li, Wenbo Zhou, Yuefeng Chen, Yuan He, Hui Xue, Weiming Zhang, and Nenghai Yu. Spatial-phase shallow learning: Rethinking face forgery detection in frequency domain. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 772–781, June 2021.

[167] Run Wang, Felix Juefei-Xu, Lei Ma, Xiaofei Xie, Yihao Huang, Jian Wang, and Yang Liu. Fakespotter: A simple yet robust baseline for spotting ai-synthesized fake faces. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, 2020.

[168] Yuezun Li and Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[169] Peng Zhou, Xintong Han, Vlad I Morariu, and Larry S Davis. Two-stream neural networks for tampered face detection. pages 1831–1839, 2017.

[170] Prabhat Kumar, Mayank Vatsa, and Richa Singh. Detecting face2face facial reenactment in videos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[171] Xurong Li, Kun Yu, Shouling Ji, Yan Wang, Chunming Wu, and Hui Xue. Fighting against deepfake: Patch&pair convolutional neural networks (ppcnn). In *Companion Proceedings of the Web Conference 2020*, pages 88–89, 2020.

[172] Md Shohel Rana and Andrew H Sung. Deepfakestack: A deep ensemble-based learning technique for deepfake detection. In *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pages 70–75. IEEE, 2020.

[173] Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. Use of a Capsule Network to Detect Fake Images and Videos. *arXiv e-prints*, page arXiv:1910.12467, 2019.

[174] H. H. Nguyen, J. Yamagishi, and I. Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2307–2311, 2019.

[175] Mengnan Du, Shiva Pentyala, Yuening Li, and Xia Hu. Towards generalizable deepfake detection with locality-aware autoencoder. In *Proceedings of the 29th ACM International Conference on Information Knowledge Management*, page 325–334. Association for Computing Machinery, 2020. doi: 10.1145/3340531.3411892.

[176] Shahroz Tariq, Sangyup Lee, Hoyoung Kim, Youjin Shin, and Simon S Woo. Detecting both machine and human created fake face images in the wild. In *Proceedings of the 2nd international workshop on multimedia privacy and security*, pages 81–87, 2018.

[177] T. Fernando, C. Fookes, S. Denman, and S. Sridharan. Detection of fake and fraudulent faces via neural memory networks. *IEEE Transactions on Information Forensics and Security*, 16:1973–1988, 2021. doi: 10.1109/TIFS.2020.3047768.

[178] Hanqing Zhao, Wenbo Zhou, Dongdong Chen, Tianyi Wei, Weiming Zhang, and Nenghai Yu. Multi-attentional deepfake detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2185–2194, June 2021.

[179] Chih-Chung Hsu, Yi-Xiu Zhuang, and Chia-Yen Lee. Deep fake image detection based on pairwise learning. *Applied Sciences*, 10:370, 01 2020. doi: 10.3390/app10010370.

[180] Y. Zhuang and C. Hsu. Detecting generated image based on a coupled network with two-step pairwise learning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3212–3216, 2019. doi: 10.1109/ICIP.2019.8803464.

[181] Trisha Mittal, Uttaran Bhattacharya, Rohan Chandra, Aniket Bera, and Dinesh Manocha. Emotions don't lie: An audio-visual deepfake detection method using affective cues. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2823–2832, 2020.

[182] Huaxiao Mo, Bolin Chen, and Weiqi Luo. Fake faces identification via convolutional neural network. In *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, pages 43–47, 2018.

[183] Zhiqing Guo, Gaobo Yang, Jiyou Chen, and Xingming Sun. Fake face detection via adaptive residuals extraction network. *arXiv e-prints*, page arXiv:2005.04945, 2020.

[184] Nhu-Tai Do, In-Seop Na, and Soo-Hyung Kim. Forensics face detection from gans using convolutional neural network, 2018.

[185] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

[186] Ekraam Sabir, Jiaxin Cheng, Ayush Jaiswal, Wael AbdAlmageed, Iacopo Masi, and Prem Natarajan. Recurrent convolutional strategies for face manipulation

detection in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[187] A. Mehra. Deepfake detection using capsule networks with long short-term memory networks, August 2020. Retrieved Dec 18, 2020 from http://essay.utwente.nl/83028/.

[188] Irene Amerini and Roberto Caldelli. Exploiting prediction error inconsistencies through lstm-based classifiers to detect deepfake videos. In *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, pages 97–102, 2020.

[189] Iacopo Masi, Aditya Killekar, Royston Marian Mascarenhas, Shenoy Pratik Gurudatt, and Wael AbdAlmageed. Two-branch recurrent network for isolating deepfakes in videos. In *European Conference on Computer Vision*, pages 667–684. Springer, 2020.

[190] Zekun Sun, Yujie Han, Zeyu Hua, Na Ruan, and Weijia Jia. Improving the efficiency and robustness of deepfakes detection through precise geometric features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3609–3618, June 2021.

[191] Hasam Khalid and Simon S Woo. Oc-fakedect: Classifying deepfakes using one-class variational autoencoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 656–657, 2020.

[192] Tianfei Zhou, Wenguan Wang, Zhiyuan Liang, and Jianbing Shen. Face forensics in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5778–5788, June 2021.

[193] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[194] Tiancai Wang, Tong Yang, Jiale Cao, and Xiangyu Zhang. Co-mining: Self-supervised learning for sparsely annotated object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2800–2808, 2021.

[195] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *Advances in Neural Information Processing Systems*, 33:3407–3418, 2020.

[196] Liang Chen, Yong Zhang, Yibing Song, Lingqiao Liu, and Jue Wang. Self-supervised learning of adversarial example: Towards good generalizations for deepfake detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18710–18719, 2022.

[197] Pavel Korshunov and Sebastien Marcel. DeepFakes: a New Threat to Face Recognition? Assessment and Detection. *arXiv e-prints*, page arXiv:1812.08685, 2018.

[198] Conrad Sanderson. The VidTIMIT Database. Idiap-Com Idiap-Com-06-2002, IDIAP, 2002.

[199] A. Khodabakhsh, R. Ramachandra, K. Raja, P. Wasnik, and C. Busch. Fake face detection methods: Can they be generalized? In *2018 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–6, 2018.

[200] Yinan He, Bei Gan, Siyu Chen, Yichun Zhou, Guojun Yin, Luchuan Song, Lu Sheng, Jing Shao, and Ziwei Liu. Forgerynet: A versatile benchmark for comprehensive forgery analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4360–4369, June 2021.

[201] Houwei Cao, David G Cooper, Michael K Keutmann, Ruben C Gur, Ani Nenkova, and Ragini Verma. Crema-d: Crowd-sourced emotional multimodal actors dataset. *IEEE transactions on affective computing*, 5(4):377–390, 2014.

[202] Steven R Livingstone and Frank A Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391, 2018.

[203] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv e-prints*, page arXiv:1806.05622, 2018.

[204] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T Freeman, and Michael Rubinstein. Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. *arXiv-eprints*, page arXiv:1804.03619, 2018.

[205] Chengrui Wang and Weihong Deng. Representative forgery mining for fake face detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14923–14932, June 2021.

[206] Y. Nirkin, L. Wolf, Y. Keller, and T. Hassner. Deepfake detection based on discrepancies between faces and their context. *IEEE Transactions on Pattern Analysis Machine Intelligence*, pages 1–1, 2021.

[207] Xiangyu Zhu, Hao Wang, Hongyan Fei, Zhen Lei, and Stan Z. Li. Face forgery detection by 3d decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2929–2939, June 2021.

[208] Yisroel Mirsky and Wenke Lee. The creation and detection of deepfakes: A survey. *ACM Computing Surveys (CSUR)*, 54(1):1–41, 2021.

[209] Xin Tong, Luona Wang, Xiaoqin Pan, and Jin gya Wang. An overview of deepfake: The sword of damocles in ai. In *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pages 265–273. IEEE, 2020.

[210] T. Zhang, L. Deng, L. Zhang, and X. Dang. Deep learning in face synthesis: A survey on deepfakes. In *2020 IEEE 3rd International Conference on Computer and Communication Engineering Technology (CCET)*, pages 67–70, 2020. doi: 10.1109/CCET50901.2020.9213159.

[211] Weixin Liang, Zixuan Liu, and Can Liu. DAWSON: A Domain Adaptive Few Shot Generation Framework. *arXiv e-prints*, page arXiv:2001.00576, 2020.

[212] Zhikai Chen, Lingxi Xie, Shanmin Pang, Yong He, and Bo Zhang. Magdr: Mask-guided detection and reconstruction for defending deepfakes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9014–9023, June 2021.

[213] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *Advances in Neural Information Processing Systems*, pages 7137–7147, 2019.

[214] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346 – 360, 2020. doi: https://doi.org/10.1016/j.eng.2019.12.012.

[215] Xiao Wang, Siyue Wang, Pin-Yu Chen, Xue Lin, and Peter Chin. Block switching: a stochastic approach for deep learning security. *arXiv preprint arXiv:2002.07920*, 2020.

[216] Shilin Qiu, Qihe Liu, Shijie Zhou, and Chunjiang Wu. Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences*, 9(5), 2019. ISSN 2076-3417. doi: 10.3390/app9050909. URL https://www.mdpi.com/2076-3417/9/5/909.

[217] Apurva Gandhi and Shomik Jain. Adversarial Perturbations Fool Deepfake Detectors. *arXiv e-prints*, art. arXiv:2003.10596, March 2020.

[218] Hanqing Zhao, Wenbo Zhou, Dongdong Chen, Tianyi Wei, Weiming Zhang, and Nenghai Yu. Multi-attentional deepfake detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2185–2194, 2021.

[219] Paarth Neekhara, Brian Dolhansky, Joanna Bitton, and Cristian Canton Ferrer. Adversarial threats to deepfake detection: A practical perspective. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 923–932, 2021.

[220] Shehzeen Hussain, Paarth Neekhara, Malhar Jere, Farinaz Koushanfar, and Julian McAuley. Adversarial deepfakes: Evaluating vulnerability of deepfake detectors to adversarial examples. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3348–3357, 2021.

[221] Shizhao Sun, Wei Chen, Liwei Wang, Xiaoguang Liu, and Tie-Yan Liu. On the depth of deep neural networks: A theoretical view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[222] Tim Schröder and Michael Schulz. Monitoring machine learning models: a categorization of challenges and methods. *Data Science and Management*, 5(3):105–116, 2022. ISSN 2666-7649. doi: https://doi.org/10.1016/j.dsm.2022.07.004. URL https://www.sciencedirect.com/science/article/pii/S2666764922000303.

[223] Yunus Camgözlü and Yakup Kutlu. Analysis of filter size effect in deep learning. *arXiv preprint arXiv:2101.01115*, 2020.

[224] DLib. https://github.com/davisking/dlib, 2021.

[225] Huachun Yang and Xu Wang. Cascade classifier for face detection. *Journal of Algorithms Computational Technology*, 10, 05 2016. doi: 10.1177/1748301816649073.

[226] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[227] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. In *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, pages 159–164, 2017.

[228] Hong-Shuo Chen, Mozhdeh Rouhsedaghat, Hamza Ghani, Shuowen Hu, Suya You, and C-C Jay Kuo. Defakehop: A light-weight high-performance deepfake detector. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[229] Nyee Thoang Lim, Meng Yi Kuan, Muxin Pu, Mei Kuan Lim, and Chun Yong Chong. Metamorphic testing-based adversarial attack to fool deepfake detectors. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 2503–2509. IEEE, 2022.

[230] Muxin Pu, Meng Yi Kuan, Nyee Thoang Lim, Chun Yong Chong, and Mei Kuan Lim. Fairness evaluation in deepfake detection models using metamorphic testing. In *2022 IEEE/ACM 7th International Workshop on Metamorphic Testing (MET)*, pages 7–14. IEEE, 2022.

[231] Shehzeen Hussain, Paarth Neekhara, Brian Dolhansky, Joanna Bitton, Cristian Canton Ferrer, Julian McAuley, and Farinaz Koushanfar. Exposing vulnerabilities of deepfake detection systems with robust attacks. *Digital Threats*, 3(3), feb 2022. ISSN 2692-1626. doi: 10.1145/3464307. URL https://doi.org/10.1145/3464307.

[232] M. Pu, M. Kuan, N. Lim, C. Chong, and M. Lim. Fairness evaluation in deepfake detection models using metamorphic testing. In *2022 IEEE/ACM 7th International Workshop on Metamorphic Testing (MET)*, pages 7–14, Los Alamitos, CA,

USA, may 2022. IEEE Computer Society. doi: 10.1145/3524846.3527337. URL https://doi.ieeecomputersociety.org/10.1145/3524846.3527337.

[233] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[234] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, art. arXiv:1412.6572, December 2014.

[235] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[236] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[237] Yuancheng Li and Yimeng Wang. Defense against adversarial attacks in deep learning. *Applied Sciences*, 9(1):76, 2018.