

MACHINE LEARNING METHODS FOR DATA QUALITY ASPECTS
IN EDGE COMPUTING PLATFORMS

Alakananda Mitra, B.Sc., B.Tech., M.Tech.

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

December 2022

APPROVED:

Saraju P. Mohanty, Major Professor

Elias Kougianos, Co-Major Professor

Hui Zhao, Committee Member

Cihan Tunc, Committee Member

Gergely Záruba, Chair of the Department
of Computer Science and
Engineering

Shengli Fu, Interim Dean of the College of
Engineering

Victor Prybutok, Dean of the Toulouse
Graduate School

Mitra, Alakananda. *Machine Learning Methods for Data Quality Aspects in Edge Computing Platforms*. Doctor of Philosophy (Computer Science and Engineering), December 2022, 212 pp., 37 tables, 90 figures, 256 numbered references.

In this research, three aspects of data quality with regard to artificial intelligence (AI) have been investigated: detection of misleading fake data, especially deepfakes, data scarcity, and data insufficiency, especially how much training data is required for an AI application. Different application domains where the selected aspects pose issues have been chosen. To address the issues of data privacy, security, and regulation, these solutions are targeted for edge devices. In Chapter 3, two solutions have been proposed that aim to preempt such misleading deepfake videos and images on social media. These solutions are deployable at edge devices. In Chapter 4, a deepfake resilient digital ID system has been described. Another data quality aspect, data scarcity, has been addressed in Chapter 5. One of such agricultural problems is estimating crop damage due to natural disasters. Data insufficiency is another aspect of data quality. The amount of data required to achieve acceptable accuracy in a machine learning (ML) model has been studied in Chapter 6. As the data scarcity problem is studied in the agriculture domain, a similar scenario—plant disease detection and damage estimation—has been chosen for this verification. This research aims to provide ML or deep learning (DL)-based methods to solve several data quality-related issues in different application domains and achieve high accuracy. We hope that this work will contribute to research on the application of machine learning techniques in domains where data quality is a barrier to success.

Copyright 2022
by
Alakananda Mitra

ACKNOWLEDGMENTS

I would like to sincerely thank and express my gratitude to my advisor Prof. Saraju P. Mohanty for holding my hand in each step of the PhD program. Not only did I find him to be a brilliant and thoughtful guide, but also a teacher with kindness who always keeps his students' first and gives timely advice on their research and career. He taught me how to assess a research problem before plunging in, question my methods at each stage, and write up my findings. His constant encouragement, supervision, constructive feedback, and invaluable suggestions have made this dissertation possible. He encouraged me to explore different research areas and gave me the opportunity to collaborate with others. He is a great educator, mentor, and guide. I sincerely appreciate what you did for me, Sir.

I would also like to thank Prof. Elias Kougiianos for guiding me in my Ph.D. journey as co-major professor. His unconditional support and trust on me made me confident and a better researcher. Without his help, the papers would not have come error-free. Thank you Dr. Kougiianos for being supportive and kind to me during my difficult times.

I greatly appreciate my dissertation committee members, Dr. Hui Zhao and Dr. Cihan Tunc, for taking the time out from their busy schedules and supporting me. This dissertation would not have been possible without their encouragement and guidance. I would like to sincerely thank the Faculties and Staffs of the Department of Computer Science and Engineering for recognizing me as the Outstanding Early-Stage PhD Student 2022 and supporting me through assistantships and travel grants during my doctoral studies.

I am greatly honored to have Prof. Peter Corcoran as my mentor. I have learned innumerable things, from the fundamentals of computer vision and deep learning techniques to state-of-the-art methods, from him and his group. All those bi-weekly meetings were very insightful.

I would like to extend my thanks to Prof. Chittaranjan Ray for believing in me. His detailed and constructive comments made the crop damage estimation paper better. I have learned much from him, not only the agricultural domain knowledge, but also in terms of discipline and helping others.

A special thank goes to Dr. Laavanya Rachakonda for always giving me the necessary information for completing this dissertation.

I would also like to thank all my lab colleagues to keep my motivation up and making this journey beautiful.

I can not express my gratitude in words to my husband, Mr. Chiradeep Das, for standing by me through this journey and beyond. I am deeply grateful to him for bearing me, inspiring me, and understanding me even when I did not understand myself. His constant inspiration, unfailing support, and strong criticism made this dissertation better.

I am really lucky to have a daughter like Anusha, who never complained about anything. Without her understanding and sacrifice, I would not have completed this dissertation.

I would like to end by saying that nothing would have been possible without the continuous inspiration, staunch support, unwavering faith, and unconditional love that my parents, Mr. Nirod Mitra and Mrs. Anjali Mitra, have shown towards me. They are my support system. They have respected all my choice and have never once questioned my judgment. They instilled in me the values of perseverance, diligence, and dignity in the face of setbacks. I owe my parents absolutely everything, and possibly even more.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	iii
LIST OF TABLES	xi
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS	xviii
LIST OF NOTATIONS	xix
LIST OF PUBLISHED MATERIALS REPRODUCED IN THE DISSERTATION	xx
CHAPTER 1 INTRODUCTION	1
1.1. Artificial Intelligence (AI)	1
1.1.1. The Beginning	1
1.1.2. Today and Tomorrow	2
1.2. Artificial Intelligence (AI)/ Machine Learning (ML)/ Deep Learning (DL), and Computer Vision (CV)	2
1.3. Data Quality Aspects	4
1.3.1. Deepfake: A Type of Fake Data	7
1.3.2. Data Scarcity	9
1.3.3. Data Insufficiency	10
1.4. Computing Platforms	10
1.4.1. Cloud Computing Platform	10
1.4.2. Fog Computing Platform	11
1.4.3. Edge Computing Platform	11
1.5. ML Methods for Data Quality Aspects in Edge Computing Platforms: Problems and Objectives	12
1.5.1. Deepfake Data	12

1.5.2.	Data Scarcity and Data Insufficiency	13
1.6.	Main Contributions	14
1.6.1.	Contribution I-ML/DL Methods for Fake Data Detection on Edge Devices	14
1.6.2.	Contribution II-A Deepfake Resilient Digital ID System for Smart Cities	17
1.6.3.	Contribution III-DL-based Solution for Data Scarcity Problem on Edge Devices	20
1.6.4.	Contribution IV-Verification of the Effect of Data Insufficiency on Accuracy	21
1.7.	Dissertation Organization	24
CHAPTER 2 RELATED PRIOR WORKS		25
2.1.	Deepfake Video Detection	25
2.2.	Deepfake Image Detection	30
2.3.	Crop Damage Estimation - Damage Caused by Natural Calamities	31
2.4.	Leaf Damage Estimation - Damage Caused by Plant Diseases	35
2.4.1.	Single Plant Crop Diseases Detection	35
2.4.2.	Multi Plants Crops Diseases Detection	36
2.5.	Discussions	37
CHAPTER 3 DATA FORGERY: DETECTION OF DEEPPFAKE VIDEOS AND IMAGES IN SOCIAL MEDIA		38
3.1.	Introduction	38
3.2.	Deepfakes: Threat Environment	38
3.3.	Why are Deepfakes Hard to Detect?	41
3.3.1.	By Autoencoders	41
3.3.2.	By Generative Adversarial Networks(GAN)	41
3.4.	Addressed Research Problem	42

3.5.	Social Media Deepfake Video Detection	44
3.5.1.	The Proposed Solution	44
3.5.2.	Theoretical Perspective	48
3.5.3.	Experimental Validation	52
3.5.4.	Results	58
3.5.5.	Discussions	64
3.6.	Social Media Deepfake Image Detection at Edge Platform	65
3.6.1.	Overview	65
3.6.2.	Detection Methodology at Edge Platform	66
3.6.3.	Experimental Validation	69
3.6.4.	Performance Evaluation	72
3.6.5.	Discussions	74
CHAPTER 4 DATA FALSIFICATION: DEEPFAKE RESILIENT DIGITAL IDENTIFICATION FOR SMART CITIES		78
4.1.	Digital ID System for Smart Cities	78
4.1.1.	Role of Digital ID in Smart City	79
4.1.2.	Challenges of Digital Identification System	80
4.2.	Background	83
4.2.1.	Cryptographic Key Generation	84
4.2.2.	Biometric Authentication Systems in IoT	84
4.2.3.	FR Systems	85
4.2.4.	Facial Features-based Authentication Systems	86
4.2.5.	Attack Detection	87
4.3.	iFace: Proposed Biometric-based Digital ID for Smart Cities	88
4.3.1.	End-to-End System Level Architecture	88
4.3.2.	System Overview	90
4.3.3.	Deepfake Attack Detection	91
4.3.4.	Digital ID System	94

4.3.5.	Discussions	102
4.4.	iFace 1.1: Biometric-based Improved Digital ID System for Smart Cities	102
4.4.1.	System Overview	102
4.4.2.	System Modules	103
4.4.3.	Performance Evaluation of the Proposed Digital ID System	120
4.4.4.	Conclusions and Future Work	123
4.5.	Discussions	126
CHAPTER 5 DATA SCARCITY: A NOVEL FRAMEWORK FOR AUTOMATIC		
CROP DAMAGE ESTIMATION		127
5.1.	Introduction	127
5.2.	Addressed Research Problem	128
5.3.	Proposed Solution: eCrop	130
5.4.	eCrop: A Novel Method to Evaluate the Extent of Crop Damage	130
5.4.1.	Proposed Agro Cyber Physical System (A-CPS)	130
5.4.2.	Proof-of-Concept of eCrop	132
5.4.3.	eCrop Grid	133
5.4.4.	eCrop Grid Generation	134
5.4.5.	Extent of Damage Calculation	135
5.5.	Meta Learning-based Detection of Crop Damage for Each Grid	137
5.5.1.	Architecture	137
5.5.2.	Data Pair Generation	138
5.5.3.	Energy Function and Similarity Score	140
5.5.4.	Method and Training Protocol	140
5.5.5.	Loss	141
5.5.6.	Proposed Algorithm	142
5.6.	Evaluation of the Proposed Crop Damage Detection Method for Each	
	Grid	142
5.6.1.	Dataset	142

5.6.2.	Implementation	142
5.6.3.	Validation	145
5.7.	Results and Comparative Study	145
5.8.	Conclusions and Future Work	151
CHAPTER 6 DATA INSUFFICIENCY: A NOVEL FRAMEWORK FOR PLANT DISEASE DETECTION AND LEAF DAMAGE ESTIMATION		153
6.1.	Introduction	153
6.2.	Addressed Research Problem	154
6.3.	Proposed Solution	156
6.4.	aGROdet: Proposed Method for Detection of Plant Disease and Damage Estimation	156
6.4.1.	Proposed A-CPS	156
6.4.2.	Plant Disease Detection	157
6.4.3.	Estimation of Leaf Damage Severity	162
6.4.4.	Performance Evaluation of aGROdet	167
6.5.	Conclusions and Future Work	173
CHAPTER 7 CONCLUSIONS AND FUTURE WORK		176
7.1.	Contribution I: Summary, Limitations, and Future Work	176
7.1.1.	Detection of Deepfake Videos	176
7.1.2.	Detection of Deepfake Images at Edge	177
7.2.	Contribution II: Summary, Limitations, and Future Work	178
7.2.1.	iFace	178
7.2.2.	iFace 1.1	179
7.3.	Contribution III: Summary, Limitations, and Future Work	181
7.4.	Contribution IV: Summary, Limitations, and Future Work	182
7.5.	Discussions	184
REFERENCES		185

LIST OF TABLES

		Page
2.1	A Comparative Perspective with Existing Works on Deepfake Video Detection.	28
2.2	A Comparative Perspective of Existing Works with eCrop.	33
3.1	Dataset for Selecting Feature Extractor.	53
3.2	Dataset Details for Deepfake Detection.	53
3.3	Details of the Frames for Training and Validation.	58
3.4	Confusion Matrix - Definition of TP, TN, FP and FN.	63
3.5	Performance Evaluation of the Detection Method.	65
3.6	Performance Comparison of Xception Network paired with Proposed Classifier.	67
3.7	Classification report of EasyDeep on test images.	72
3.8	Accuracy Variation with Tree Structure.	74
4.1	Vulnerable Points in DIS.	82
4.2	Dataset for Deepfake Detection in iFace System.	93
4.3	Classification Report for Deepfake Detection of iFace.	94
4.4	Dataset Details for iFace Verification.	100
4.5	Performance Evaluation of iFace.	100
4.6	Dataset for Deepfake Detection of iFace 1.1.	109
4.7	Dataset Division for both DF-TIMIT HQ and LQ Dataset.	110
4.8	Accuracy and Inference Time for Different Evaluation Scenarios.	113
4.9	Classification report-trained and tested on DF-TIMIT HQ.	114
4.10	Performance Comparison of Deepfake Detection Module of iFace 1.1 with State-of-the-Art Solutions.	115
4.11	Dataset for Presentation Attack Detection Module.	115
4.12	Classification report of presentation attack module -trained and tested on Replay Attack dataset.	116

4.13	Customized Dataset for Face Features Extraction Module and Classification Module.	118
4.14	Performance of the Proposed Digital ID System iFace 1.1.	123
4.15	Performance Metrics of Facial Authentication System.	124
4.16	Performance Comparison of the Proposed Facial Authentication System with Existing Papers.	125
5.1	Sister Network Architecture Details.	139
5.2	Dataset Details.	144
5.3	No. of Images for Training, Validation, and Testing.	148
5.4	Accuracy for Different N and K.	149
5.5	A Quantitative Comparison of the Current Paper with Existing Works.	151
6.1	CNN Architecture for Plant Disease Identification.	159
6.2	Grade Scale for Calculating Damage Severity.	167
6.3	Accuracy for Disease Detection Network.	169
6.4	Effect of Data Insufficiency on Accuracy (Trained without reduced learning rate).	171
6.5	Damage Severity Prediction through aGROdet.	173
6.6	A Quantitative Analysis of the Current Paper with Existing Works.	174

LIST OF FIGURES

		Page
1.1	AI application triangle: key components of AI application.	3
1.2	Relation among AI, ML, DL, and CV [6]. Yellow shaded area describes our area of work.	5
1.3	Different data quality aspects.	6
1.4	Deepfakes created by Facebook to fight against a disinformation disaster-source Facebook.	7
1.5	Different computing platforms.	11
1.6	Context of EasyDeep.	17
1.7	IEEE smart village map for smart energy projects [13].	23
2.1	Outline of the current chapter.	25
3.1	Deepfake creation by autoencoder.	42
3.2	Deepfake creation by GAN.	43
3.3	System level overview of the proposed network.	44
3.4	A detailed representation of the proposed method.	45
3.5	Generated key video frames from a 20 second video.	46
3.6	XceptionNet as the feature extractor.	47
3.7	Classifier network.	48
3.8	Different deepfake datasets [71].	53
3.9	Key video frames from different length videos.	55
3.10	The proposed flow of video processing.	56
3.11	Accuracy and loss plots for two different scenarios. (a) and (b) are accuracy and loss plots respectively, when the end-to-end network is trained for 8 epochs. (c) and (d) are accuracy and loss plots respectively, when the classifier is first trained for 4 epochs, keeping the feature extractor's weight frozen, and then the end-to-end network is trained for 10 epochs.	57

3.12	End-to-end workflow.	58
3.13	Developmental workflow of the detection system.	59
3.14	Selection of CNN as feature extractor.	60
3.15	Test accuracy for various testing scenarios.	61
3.16	Sample CNN layers outputs.	62
3.17	Confusion matrix.	64
3.18	Accuracy comparison.	66
3.19	Overall workflow diagram at edge platform.	68
3.20	Detection API workflow.	70
3.21	StarGAN generated sample images.	70
3.22	Histogram comparison of StarGAN generated images and real images.	71
3.23	Implementation of EasyDeep.	71
3.24	Performance metrics calculation for EasyDeep.	73
4.1	Components of smart city.	79
4.2	Mandatory requirements for digital ID in a smart city.	83
4.3	End-to-end system level framework for digital ID system in a smart city.	89
4.4	Registration of a new user in iFace.	90
4.5	Authentication of existing user in iFace.	91
4.6	Deepfake image detection method.	92
4.7	Facial landmark points detection workflow.	95
4.8	Facial distance calculation.	96
4.9	Binary key generation from face landmark points.	97
4.10	Face matching workflow.	98
4.11	Performance evaluation of iFace for different datasets.	101
4.12	New user registration in iFace 1.1.	103
4.13	Authentication at edge in iFace 1.1.	104
4.14	Username generation module workflow.	105
4.15	Sample lookup table.	106

4.16	Extracted faces using Haar Cascade (middle) and Dlib HoG (right most) face detectors from original face (left most) (photo courtesy: Microsoft Power Point).	106
4.17	Attack detection module.	107
4.18	Feature visualization of MobileNetV2 for sample layers.	111
4.19	Class activation map visualization using GRAD-CAM for MobileNetV2.	112
4.20	Confusion matrix - trained and tested on DF-TIMIT HQ.	113
4.21	Confusion matrix for presentation attack module.	116
4.22	Training of face features extraction module.	117
4.23	Embedding plots of six main characters of American sitcom “Friends”.	119
4.24	Classifier training.	120
4.25	Authentication process of iFace 1.1.	120
4.26	Embedding plot of sample person in the clustered training dataset.	122
4.27	Digital ID performance.	123
5.1	Effects of climate change - drought, wildfire, ice melting at the poles, flood, and storms	128
5.2	Corn yield projection in 2070 [73]. In the color gradient scale, red means the mostly affected whereas green means not affected.	129
5.3	Proposed agro cyber physical system (A-CPS). eCrop is a part of the proposed A-CPS.	131
5.4	eCrop grid for evaluating the extent of crop damage.	132
5.5	eCrop system overview.	133
5.6	Automatic detection of crop damage for each grid.	134
5.7	Grid generation of eCrop system for detecting crop damage.	135
5.8	CNN structure used in sister networks of damage detection system.	138
5.9	eCrop network consisting of CNN structure in Fig. 5.8 as sister networks.	139
5.10	Sample damaged grain images used for training [5]. In practice images taken by UAV will be used.	144

5.11	Crop damage detection pipeline.	145
5.12	Accuracy vs number of shots (K) for number of ways (N)=4.	148
5.13	Accuracy vs number of shots (K) for number of ways (N)=3.	149
5.14	Training time vs number of shots (K) for number of ways (N)=4.	149
5.15	Training time vs number of shots (K) for number of ways (N)=3.	150
5.16	First 3 features of support images from the best trained model for N=4.	150
6.1	Agricultural problems solvable using agriculture cyber physical systems.	154
6.2	Disease triangle [204].	155
6.3	Agriculture cyber physical system.	157
6.4	Plant disease detection network.	158
6.5	Sample images from PlantVillage dataset [108].	160
6.6	Sample augmented data. Data is augmented on the fly for different rotation, zoom, brightness, horizontal and vertical flip.	161
6.7	Plant disease detection workflow.	161
6.8	Leaf damage estimation workflow.	162
6.9	Leaf area detection by creating leaf mask. a. Input image b. background segmentation c. mask creation for the leaf d. noise reduction from the mask. Red large ovals show the shadow around the foreground object and small circles highlight the shadows on the foreground object.	164
6.10	Removal of shadow around the leaf. a. Input image b. detection of shadow around the leaf c. shadow removal d. leaf mask creation e. noise reduction from the mask. Red large ovals show shadow around the leaf and brown ovals highlight the shadow on the leaf.	165
6.11	Leaf damage area detection a. Input image b. detection of shadow around the leaf c. shadow removal d. leaf mask creation e. merging of mask and input image f. recoloration of the black background to differentiate them from the damage g. damage mask creation.	166
6.12	Leaf damage estimation.	167

6.13	Confusion matrix for disease detection network (trained without reduced learning rate). Classes are denoted by numbers instead of the class names to fit into the figure space.	168
6.14	Performance evaluation curves for disease detection (trained without reduced learning rate).	170
6.15	Accuracy and loss plots at various scenarios. a. data augmentation has been used. b. and c. no data augmentation has been used.	171

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
NLP	Natural language Processing
FF++	FaceForensics ++
DFDC	Deepfake Detection Challenge
GLCM	Gray Level Co-occurrence Matrix
GAN	Generative Adversarial Network
CNN	Convolutional Neural Network
NFF	Neutral Frontal Face
UAV	Unmanned Aerial Vehicle
A-CPS	Agriculture Cyber Physical System
IoT	Internet-of-Things
IoAT	Internet-of-Agro-Things
AUC	Area Under the Curve
ROC	Receiver Operating Characteristic
ICT	Information and Communication Technologies
FR	Facial Recognition
AD	Attack Detection
DD	Deepfake Detection
PAD	Presentation Attack Detection
FFE	Face Features Extraction
CM	Confusion Matrix
PCA	Principal Component Analysis
FRR	False Rejection Ratio
FAR	False Acceptance Ratio
EER	Equal Error Rate

LIST OF NOTATIONS

- Chapter 3 $I \rightarrow$ Input Image; $h \rightarrow$ Filter Kernel; $N \rightarrow$ Number of Filters; $M \rightarrow$ Number of Channels; $D_G \rightarrow$ Output Size; $D_K \rightarrow$ Filter Size; $n \rightarrow$ Number of Gray Levels; $p(i, j) \rightarrow$ GLCM Element for the distance d between gray level values i and j ; $\mu \rightarrow$ Mean Intensity of All Gray Values Present; $\sigma \rightarrow$ Variance of the Intensities of All Gray Values Present.
- Chapter 4 $d_m \rightarrow$ Threshold Value; $\mathcal{F}_{1b} \rightarrow$ Binary Feature Vector; $\mathcal{F}_{io} \rightarrow$ Final Feature Vector; $\mathcal{F}_{mod} \rightarrow$ Modified Feature Vector; $\mathfrak{F}_{io} \rightarrow$ Encoded Bio Key.
- Chapter 5 $\phi \rightarrow$ Latitude; $\lambda \rightarrow$ Longitude; $R \rightarrow$ Earth's Radius; $c \rightarrow$ Insured value in $\$/sq.m.$; $u \rightarrow$ Number of Damaged Grids; $M \rightarrow$ Claim Value; $Y \rightarrow$ Label Value; $D_w \rightarrow$ Euclidean Distance; $L_{con} \rightarrow$ Contrastive Loss; $Y \rightarrow$ Label Value; $N \rightarrow$ Number of Ways; $K \rightarrow$ Number of Shots.

LIST OF PUBLISHED MATERIALS REPRODUCED IN THE DISSERTATION

The following articles have been reproduced, either in whole in part, in this dissertation with permission from the copyright holder:

- Chapters 1 and 7 presents material from Articles 1, 2, 3, 5, 6, and 7.
 - Chapter 2 presents material from Articles 1, 2, 6 and 7.
 - Chapter 3 presents significant portions of material from Articles 1 and 2.
 - Chapter 4 presents significant portions of material from Articles 3, 4, and 5.
 - Chapter 5 is adapted entirely from Article 6.
 - Chapter 6 is adapted entirely from Article 7.
- (1) A. Mitra, S. P. Mohanty, P. Corcoran, and E. Kougianos, “A Machine Learning Based Approach for DeepFake Detection in Social Media through Key Video Frame Extraction,” Springer Nature Computer Science (SN-CS), Vol. 2, No. 2, Feb 2021, pp. 1–18, DOI: <https://doi.org/10.1007/s42979-021-00495-x>, with permission from Springer Nature.
 - (2) A. Mitra, S. P. Mohanty, P. Corcoran, and E. Kougianos, “EasyDeep: An IoT Friendly Robust Detection Method for GAN Generated DeepFake Images in Social Media,” in Proceedings of the 4th IFIP International Internet of Things (IoT) Conference (IFIP-IoT), 2021, DOI: https://doi.org/10.1007/978-3-030-96466-5_14, with permission from Springer Nature.
 - (3) A. Mitra, S. P. Mohanty, P. Corcoran, and E. Kougianos, “iFace: A Deepfake Resilient Digital Identification Framework for Smart Cities,” in Proceedings of the 7th IEEE International Symposium on Smart Electronic Systems (iSES), 2021, pp. 361–366, DOI: <https://doi.org/10.1109/iSES52644.2021.00090>, with permission from IEEE.
 - (4) A. Mitra, S. P. Mohanty, P. Corcoran, and E. Kougianos, “Detection of Deep-Morphed Deepfake Images to Make Robust Automatic Facial Recognition Systems,” in Proceedings of the OITS International Conference on Information Technology

(OCIT), 2021, pp. 149–154, DOI: <https://doi.org/10.1109/OCIT53463.2021.00039>, with permission from IEEE.

- (5) A. Mitra, D. Bigioi, S. P. Mohanty, P. Corcoran, and E. Kougianos, “iFace 1.1: A Proof-of-Concept of a Facial Authentication Based Digital ID for Smart Cities,” *IEEE Access Journal*, Vol. 10, 2022, pp. 71791–71804, DOI: <https://doi.org/10.1109/ACCESS.2022.3187686>. Article was originally published by IEEE under CC-BY license; authors retain copyright.
- (6) A. Mitra, A. Singhal, S. P. Mohanty, E. Kougianos, and C. Ray, “eCrop: A Novel Framework for Automatic Crop Damage Estimation in Smart Agriculture,” *Springer Nature Computer Science (SN-CS)*, Vol. 3, No. 4, July 2022, DOI: <https://doi.org/10.1007/s42979-022-01216-8>, with permission from Springer Nature.
- (7) A. Mitra, S. P. Mohanty, and E. Kougianos, “aGROdet: A Novel Framework for Plant Disease Detection and Leaf Damage Estimation,” in *Proceedings of the IFIP International Internet of Things Conference (IFIP-IoT)*, 2022, pp. 3–22, DOI: https://doi.org/10.1007/978-3-031-18872-5_1, with permission from Springer Nature.

CHAPTER 1

INTRODUCTION

In computing, the old adage “Garbage In, Garbage Out” has been around for quite some time. In the era of artificial intelligence, machine learning, and data quality, however, this aphorism is more applicable than ever.

1.1. Artificial Intelligence (AI)

1.1.1. The Beginning

In the first half of the 20th century, science fiction introduced the concept of artificially intelligent robots to the general public. It started with the “heartless” Tin Man in “The Wizard of Oz” and continued with the impersonating robot in “Metropolis” [30]. By 1950s, artificial intelligence (AI) was culturally accepted by a group of mathematicians, physicists, and philosophers of that time. British mathematician Alan Turing was one of such pioneers who explored the mathematical possibilities of AI in his 1950 paper “Computing Machinery and Intelligence” in which he described how to develop intelligent machines and how to test their intelligence [217]. However, the term “artificial intelligence” (AI) was coined by John McCarthy in 1956 [99].

From 1957 to 1974 AI grew rapidly due to the advancement of machine learning (ML) algorithms, the U.S. Defense Advanced Research Projects Agency (DARPA) funding, and improved domain specific knowledge [30]. But, lack of computation resources were still the barrier. Then, the first AI winter came. The field experienced disappointment and criticism. Lack of funding and reduced interests on AI research worsened the situation. It lasted from 1974 to 1980.

In the 1980s, the AI field experienced the second AI boom with the introduction of expert systems and the heavy funding from the Japanese Government in Fifth Generation Computer Project (FGCP) [30]. However, it did not long last due to the failure of the projects. Arrival of Second AI winter was obvious. AI research community experiences the second AI winter from 1987 to 1993.

In spite of the lack of government financing and public attention, AI again flourished. There were numerous notable achievements in AI between the 1990s and 2000s. Resource limitation was not a barrier any more. In some cases, Moore's Law excelled the prediction.

Rapid advancement in deep learning technologies along with government funding and monumental improvement in computing resources propelled the rise of AI. Today AI is everywhere.

1.1.2. Today and Tomorrow

We are now living in the "big data" era, which is defined as an era in which we have the ability to collect enormous amount of data through Internet-of-Things (IoT) that are too complicated for a single person to process. In this context, the application of artificial intelligence has already proven to be fairly beneficial in a number of different areas, including technology, banking, marketing, and entertainment, among many others [30].

Three key components: AI algorithms, computing resources, and data drive the AI revolution as in Fig. 1.1. Open source machine learning frameworks e.g., TensorFlow, PyTorch, Caffe2, Theano, high level API like Keras have made building of AI algorithms easy. These free, well documented and supported by an active community tools are excellent choice. Building machine learning applications has never been easier because of the availability of these tools. Computing resources in terms of CPU power, GPU, TPU, AI accelerators, cloud services or large data storage make the training of the models easy and possible. Finally, the huge amount of data collected from IoT devices, helps to achieve higher accuracy of the applications. However, this enormous amount of data has prompted an essential question: how important is the quality of these data [22] and how much data is needed for the best efficacy of a machine learning algorithm [199]?

1.2. Artificial Intelligence (AI)/ Machine Learning (ML)/ Deep Learning (DL), and Computer Vision (CV)

In today's business world, Artificial Intelligence(AI), Machine Learning(ML), and Deep Learning(DL) are the most discussed technologies because companies are using them

to build smart machines and apps. While artificial intelligence (AI) is the broad science of emulating human abilities, machine learning is a subset of AI that teaches a machine how to learn. Similarly deep learning (DL) is a subset of ML. It uses deep neural networks and learn the pattern of complex data. Fig. 1.2 shows the relation among these technologies.

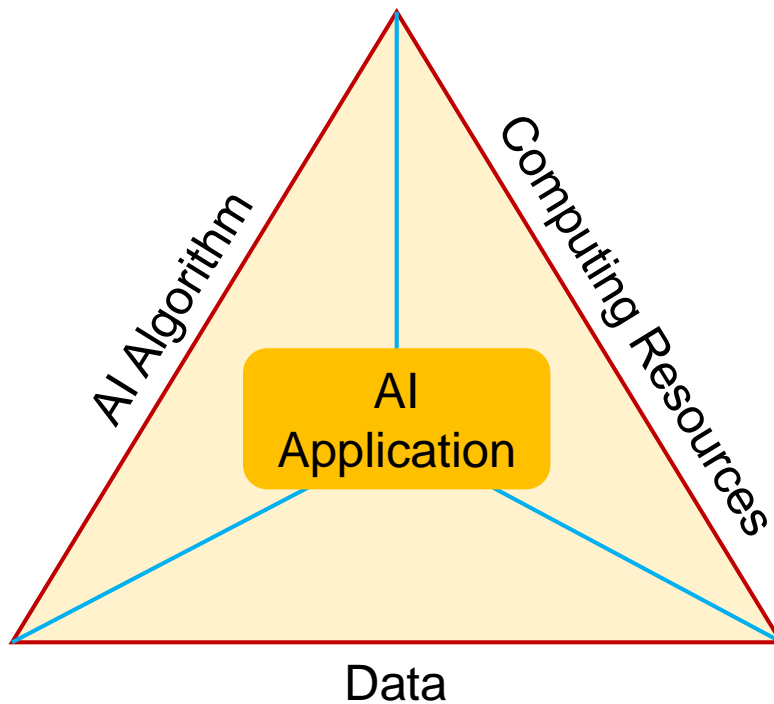


FIGURE 1.1. AI application triangle: key components of AI application.

In its most fundamental form, machine learning refers to the process of analyzing data with the use of algorithms, drawing conclusions or making forecasts based on the information gained from doing so, and then applying those findings to the real world. Therefore, rather than manually coding software routines with a specific set of instructions to complete a specific task, the machine is “trained” using large amounts of data and algorithms that give it the ability to learn how to perform the task. This allows the machine to accomplish the task without the need for hand-written instructions. The early pioneers of artificial intelligence were the ones who conceptualized machine learning, and some of the algorithmic approaches that were developed over the years to implement machine learning include decision tree learning, inductive logic programming, clustering, reinforcement learning, and

Bayesian networks. It is common knowledge that no one has succeeded in achieving the ultimate objective of general artificial intelligence (AI), and even the goal of narrow AI was mostly unattainable with the early methods of machine learning [58].

Another algorithmic approach that was popular among early adopters of machine learning is artificial neural networks, came and went for the most part over the course of several decades. Our knowledge of the biology that makes up our brains, including all of the connections that exist between the neurons, is the source of inspiration for neural networks [58]. However, in contrast to a biological brain, in which any neuron can connect to any other neuron within a given physical distance, these artificial neural networks consist of large number of layers, connections, and directions of data transmission. It uses enormous amount of data to learn.

On the other hand, the goal of computer vision is to endow computer systems with the ability to perceive images in a manner analogous to that of humans. It is an interdisciplinary field that focuses on the processing, analysis, and accurate interpretation of the visual environment we live in by computer systems. Computer vision (CV), for instance, enables computers to extract meaningful information from video and image files in the same manner that humans do [24]. ML and DL-based computer vision techniques have advanced visual data based problems.

In this dissertation, an overlapping region of ML, DL, and Computer Vision has been employed. The yellow region in Fig. 1.2 shows the area of this dissertation.

1.3. Data Quality Aspects

An algorithm's ability to learn is dependent on the quality and quantity of data it receives, as well as how much relevant information it contains. Data is considered valuable if it delivers meaningful information.

AI and ML have the potential to deliver astounding new perspectives. However, AI is unable to recognize the quality of the data or differentiate between reliable data and inaccurate data on its own during learning, and the algorithms that power AI operates under the assumption that the data being processed is accurate.

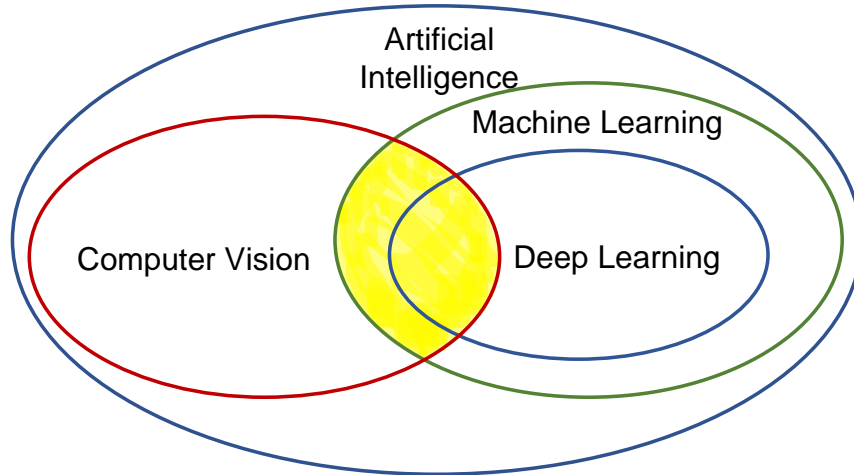


FIGURE 1.2. Relation among AI, ML, DL, and CV [6]. Yellow shaded area describes our area of work.

Bad data may ultimately lead to findings that are not actionable and do not provide any insights at all. There is even a bigger problem: inaccurate data can lead to conclusions that are not really helpful to anyone. Artificial intelligence systems can urge a corporation to take actions that are even more inefficient, in addition to the time and money that are spent by evaluating inaccurate data [22].

How much data is needed to train a successful machine learning model does not come with an analytical “equation,” but rather requires a series of parallel logical conclusions to determine whether it is feasible to drop data or augment data, and finally what information is necessary to train the model. The model should have been exposed to a variety of data points that represent a range of variation across the subspace of data while creating a machine learning model. With this, the model can easily adapt to new input data. In other words, the data needs to expose to enough knowledge or circumstances, so that the model can generalize when it is confronted with real-world data. Furthermore, because machine learning models can be used to a wide range of tasks, it is critical to understand the model’s core functionality [199].

There are many different ways to define data quality. However, we define “data quality” as the condition of qualitative and quantitative information in a dataset. Data

is thought to be high quality when they are suitable for its intended applications. The characteristics or aspects shown in Fig. 1.3 that define “data quality” are -

- Accuracy.
- Legitimacy so no counterfeit or forged.
- Accessibility and Availability or No Data Scarcity..
- Sufficiency.
- Fairness.
- Cleanliness.
- Reliability and Consistency.
- Relevance.
- Completeness and Comprehensiveness or No missing data.
- Granularity.

Among these different elements of data quality, three aspects have been investigated in this research. Different application domains where the selected aspects pose issues have been chosen.



FIGURE 1.3. Different data quality aspects.

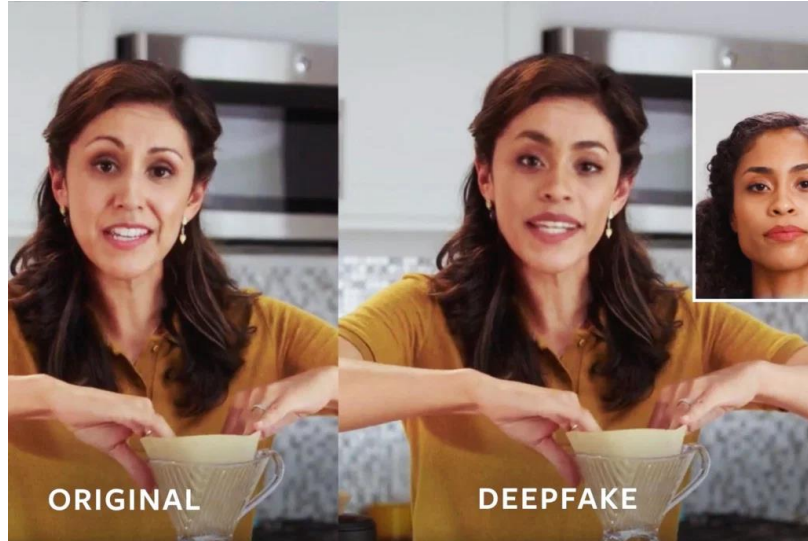


FIGURE 1.4. Deepfakes created by Facebook to fight against a disinformation disaster- source Facebook.

1.3.1. Deepfake: A Type of Fake Data

A new era of multimedia forgeries has begun because of advances in artificial intelligence. Deepfake, a Reddit user, created some fake videos in 2017 using a deep learning network. The use of convolution auto encoders [218] and generative adversarial networks [83] resulted in manipulated images/videos that are often visually indistinguishable from real ones. Forgery of multimedia content is now commonplace. Today, anyone can download smartphone apps that allow them to manipulate images. FaceApp, AgingBooth, Meitu, MSQRD, Reflect – Face Swap, Face Swap Live, and other applications are available. Deepfake videos/images are so well-made that it is hard to tell if they were edited [117, 253, 255]. Forged data changes the perception of truth. Deepfake takes it to a whole new level. The term was coined from the words “deep learning” and “fake”. Deepfake images/videos use deep learning technology to change a person’s face, emotion, or speech to that of someone else’s face, emotion, or speech [9, 11] as in Fig. 1.4. These deepfake images and videos are visually indistinguishable from real ones [255, 117, 253]. Cloud computing, public research AI algorithms, and copious data have made the ultimate storm to enable the democratization of deepfakes for distribution via social media platforms at a large scale [111].

1.3.1.1. Deepfake-A Social Issue

Deepfakes could be employed in art, expressiveness, accessibility, and business, but they are mostly used maliciously. “Deepfake” technology can distort reality unbelievably. This technology disrupts the truth. Most are not meant to be hilarious. They threaten individuals, businesses, society, and democracy [56, 121] and erode media confidence. Erosion of trust will foster factual relativism, unraveling democracy, and civil society. Deepfakes can help the least democratic and authoritarian governments prosper by using the “liar’s dividend,” where unpalatable truths are swiftly rejected as “fake news [111].” People no longer trust media news or images/videos. Political tension or violence can happen. It can destroy a politician’s career or a teen’s dream. Corporates are interested to avoid fraud or stock manipulation [236].

Deepfakes can hurt individuals and society, both intentionally and unintentionally. Deepfakes could deepen the worldwide post-truth dilemma since they are so lifelike, they mislead our sight and hearing. Putting words in someone else’s mouth, switching faces, and generating synthetic images and digital puppets of public personas are morally dubious behaviors that could hurt individuals and institutions. They can slander and infringe privacy [80, 183].

Insurgent groups and terrorist organizations can employ deepfakes to portray their opponents as making offensive remarks or engaging in provocative behaviors to stir up anti-state emotions. States can deploy computational propaganda against a minority community or another country[111].

Our face is who we are. People remember people by how they look. So, when fake images and videos are used, face manipulation is the one that is most often used.

1.3.1.2. Deepfake-Illegal and Unethical

Misusing someone’s voice and images without her consent is unethical and also illegal. Though synthetic data generation through deep learning is gaining popularity in the AI community as it solves the problem of data scarcity, it is ethically improper. Those synthetic images are also deepfakes and the images of real people are being used without their proper

consent. Additionally, if those images are created with ill intention, then bias, racism, color, segregation, and wrong ideologies can affect the created images. When those images are used to train an AI/ML/deep learning model for any decision making / prediction it predicts wrong, unethical, and biased way.

1.3.2. Data Scarcity

Data scarcity is another significant obstacle for a successful AI model. Deep learning relies on enormous amounts of data to quantitatively identify hidden patterns and connections; its effectiveness is primarily dependent on the amount of data available, and a lack of data will render deep learning approaches ineffective.

Training neural networks in areas such as image recognition and natural language processing (NLP) is easier. For NLP, datasets for training the neural networks e.g., BERT [66] is generally unlabeled. Similarly, for image recognition crowd-sourcing can be used e.g., ImageNet [62], COCO [141], and CIFAR [130] are widely used datasets.

However, the acquisition of a large labeled dataset for problems in the natural sciences is typically more difficult and often demands resource-, time- or labor-intensive computational or experimental efforts, making the data scarcity issue more apparent.

The IoT systems generate a huge amount of data through its network of sensors, cameras, or other “things”. However, these enormous amount of data do not always available to research community. The reasons are versatile -

- (1) Data are not publicly available.
- (2) Data are not annotated.
- (3) Privacy and security reasons.
- (4) Unfamiliarity about the technology options make domain specific people uninterested.
- (5) Passivity from the domains in turn lowers the efforts from the research community.

For example, artificial intelligence (AI) in agriculture is still in its budding stage. The lack of communication between the AI research community and farmers is creating bottle-

necks and communication gaps that are stalling the growth of research in that field. The data needed to train an AI/ML/DL model in agriculture and to transform agriculture into smart agriculture is not always publicly available or does not exist. As a result, agriculture does not fully embrace AI. Additionally, data privacy regulations are not standardized across the world. They vary across countries and continents, eventually making the accessibility of data harder.

1.3.3. Data Insufficiency

Data insufficiency is another barrier for high accuracy AI model. Sufficient amount of data boosts up the performance of an AI system whereas insufficient data leads to a less accurate model.

For example in agriculture IoT systems, there are certain areas in agriculture where datasets are available for research, such as plant disease [108], soil health [21], groundwater nitrate contamination [12], and disaster analysis [10]. However, the lack of sufficient datasets is one of the reasons for the gradual digital transition of traditional agriculture to smart agriculture.

1.4. Computing Platforms

A computing platform is an environment which provides the computation resources to develop, deploy, and manage software, models, and methods. Fig. 1.5 shows various computing platforms. Depending on the location of the platforms, three major computing platforms exist:

1.4.1. Cloud Computing Platform

In cloud computing, data storage, processing, diagnostic, and decision all the steps are done at a remote location and the application is accessed as service using internet. It has several financial and operational benefits. It is scalable and depending on the necessity its resource can be adjusted. However, Internet bandwidth saturation, latency, downtime, and data privacy regulations demand new computing paradigms.

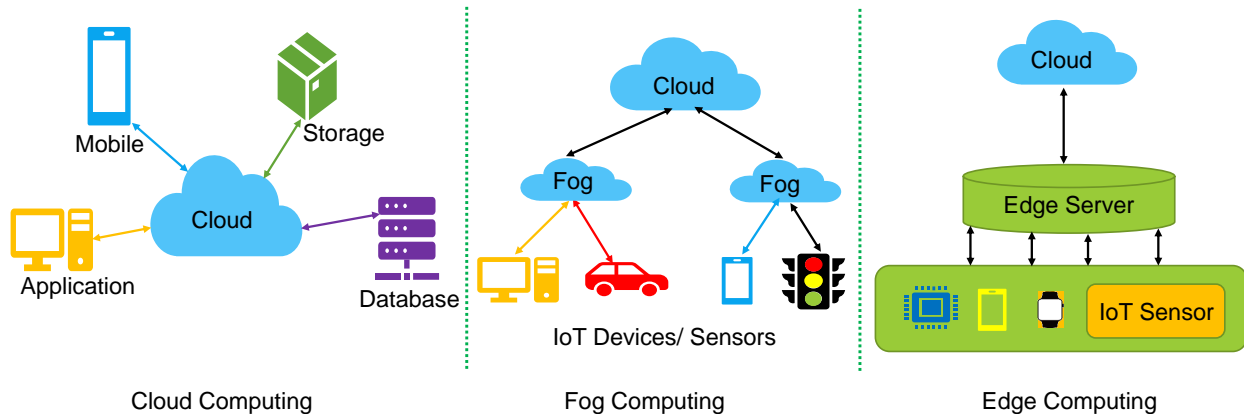


FIGURE 1.5. Different computing platforms.

1.4.2. Fog Computing Platform

Fog computing is used to bridge the gap between the traditional cloud-based data storage model and the distributed edge devices. The key distinction between fog and cloud computing is the fact that fog comprises of numerous fog nodes that form a local network. When information reaches the fog layer, the node can choose to handle it locally or send it to the cloud. Since some of the data is also saved locally, it can be accessible even when there is no network connection. As it is dependent on multiple links for transferring data from the edge to the cloud, it is prone to network failure.

1.4.3. Edge Computing Platform

Edge computing is a distributed computing platform. It puts enterprise applications closer to data sources like IoT devices or local edge servers. This proximity to data sources can provide faster insights, faster response times, and more bandwidth. When decision is time sensitive and real time solutions are needed, it can access locally stored data in absence of steady internet, process, and analyze them.

The time required to process data is reduced because no longer must it be sent to and received from a centralized server. This results in less strain on the network and faster processing times for data so less bandwidth and latency issue. It reduces overhead expenses and paves the way for the deployment of software in areas with spotty Internet service. By reducing the amount of time spent interacting with public cloud platforms and networks,

security is also improved. As data does not travel over a network to the cloud, it is mostly various network attacks free. Sensors, laptops, and cellphones are all examples of edge devices. It also solves the problem of data privacy and security and does not violate the data regulation policies as there is no traversal of data from the location they originate.

1.5. ML Methods for Data Quality Aspects in Edge Computing Platforms: Problems and Objectives

In recent years, ML models have become the primary method for solving a wide range of real-world issues in domains such as image recognition, healthcare, financial sectors, social media, speech recognition, self driving cars, automatic language translation, product recommendation, fraud detection, virtual personal assistant, email spam and malware filtering etc. These models show improved performance when trained on a large, high-quality, and versatile dataset. However, this is not the ideal scenario in real-life. There are several issues that can impact negatively for achieving high accuracy in ML methods.

1.5.1. Deepfake Data

1.5.1.1. Problem I

In everyday life, each event generates a series of data - audio, visual, numbers, texts, information etc. However most of the data are not recorded. So, even our life is a large data generator, we still lack a large dataset when it comes to ML model training. As a result, synthetic data generation becomes a common practice in ML training. Deep learning technology is being used to generate such data. However, these fake data are not always generated with good intention. The result is “deepfake”. It has been discussed in detail in Section 1.3.1. Deepfakes are such data that perturbs the truth. These inaccurate fake data must not be used in training a ML/DL model.

1.5.1.2. Objective I

Deepfakes and the inappropriate use of synthetic content offer a threat to the public that is undeniable, ongoing, and ever evolving in the areas of national security, law enforcement, the financial sector, and society. Hence, these fake data need to be detected.

1.5.1.3. Problem II

Deepfakes use AI/deep learning technology to alter a person's face, emotion, or speech to that of someone else's face, emotion, or speech. These deepfake images/videos/audios/texts are indistinguishable from real ones. Cloud computing, public research AI algorithms, and copious data have made the ultimate storm to enable the democratization of deepfakes for distribution via social media platforms at a large scale. Furthermore, these deepfake photos and videos are prevalent on social media platforms. So, we need a deepfake detection system that work on edge devices.

1.5.1.4. Objective II

Additionally, social media is the place where these deepfake images and videos are abundant. People primarily access social networking sites through their smartphones. Hence an edge friendly deepfake image/video detection system is required with a particular emphasis on social media images/videos.

1.5.2. Data Scarcity and Data Insufficiency

1.5.2.1. Problem III

Data scarcity or unavailability of data can be another bottleneck for achieving high accuracy in ML methods. Most ML/DL methods require large datasets to be trained on. Scarcity of training data can negatively impact the success rate of an AI model. However, the data required to train an AI/ML/DL model is not always publicly available or does not exist e.g., agriculture sector is not making full use of AI due to data scarcity or unavailability of sufficient dataset.

Furthermore, there is no uniformity in data privacy rules around the world. They differ from one country or continent to another, making it more difficult to gain access to information. It results in data scarcity in several areas of agriculture research. One of such area is crop damage by natural calamities. Additionally, an AI model learns better from a larger dataset or sufficient data. In reality, there are cases when there is no data scarcity but, enough data is not available.

1.5.2.2. Objective III

Data scarcity in agriculture stalls the application of AI/ML/DL in agriculture, it demands such AI/ML/DL methods that are trained on small datasets and still exhibit high accuracy and precision. Likewise, a typical ML or DL model exhibits high accuracy when there is adequate data available.

1.6. Main Contributions

In this section a brief overview of the significant contribution of this dissertation has been provided in terms of the Contributions I, II, III, and IV. Total 7 papers, mentioned in , have been discussed in these four chapters.

1.6.1. Contribution I-ML/DL Methods for Fake Data Detection on Edge Devices

To fulfill Objectives I and II, we propose efficient methods of deepfake video and image detection in Chapter 3.

1.6.1.1. Deepfake Video Detection-“A Machine Learning based Approach for Social Media Deepfake Video Detection through Key Video Frame Extraction”

In the last few years, with the advent of deepfake videos, image forgery has become a serious threat. In a deepfake video, a person’s face, emotion or speech are replaced by someone else’s face, different emotion or speech using deep learning technology. These videos are often so sophisticated that traces of manipulation are difficult to detect. These videos have a heavy social, political and emotional impact on individuals as well as on the society. Social media is the most common and serious target as it is a vulnerable platform susceptible to blackmailing or defaming a person.

Deepfake videos can be made in two ways: with autoencoders or with generative adversarial networks (GANs). Both are methods of deep learning. Inconsistencies in these videos are difficult to detect because they are not visible with the naked eye. We use textural artifacts of video frames in detecting deepfake videos.

First, we detected each video frame until a fake frame was identified. Then, instead of detecting each frame individually, we utilized a key video frame extraction technique and propose a hypothesis. We presume that -

Hypothesis: *While key frames contain all of a video’s information, they also carry the manipulation method’s visual artifacts.*

Consequently, our detection method is predicated on the identification of changes in visual artifacts in a frame which resulted due to deepfake falsification. This simple premise permits us to suggest a method that is both precise and computationally efficient. By isolating key video frames from each video, we decrease the quantity of data. It reduces the amount of video frames that must be examined for authenticity while retaining a high level of precision. During detection, we check the textural artifact changes for the key video frames. This reduces the amount of computations required.

Novelties: In contrast to the prior works that are computationally intensive, we offer a lightweight technique. The proposed approach has a reduced computing cost and can be implemented at the network’s edge. Memory limitations of the smart phones will not be a problem. The novel contribution of the work are as follows -

- An algorithm of lower complexity to detect deepfake videos.
- Our novelty here is to combine a well known method of computer vision to our simple classifier model for detecting deepfake videos. The existing works have very complex structures [169], [92], [132] for detection. Our main goal was to reduce the computational cost of detection without excessively sacrificing accuracy.
- Due to the lower computation and smaller model size of an efficient network, the model is edge friendly.
- Training of the model has been performed without a very large training dataset.
- For training and testing, we primarily used deepfake videos of Face Forensics++ (FF++ DF) [185] and Deepfake Detection Challenge (DFDC) [71] data sets. We used the compressed deepfake and original videos of the former. These compressed videos have two different compression levels - one is low loss and the other is high

loss. The data set is a good representation of social media scenarios. However, to obtain a better generalized model, we added DFDC dataset. We compressed the DFDC dataset at three compression levels by the H264 video compressor. Finally we trained our network with this mixed dataset.

1.6.1.2. Deepfake Image Detection-“EasyDeep: An IoT Friendly Robust Detection Method for GAN Generated Deepfake Images in Social Media”

GANs have improved the quality of the generated images [191, 42, 118, 85]. Presently, GAN approaches have accomplished monumental success in creating synthetic images [118, 117, 116, 227] and in transferring image styles between different domains [255]. Image-to-image translation can be used in changing season in a photo, photo enhancement, object transfiguration, etc. [255]. But, these applications can also be used in negative way. It is difficult for people to distinguish between a GAN generated deepfake image and real image with bare eyes. These fake images spread misinformation through social media or news channels.

A novel method has been presented for detecting deepfake images on an edge device in this paper. These images are GAN-generated. The method is machine learning-based and uses textural features of the images. Fig.1.6 shows the context of image forgery where the method detects fake images.

Novelties: The novelties of the work are as follows:

- GAN generated deepfake images have been detected in edge devices with lighter computational load.
- The training time for the ML model is much shorter. It was close to 30 minutes.
- The detection process is totally automatic due to the detection API.
- Success rate is high.

We compute the Gray Level Co-occurrence Matrix (GLCM) for four distances at $d = 1, 2, 3, 5$ and three angles $\theta = 0, \frac{\pi}{4}, \frac{\pi}{2}$. We then globally calculate five of Haralick’s textural features- contrast, dissimilarity, homogeneity, energy, and correlation- over the image frame

from the GLCMs to generate the feature vector. After the feature extraction, a machine learning algorithm is used to classify the image. As resources are limited in edge platform, we carefully choose our classifier as LightGBM with boosting type “Gradient Boosting Decision Tree (gbdt).”

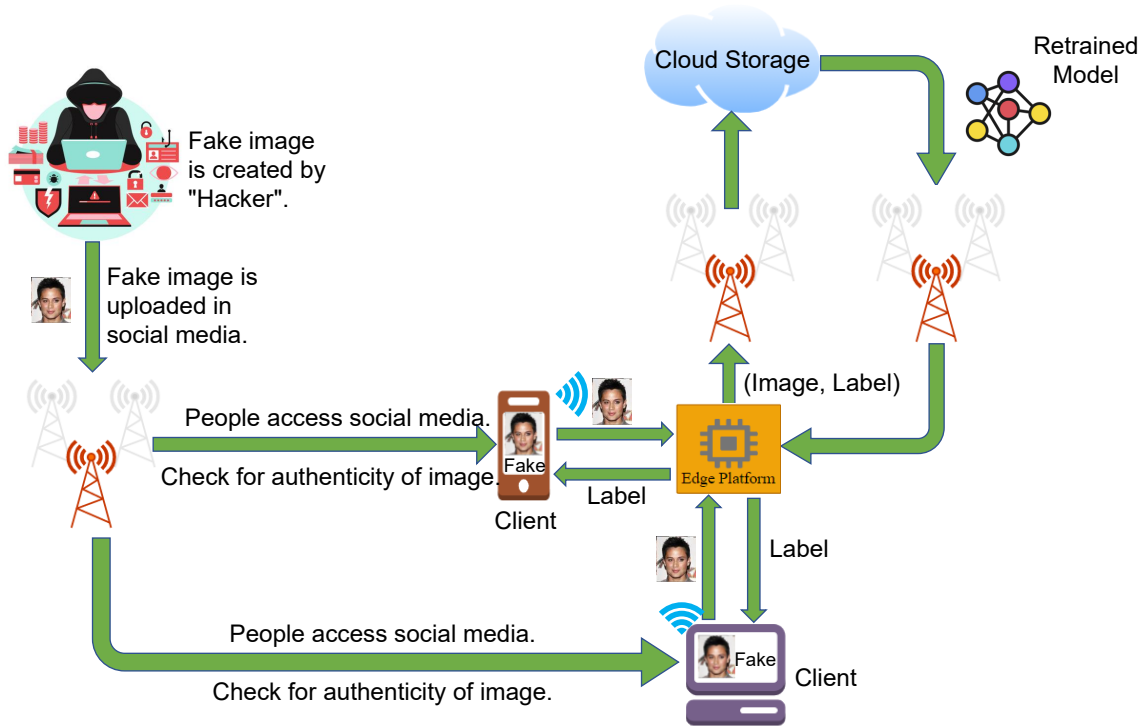


FIGURE 1.6. Context of EasyDeep.

1.6.2. Contribution II-A Deepfake Resilient Digital ID System for Smart Cities

We extend the Contribution-I to a scenario where deepfakes can cause serious problems. Contribution II provides the solution of this problem fulfilling Objective I.

1.6.2.1. iFace

“Smart Cities” are a viable option to various issues caused by accelerated urban growth. To make smart cities a reality, smart citizens need to be connected to the “Smart City” through a digital ID. A digital ID enables citizens to utilize smart city facilities health-care, transportation, finance, and energy with ease and efficiency. But, deepfake along with

various presentation attacks, poses a threat to the digital ID system. Here, we provide a deep learning based method for deepfake resilient digital id system of smart cities.

The use of digital IDs has already started to become the norm in various parts of the world, e.g., India’s *Aadhaar*, a biometric based identification number, the digital ID system in Australia, and the digital ID wallet initiative in the EU. Using digital ID in smart cities is a similar idea.

We propose a facial authentication-based end-to-end digital ID systems for smart cities. Facial authentication systems are prone to various biometric template attacks and cyber security attacks. Our proposed systems are designed to detect the first type of attack, especially deepfake.

iFace is in cloud-edge setting where encoded facial biometric data is sent from the cloud server to the edge device during authentication.

Novelties: The novel features of this method are as follows:

- The registration process is done at the edge, remotely and securely. The encoded bio key is stored at a cloud server.
- Authentication of the digital ID is also performed at the edge. As it is done near to the user, it is free of various indirect attacks.
- During authentication, a photo is taken at the edge and bio keys are generated with the help of the stored registration bio key. The system will not allow any tampering at this stage by checking for deepfake attack.
- Biometric data can vary at various photos taken at different times. It is not possible for people to keep their face in the same way all the time. Our method can accommodate a certain amount of modification of biometric data due to these changes.
- Only a neutral frontal face (NFF) is needed for the authentication. Our model is robust for faces with light makeup.
- We use face features like facial landmark points and certain distances in the eye and nose regions as the biometric of our proposed digital ID. Identification of the face is much more relevant in a smart city setting. Fingerprints are not always accessible

as our hands might not always be free. Behavioral traits are highly affected by the environment.

1.6.2.2. iFace 1.1

iFace 1.1 is an improved version of iFace, the biometric-based digital id. However, to prevent biometric data movement over the network iFace 1.1 has been proposed. The authentication process is performed at an edge device. Here, the novelty of our work is discussed:

- The user registration process is remotely and securely performed on the edge device. Facial embeddings are extracted using the user's device in an offline application. These embeddings contain no identifiable information and are private by design as they are just a series of numbers generated by a neural network. However, they are encrypted for extra protection, and sent to be stored on a remote cloud server operated by the smart city.
- The authentication process is also done on the edge device. The photograph, face embedding extraction, and authentication of the user are all done on the edge. No facial data is sent to the cloud during authentication. It makes the process resilient to various indirect attacks.
- Due to its fast and easy deepfake generation technique, FSGAN deepfakes pose a serious threat to people. Our system is capable of detecting deepfakes created by FSGAN.
- The proposed system can detect the presentation attack. A check is performed each time a user tries to access any facility within the smart city by taking a picture during the time of access. This ensures the correct user is present and decreases the risk of presentation attack. No data is stored anywhere in the system during authentication. It is performed as users come and go. With this feature, the risk of presentation attack on this kind of system goes down.
- No facial biometric data leaves the edge platform during the authentication phase. No data is stored during authentication. The only data that is stored is the reference

facial embeddings extracted during enrollment. These are used to retrain the model on the cloud, and it never leaves the cloud storage at any time.

1.6.3. Contribution III-DL-based Solution for Data Scarcity Problem on Edge Devices

A DL-based method, *eCrop*, has been proposed to fulfill Objective III in Chapter 5.

1.6.3.1. eCrop

Natural disasters impact agriculture. Farmers incur large losses due to crop damage. Climate/weather driven natural events or disasters are happening often and are causing billions of dollars in losses. Crop insurance provides economic stability to the agricultural industry to make up for losses. A crop insurance claim is an extensive process and it takes time to process claims. Recent AI developments spurred the application of machine learning or deep neural network-based models for numerous research issues. Deep neural networks need large training datasets to forecast accurately. However, no public dataset is available for crop damage estimation. In this paper, we present a novel crop damage detection method, built from a very small dataset. It is the core of grid-based crop damage estimation method *eCrop*. We present a proof-of-concept of the method, *eCrop* which is a part of our proposed agriculture cyber physical system. The crop damage detection method is a Convolutional Siamese Neural Network (CSNN) based model. A meta learning approach has been taken to train the model from a very small dataset. An accuracy of 92.86% has been achieved. Our *eCrop* method can be adapted to agricultural insurance claim processing to automatically estimate the crop damage. It is scalable to any size of the cropland and any type of crop. The novel contributions of this paper are the following:

- (1) The method requires very few good quality training data samples.
- (2) We propose our *eCrop* system as a part of an agro cyber physical system. It makes the process more efficient, adaptable, risk free, and resilient.
- (3) Our proposed *eCrop* system estimates the crop damage from the images taken by an Unmanned Aerial Vehicle (UAV). Hence, there is no need to set foot in the field for taking pictures. It essentially reduces the risk of more damage to the crop land.

- (4) Our method is applicable to any stage of crop growth and to any crop type.
- (5) We achieved high accuracy in detecting the damages caused by natural events.
- (6) Our proposed proof-of-concept estimates the overall crop damage precisely.
- (7) The data processing and computation is done at the edge server. Real time processing is also possible.
- (8) In general deep learning based methods need a large number of data sets for training but application of artificial intelligence (AI) in agriculture is not in a mature state yet. As a result, the required data is not always available which in turn poses a bottleneck to transform agriculture to smart agriculture [153, 223]. However, our method does not suffer from the unavailability of data issue. The model for crop damage detection has been trained with very few data. Our work can be a promising method for the researches in agriculture domain when large datasets are not available.
- (9) The method is scalable.

1.6.4. Contribution IV-Verification of the Effect of Data Insufficiency on Accuracy

Contribution IV is the extension of Contribution III. The amount of data required to achieve acceptable accuracy in a ML model has been studied along with a proposed solution for the problem. As the data scarcity problem is studied in the agriculture domain, a similar scenario has been chosen for this verification. However, in this example, sufficient data is available for the training and evaluation of the model.

1.6.4.1. aGROdet

Plant disease outbreaks are one of the major causes of crop damage, which is essentially one of the causes of food wastage. Hence, plant disease detection and damage estimation are important to prevent crop loss. By 2050, 60% more food will be required to feed a world population of 9.7 billion. Producing more food with traditional agriculture will stress the earth's limited natural resources. To avoid such a scenario, greener, sustainable, and modern agricultural practices should be followed. More efficient food production

along with a reduction of food wastage at different levels of the food supply chain will ease our ecosystem. Recent advances in AI prompted use of machine learning or deep neural network-based models for various research problems in different domains. To forecast effectively, deep neural networks require vast training datasets. In Chapter 5, a solution has been provided for a problem in agriculture where lack of data stalls the effective use of AI however in Chapter 6, how much data is sufficient for a high accuracy has been explored.

We propose a novel method, *aGROdet*, to detect plant disease and to estimate the leaf damage severity. The optimum data needed for an acceptable accuracy has also been studied. *aGROdet* is aimed at being implemented at the edge platform of IoT systems in the proposed Agriculture Cyber Physical System. A convolutional neural network-based model has been proposed to detect different plant diseases. The model has been trained with large publicly available datasets. More than 97% accuracy has been achieved in the initial phase of the experiment. A pixel-based thresholding method has been used for estimating the severity of the damage. Damage estimation limiting factors, such as on the leaf and around the leaf shadows, have also been addressed.

Significance of aGROdet in a Smart Village Context:

Today, close to 3.4 billion people live in rural areas. The majority of villages lack technology, innovation, energy, and industry even today. However, the modernization of villages with Internet connectivity, smart agriculture, smart healthcare, smart grid, and education is required. A holistic approach is needed for rural areas to ensure the sustainable development of society. To implement that goal, various smart village movements have recently emerged across the globe in various sectors. For example, Fig. 1.7 shows the smart energy project sites of IEEE Smart Village initiatives [13].

The application of heterogeneous technologies centered on the Internet-of-Things (IoT) can shape rural areas as smart villages [61]. As the financial backbone of the smart village is agriculture industry, it is one of the most important areas of research for smart villages. To transform the traditional agriculture to an efficient, sustainable, and green agriculture, digital transformation is the key. In this context, our proposed method aGROdet is



FIGURE 1.7. IEEE smart village map for smart energy projects [13].

appropriate.

- Plant disease is a major challenge for sustainable agriculture. It is a nightmare for farmers as disease can destroy the plants and cause huge losses. The common method of plant disease detection in developing countries even today is manual observation. It is an arduous process. It needs expertise, and the service is so expensive that it is not always affordable for farmers [201]. In such a scenario, the farmer can have an overall idea of the disease and its severity through the proposed method, *aGROdet*.
- It automatically and accurately detects plant diseases and estimates damage. Significantly less effort is needed from the farmers' perspective to use *aGROdet*. It is accessible through a mobile app. To get the results, farmers only need to take a photo of the diseased leaf. The rest of the process is automatic.
- It is an edge-based Internet-of-Agro-Things (IoAT) method that can detect plant disease and estimate the damage even without an Internet connection. If an Internet connection is not available for any reason, the damage estimation procedure will not

be affected. An Internet connection is used to store data in the cloud. This stored data is used for future training of the model.

- This is a very useful tool for farmers who can detect plant diseases with an estimation of plant damage on their own. No expert knowledge is required.
- We hope that *aGROdet* will help farmers take proper control measures and save time, money, and secondary plant losses.

1.7. Dissertation Organization

The rest of this dissertation is organized as follows: Chapter 2 presents the related research, especially in the same application areas: deepfake detection, solution for crop damage estimation, and plant disease detection and damage estimation. Chapter 3 presents fake data detection methods, especially the solutions for deepfake video and image detection at edge devices. Chapter 4 describes a real-world application area: smart city facility access via a deepfake resilient digital ID system. Chapter 5 proposes solution for the application when data scarcity poses a bottleneck. The impact of data insufficiency has been studied, and how sufficient data leads to proposing a successful model for a problem, has also been presented in Chapter 6. Finally, the dissertation concludes with future research directions for the work in Chapter 7.

CHAPTER 2

RELATED PRIOR WORKS

In this chapter, we discuss a detailed literature survey that motivate us to propose ML methods for data quality aspects related problems in edge computing platforms. Following are discussions of publications relating to the application areas for which we supplied solutions or validated methodologies [156, 104, 157, 154, 159, 158]. Outline of the chapter is depicted in Fig. 2.1.

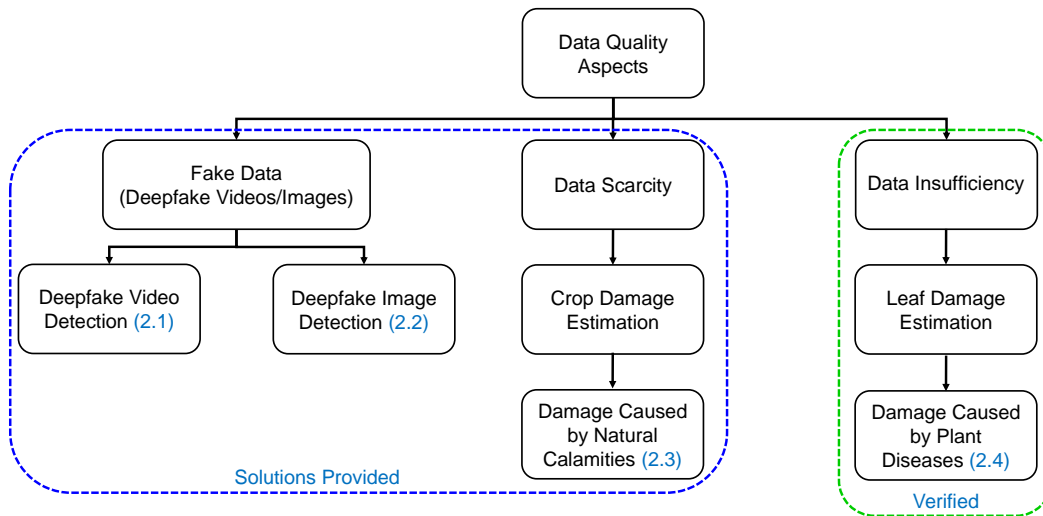


FIGURE 2.1. Outline of the current chapter.

2.1. Deepfake Video Detection

Identifying manipulated and falsified contents is technically demanding and challenging. In the past two decades in media forensics, a lot of work to detect image and video forgery has been done. Most of the solutions proposed for video forensics are for easy manipulations such as copy-move manipulation [60], dropped or duplicated frames [81], or varying interpolation [68]. But use of auto-encoders or generative adversarial networks made image/video forgery sophisticated. These computer generated forged videos are hard to detect with previously existing detection techniques. Stacked auto-encoders, convolutional neural

network (CNN), Long Short Term Memory (LSTM) network, or generative adversarial network (GAN) in detection models have been explored to detect video manipulation. Some of the existing works are summarized in Table 2.1.

Among deep learning solutions, some are temporal feature based and some are based on visual artifacts. In visual-artifact based works, videos are processed frame-by-frame. Each frame contains different features which generate various inconsistencies in the manipulated region of an image. These features are first extracted and then used as input to a deep learning classifier as CNN models can detect these artifacts. The classifiers are ResNet152 [93], VGG16 [200], Inception V3[210], DenseNet [106] etc.

Certain works are associated with detection techniques based on eye blinking rate [136], noting the difference between head pose [246] of an original video and a fake video, and detecting the artifacts of eyes, teeth and face [150]. Human blinking pattern has also been used in another recent paper[115]. A general capsule network based method has been proposed to detect manipulated images and videos [169]. A VGG19 [200] network has been used for latent feature extraction along with a capsule network to detect different spoofs, replay attacks etc. Two inception modules along with two classic convolution layers followed by maxpooling layers have been explored in [27]. This approach is at a mesoscopic level. Audio and video parts of a video clip have been used in getting emotion embedding to detect fake videos too [160]. A comparative study among different approaches have been shown in [127] where the authors evaluated some existing techniques.

Other than the visual artifact based works, there is another parallel type of work that is prevalent. These are based on temporal features of a video. A combined network of a CNN and LSTM architectures has been explored [84]. The CNN module extracts features from the input image sequence. The feature vector is fed into the LSTM network. Here, the long-short term memory generates a sequence detector from the feature vector. Finally, a fully connected layer classifies the video as manipulated or real. Another combined network was used in a different paper to classify forged videos [189]. A DenseNet structure combined with recurrent neural network has been used. A blockchain based approach to detect forged

videos [90] has been proposed by Hasan and Salah. Each video is linked to a smart contract and it has a hierarchical relation to its child video. A video is called pristine if the original smart contract is traced. Unique hashes help to store the data to IPFS. This model is claimed to be extendable to audio or images. In [198], the ownership of a video has been stated by detecting fake video and distinguishing it from real video. Spatio-temporal features has also been used in detecting deepfake video [78].

In another recent work [92], deepfake video has been detected using Convolution-LSTM network. Visual counterfeits have been analyzed. A triplet architecture has been used in detecting deepfake videos at high compression levels [132]. Sharp multiple instance learning has lately been used in detecting partial face attack in deepfake videos[135]. Performance of the detectors has been improved by clustering face and non-face images and removing the latter [45].

In [238], a combination of VGG architecture and Vision Transformer has been used to detect deepfake videos. A considerable accuracy has been obtained. Vision Transformer has also been used in [57] along with EfficientNet B0. Knowledge distillation and representation learning have been utilized in [122]. In this paper special attention has been provided to avoid catastrophic forgetting. A fine grain classification approach has been taken in [252] and a multi attention detection network has been proposed. Face warping artifacts have been used in [138] to detect deepfake videos.

These are some of the works which are focused on deepfake video detection. In the next section, some additional work concentrated on deepfake image detection have been described.

TABLE 2.1. A Comparative Perspective with Existing Works on Deepfake Video Detection.

Works	DataSet	Model Features	Remarks
Sabir et al. [189]	FF++	Feature:Spatio-temporal features of video streams. Network: Bidirectional RNN + DenseNet/ResNet50.	Not applicable to long video clips. Not trained on a large dataset.
Güera and Delp [84]	HOHA	Features: Temporal inconsistencies of deepfake video. Network: Inception-V3 + LSTM.	Didn't take into account of compressed videos.
Li et al. [136]	CEW	Features: Eye blinking rate. Network: VGG16 + LSTM + FC.	Applied to uncompressed videos.
Afchar et al. [27]	Internet	Features: Mesoscopic. Network: Mesonet structures - Meso-4 and MesoInception-4.	Not used any public dataset. So what types of deepfake videos are not clearly mentioned.
Li et al. [138]	UADFV and DeepfakeTIMIT	Features: Face warping artifacts. Resolution inconsistency between the warped face area and face. Network: 4 CNN models.	Compression has not been considered.
Matern et al. [150]	Various sources.	Features: Facial texture difference and missing details in eye and teeth. Network: Logistic regression model and neural network.	Not for compressed video.

Table 2.1 Continued from Previous Page.

Works	DataSet	Model Features	Remarks
Nguyen et al. [169]	Four major datasets.	VGG-19 + Capsule Network.	Accuracy is low for highly compressed data.
Hashmi et al. [92]	DFDC	Features: Used facial landmarks and convolutional features. Network: CNN+LSTM	Computation complexity is high. Minimum video length is 10 seconds. Works well for long videos.
Kumar et al. [132]	FF++ + Celeb-DF	Features: Metric learning approach. Network: Triplet Architecture.	For highly compressed video.
Mitra et al. [155]	FF++	Features: Face Artifacts. Network: XceptionNet + Classifier Network	For compressed video. High Accuracy.
Mitra et al. [156]	DFDC and FF++	Features: Face Artifacts + Key Video Frame. Network: XceptionNet + Classifier Network.	For any level of compressed video. Applicable to any video even with only one key video frame. Faster with less computation
FF++ → FaceForensics++;			HOHA → Hollywood-2 Human Actions;
CEW → Closed Eyes in the Wild;			DFDC → Deepfakes Detection Challenge;

2.2. Deepfake Image Detection

In the last decade, due to the availability of GPUs, the research on GAN generated images has received a huge boost. Various areas of image manipulation such as high quality GAN generated images [152, 191, 31, 85, 118, 179, 134, 235], image-to-image translation [110, 248, 144, 256, 50], face completion [137, 109], various facial expression and attributes [70, 145], domain transfer [211, 123], and style transfer [116, 117] have received the attention of the computer vision community.

Researchers in image forensics and computer vision have been working to develop methods in detecting those GAN generated fake images. In our [155, 156] papers, deepfake videos have been detected and special attention was given to social media compressed videos. Deepfake images have been detected in smart city context in [102, 157]. An ensemble deep learning technique via a Random Forest classifier has been proposed in [96]. Three shallow CNN structures have been used to extract features from the images in YCbCr, HSV, and Lab color spaces. Two fully connected (FC) networks with 2048 and 1024 nodes increase the total number of trainable parameters. Parallel processing of the same image in three different color spaces makes the process resource intensive. Fake faces have been detected in [234] using a shallow neural network as a classifier and neuron activation has been monitored using deep neural network. The model has been evaluated for four perturbation attacks. This model is also resource intensive. Some of the textural properties of fake images generated by StyleGAN and PGGAN have been explored and Gram blocks have been added in ResNet structure in detecting fake faces in [147]. It is a heavy weight structure. Gray level co-occurrence matrices (GLCM) have been calculated separately on RGB channels and used as the inputs of a DNN structure [167]. GLCM calculation over three channels for an image makes it memory intensive. No effort has been made to deploy it at edge either. Both GAN generated and man made fake images have been detected in a DNN based ensemble network [216].

How a GAN generated image is different from a camera shot image from a color cue perspective, has been investigated in [151]. GAN fingerprints on image attribution have been

noted in [249]. All these works claim to have high accuracy, but no effort has been made to deploy them in IoT environments.

In last five years, deepfakes detection have been a hot topic. Researchers in image forensics and computer vision have been working to develop methods to detecting deepfake videos and images. However, from the above discussion, it is evident that there is not much work done for compressed videos which are predominantly used in social media or detection methods focused on edge computing platforms.

2.3. Crop Damage Estimation - Damage Caused by Natural Calamities

In this section, some of the relevant papers which assess crop damages by different natural causes are discussed. Most of the papers present specific type of damage such as heat, frost, hail, storm, flood, pest, or crop diseases. An unsupervised machine learning method has been used to detect hail damage using remote sensing data in [202]. Various indices have been calculated pre- and post-hail. K-means clustering has been used to determine the damage zone.

How drought affects the cropland has been studied in [126]. The relation between Normalized Difference Vegetation Index (NDVI) and Land Surface Temperature has been studied in the areas of southeastern Germany from 20 years data. Different drought indices have been calculated with soil water content and crop yield discrepancies.

Some papers also address crop damage by floods, storms, crop diseases, and wild animals attacks. Flood damage has been assessed in [55]. A simulated flood is generated using rainfall-runoff-inundation model. The water depth and the period for the flood have been considered to assess the damage of rice crops in the Stung Sen River basin of Cambodia.

Frost damage in lemons has been detected in [171]. Electrochemical resistance has been measured and different values of impedance were obtained for natural and freeze-thawed lemons. Statistical procedures have been used to differentiate between them. A prediction model has been presented using an artificial neural network.

Crop damage by wild animals has been presented in [131]. A photogrammetric reconstruction method has been used to segment the damaged part automatically. UAV taken

images have been used to evaluate the method along with satellite data. Another work [188], also suggested taking pictures by UAVs and assessing crop damage by wild boars. A Random Forest classifier has been used to estimate the damaged land and corresponding loss. Geographic Object-Based Image Analysis has also been used. In [195], a Random Forest classifier has also been used to detect cyclone, earthquake, hail storms, and flood damage. Sentinel I and II satellite data has been used. To estimate crop damage, crop layers along with NDVI have been considered. UAV images have also been used to detect crop damage by insects in [178]. K-means clustering algorithm has been used to detect the damage in an unsupervised learning way. Gaussian convolutional kernels help to reduce the high frequency noise.

Crop disease has been identified using convolutional neural networks in [225] whereas [67] addresses a more general approach. Disaster vegetation damage index (DVDI) has been utilized with MODIS images and USDA/NASS data to estimate crop damage. The damages are not specific to a cause. In [44] a different perspective has been considered. Here, the authors studied the feasibility of smart phone based photos for insurance processing. These photos are taken by the phones of the farmers. This approach has been suggested to reduce the cost.

Additional research works are discussed in Table. 2.2 along with our proposed *eCrop* method. These papers also address specific types of damages. A majority of the papers use satellite images.

TABLE 2.2. A Comparative Perspective of Existing Works with eCrop.

Works	Year	Cause of Dam- age	Method	Remark
Sosa et al. [202]	2021	Hail	Unsupervised learning with K-means clustering + Satellite Images	Specific to one type of damage.
Kwak et al. [133]	2015	Flood	Flood depth and duration + MODIS time series images	Specific to damage type
Bell et al. [36]	2019	Storm, Hail and damaging winds	Synthetic Aperture Radar (SAR) + MODIS image + NOAA/NWS severe weather reports	Specific to damage type
Di et al. [67]	2018	Natural Disaster	Disaster vegetation damage index (DVDI) + MODIS images + USDA/NASS data	Provides overall estimation
Sawant et al. [195]	2019	Cyclone, earthquakes, hail storms, and flood	Random forest classifier + Min. and Max. NDVI + Sentinel 1 and 2 data	Specific to damage type
Yang et al. [245]	2019	Cold	Hyper spectral image + Convolutional neural network	Specific to damage type

Table 2.2 Continued from Previous Page.

Works	Year	Cause of Damage	Method	Remark
Hsuan et al. [105]	2018	Fluctuating weather, heavy rain fall and typhoon	UAV aerial images + NDVI calculation	Focused on specific damage type
Pallagani et al. [174]	2019	Crop disease	CNN + Plant Village dataset	Specific to one damage type
Udotalapally et al. [225]	2020	Crop disease	CNN + Image Transformation + Plant Village dataset	Specific to one type of damage
eCrop	2022	Any natural causes like heat, frost, diseases and insect	Convolutional Siamese Network + Contrastive Loss + Few Data	Covers all damage type under MPCII. Scalable to any type of crop.
MODIS → Moderate Resolution Imaging Spectroradiometer		STRM → Shuttle Radar Topography Mission		
NDVI → Normalized Difference Vegetation Index		NOAA → National Oceanic and Atmospheric Administration		
NWS → National Weather Service				

2.4. Leaf Damage Estimation - Damage Caused by Plant Diseases

During literature review, two types of papers addressing plant or crop diseases stand out: the first addresses multi-crop disease solutions, while the second focuses on a specific crop or plant type. In the last decade, mostly traditional image processing algorithms and hand-picked features with machine learning (ML) classifiers have been used to detect plant and crop diseases. Those approaches have their own difficulties, along with not so great accuracy [143]. In recent studies, mostly computer vision-based methods with deep learning networks are being proposed for this purpose. The use of deep learning networks, mostly convolutional neural network (CNN)-based approaches, makes the disease identification automatic, reduces manual intervention, and performs better in detecting plant diseases.

Complex features are obtained automatically in deep learning network-based solutions via various layers and types of neural networks, particularly CNN. Different CNNs have been used for different purposes, such as feature extractor [91], classification network [219], and disease localization network [237].

2.4.1. Single Plant Crop Diseases Detection

Non-parametric ML classifiers are used in various works, along with the recent trend of deep learning networks for detecting plant/crop diseases. For example, the K-means algorithm is used in [166] for paddy leaf diseases. Several studies have been conducted on cotton diseases. The K-nearest neighbors (KNN) algorithm has been used in [176] for cotton leaf diseases. *Ramularia* leaf blight cotton disease has been identified using non-parametric classifiers from multi-spectral imagery of an UAV in [242]. A decision tree classifier has been used for detecting cotton crop diseases [53]. Cotton leaf spot disease has been detected in [39] using Support Vector Machines (SVM). Cucumber's powdery mildew has been segmented using U-Net at pixel level with high accuracy in [140].

A combination of InceptionV3 and ResNet50 networks has been used to identify grape leaf diseases with 98.57% testing accuracy [113]. A shallow 3D CNN structure has been used on hyperspectral images to identify a soil-borne fungal disease, charcoal rot, for soybean [165]. An improved AlexNet model has been used to identify fragrant pear diseases

and insect pests [231]. A typical accuracy of 96.26% has been achieved. In [173], a Faster RCNN has been used to detect sugar beet leaf spot disease with 95.48% accuracy. Northern maize leaf blight detection has been done in [206] using multi-scale feature fusion method with improved SSD. Mask R-CNN has also been used to segment UAV images in [205] for northern maize leaf blight detection. In [38], a YOLOv3 network was used to detect pests and diseases in tea leaves. Using SegNet, four categories of grape vine diseases have been identified in [120] from UAV images.

2.4.2. Multi Plants Crops Diseases Detection

Deep learning techniques are popular in the research community for multi-plant detection. A convolution neural network-based Teacher-Student network has been utilized to detect plant diseases [40]. A sharper visualization of the diseased leaf has been achieved with the PlantVillage dataset [108]. Another deep convolution neural network-based on GoogleNet and AlexNet has been used to detect crop diseases with 99.35% accuracy [163] using the earlier mentioned dataset. In [190], Single Shot MultiBox Detector (SSD) model has been chosen among three different deep learning models for plant disease detection. It shows 73.07% mean average precision (mAP) with the Adam optimizer on the Plant Village dataset. In [112] severity of crop leaf disease has been estimated along with crop type and crop disease prediction with an 86.70% accuracy using binary relevance (BR) multi-label learning algorithm and Convolutional Neural Network. Another CNN-based structure, built from a ResNet50 network with shuffle units, has been used to detect plant disease and estimate the severity of the disease in [139] with an accuracy of 91%, 99%, and 98% for disease severity, plant type, and plant disease classification, respectively. In [76] several networks have been tested and finally an accuracy of 99.53% in identifying plant disease has been achieved. Disease prediction has also been done along with crop selection and irrigation [225]. In this work, a CNN-based plant disease detection network has achieved an accuracy of 99.25%.

From the above discussion, it is clear that the majority of papers address various diseases for different plants or crops. However, it is highly important to estimate the disease-

related damage. Without that knowledge, plant disease management and prevention is not possible.

2.5. Discussions

It is evident that the solutions provided in prior research cover variety of aspects of these problems however several issues have not been addressed.

- Majority of the solutions in Section 2.1 and 2.2 do not address social media compressed deepfake videos/images at large scale.
- Detection methods in Section 2.1 and 2.2 are not focused for edge computing platforms.
- Most of the work in Section 2.1 and 2.2 are based on heavy computation.
- For crop damage estimation in Section 2.3 majority of the work address a specific damage type. Not all the natural calamities are handled by the same method.
- In agriculture even if the use of IoT sensors generate a huge amount of data, still public dataset are unavailable. Data scarcity makes the field difficult to work on for the academic researchers. Papers that estimate the crop damage have not considered the data scarcity part or propose any method for data scarce situation.
- Most of the paper in Section 2.4 identify the disease type instead of the damage estimation. However damage estimation is necessary to prevent the plant disease. Data scarcity is not a issue in this problem. However, insufficient data can cause less accurate model.

CHAPTER 3

DATA FORGERY: DETECTION OF DEEPPFAKE VIDEOS AND IMAGES IN SOCIAL MEDIA

This chapter presents two machine learning methods - one for detecting social media deepfake videos [156] and the other for deepfake images at edge devices [104].

3.1. Introduction

In today's world, social media/networks play an important role. They can have an impact on someone's mental health [54], social standing (albeit this should not be the case), and other factors [32, 29]. People can shoot images or films with their mobile or small cameras anywhere, at any time. They can easily connect to the rest of the world via a simple handheld gadget. Anyone may make fraudulent images/videos using commercial picture editing software[11, 7]. Deepfake technology is accelerating such trend. In deepfake videos, the real person is not present. From thousands of target's video snippets or photos deepfake technology synthesizes the videos.

These manipulated photos and videos are frequently shared on social media. Impersonating someone on social media is against the law. On order to preserve our privacy and identity, we need some countermeasures, particularly in social media where a person is at risk [8]. However, there is a disconnect between how technology is developing and how scholars and businesses are attempting to resist it. As a result, new advanced deepfake videos are being shared on social media sites such as Facebook, Twitter, and Instagram on a daily basis. The US government's Defense Advanced Research Projects Agency (DARPA) is working with a number of organizations to combat picture fraud, including deepfake [23]. Facebook, Google, Microsoft, and Amazon are all involved in this fight.

3.2. Deepfakes: Threat Environment

Our face is a representation of who we are. People remember people based on their appearance. Face modification becomes the most focused when image and video forgery are

involved. Face forgery in multimedia has skyrocketed in the previous two decades.

Among the reported works, an image based approach was used in [41] to generate a video. Face replacement of the actor without changing the expression was presented in [79]. Real time expression transfer in [220] is also important. Work on lip syncing of President Obama helped [209] people to understand how serious is video forgery.

With deepfake technology, the ability to distort reality has surpassed acceptable limits. This disruptive technological change has an impact on the truth. Many are meant to be amusing, but others are not.

- (1) “Deepfake” technology can distort reality unbelievably. This technology disrupts the truth. They threaten individuals, businesses, society, and democracy and erode media confidence. Erosion of trust will foster factual relativism, unraveling democracy, and civil society. Deepfakes can help the least democratic and authoritarian governments prosper by using the “liar’s dividend,” where unpalatable truths are swiftly rejected as “fake news.” People no longer trust media news or images/videos. Political tension or violence can happen.
- (2) Non-consensual pornography proliferated deepfake content and currently represents most AI-enabled synthetic content in the wild.
- (3) Deepfakes can hurt individuals and society, both intentionally and unintentionally. Deepfakes could deepen the worldwide post-truth dilemma since they are so lifelike, they mislead our sight and hearing. Putting words in someone else’s mouth, switching faces, and generating synthetic images and digital puppets of public personas are morally dubious behaviors that could hurt individuals and institutions.
- (4) Deepfake is used to misrepresent a company’s product, executives, and brand. This technique is aimed to harm a company’s market position, manipulate the market, unfairly diminish competition, hurt a competitor’s stock price, or target mergers and acquisitions. The corporate world is eager to protect their businesses from fraud or stock manipulation, according to [236].
- (5) They can slander and infringe privacy. They can depict a person in such a scenario

that would affect her reputation or social standing.

- (6) They may endanger national security, democracy, and an individual's identity [56, 121].
- (7) Insurgent groups and terrorist organizations can employ deepfakes to portray their opponents as making offensive remarks or engaging in provocative behaviors to stir up anti-state emotions.
- (8) States can deploy computational propaganda against a minority community or another country.
- (9) Deepfake audio/video can influence an election by spreading lies.
- (10) Impersonation is another area where deepfake plays a significant role. In today's connected world, when people access various facilities through internet, they can be victims of deepfakes. A deepfake video can be used to defame someone and invade their privacy [183, 80].
- (11) Misusing someone's voice and images without her consent is unethical and illegal. Though synthetic data generation through deep learning is gaining popularity in the AI community as it solves the problem of data scarcity, it is ethically improper. Those synthetic images are also deepfakes and the images of real people are being used without their proper consent.
- (12) Additionally, deepfake can be created with ill intention, bias, racism, color, segregation, and wrong ideologies. If these deepfakes are used to train an AI/ML/deep learning model for any decision making, wrong and biased predictions may be generated.

To prevent the catastrophic consequences of deepfake videos, Facebook, Microsoft, AWS, the Partnership on AI, and some academic institutions organized the Deepfake Detection Challenge and began building the Deepfake Detection Challenge (DFDC) dataset for research purposes. Google announced another dataset for deepfake video detection, FaceForensics++, in collaboration with Jigsaw. These are the two datasets available for deepfake video detection.

3.3. Why are Deepfakes Hard to Detect?

Deepfake videos can be made in two ways: with autoencoders or with generative adversarial networks (GANs). Both are methods of deep learning. Inconsistencies in these videos are difficult to detect because they are not visible with the naked eye.

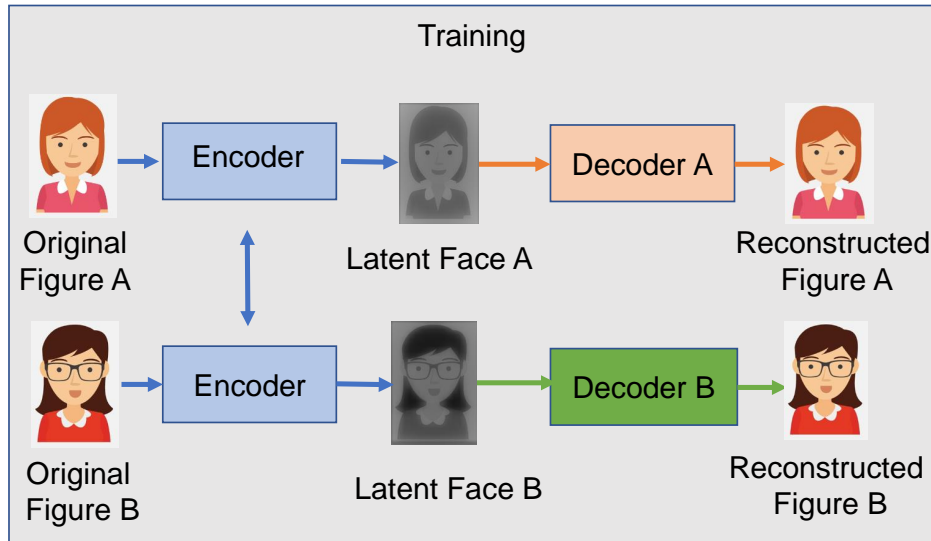
3.3.1. By Autoencoders

The process of creating a deepfake video is divided into three stages: extraction, training, and creation. The extraction process extracts all frames from a video clip and identifies and aligns faces. The term “auto-encoder” refers to a device that combines the functions of an encoder and a decoder. When an encoder receives an image as input, it creates a latent face or base vector of lower dimension. This vector is then passed to the auto-decoder, encoder’s which reconstructs the input image. The network’s shape, such as the number of layers and nodes, dictates the image’s quality. Weights are used to store the network’s description. Fig. 1(a) illustrates the training stage. These weights are optimized during training. Two sets of autoencoders are required to create a deepfake video. One is for the source video’s original face, and the other is for the target video’s target video. Both encoders share weights during training in order to have the same features on the source and target faces. There are two distinct decoders for the two image sets. Because a shared encoder creates common features for both image sets, the encoder automatically learns common features, which explains the name. After training is complete, the Decoder B receives a latent face from image A. As a result, Decoder B attempts to recreate image B using only the information contained in image A’s relative information. The process of creating a deepfake video frame is illustrated in Fig. 1(b). This is repeated for all frames or images in order to create a deepfake video.

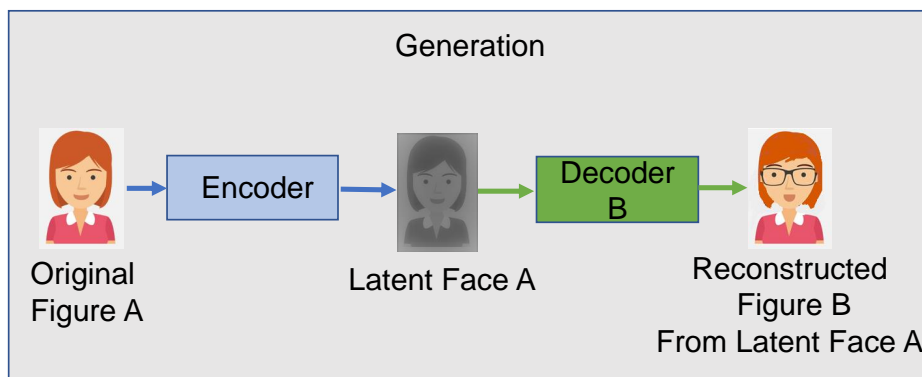
3.3.2. By Generative Adversarial Networks(GAN)

While creating deepfake videos as in Fig. 3.2 with a GAN is popular, training a GAN is challenging. GANs are made up of two neural networks. They are known as the generator (G) and discriminator (D). G serves as the autoencoder’s encoder. However, because G and D

are playing a minmax game, G is constantly attempting to outperform D by creating a better image. After adequate training, the generator produces images of such high quality that the discriminator is unable to tell the difference between real and fake images. Though the basic structure of GANs are same, they differ in overall architecture and working principles.



(a) Training phase.



(b) Generation phase.

FIGURE 3.1. Deepfake creation by autoencoder.

3.4. Addressed Research Problem

When a digital camera takes a photograph, light from the object passes through the lens and falls on the CMOS sensors, which divide the image into pixels after measuring the

light intensity and brightness. However, the way a digital camera takes a photo/video is distinct from the way deepfake videos/images are generated e.g., encoder separately learns the standard deviation (spread) and the center (mean) of the latent space distribution during training. After generating the latent space vector, the decoder is invoked. By distorting one of the centroids with the standard deviation and adding random error or noise, it attempts to recreate the source image. Similarly, a GAN generator converts the input noise into latent space vector and the GAN discriminator compares the generated image with the real image. Finally, the fake image is created following a zero sum game.

Social media images/videos are highly compressed. Along with computers, people use smartphones and tablets to access their social media accounts. Existing solutions for detecting deepfake images and videos are primarily for uncompressed data, and the models are not suitable for social media videos. As people access their social media accounts via their smartphones, the model should also be compact. Hence, three problems are to be solved concurrently: detecting deepfake videos, developing a model applicable to compressed video, and developing a lighter version of the model.

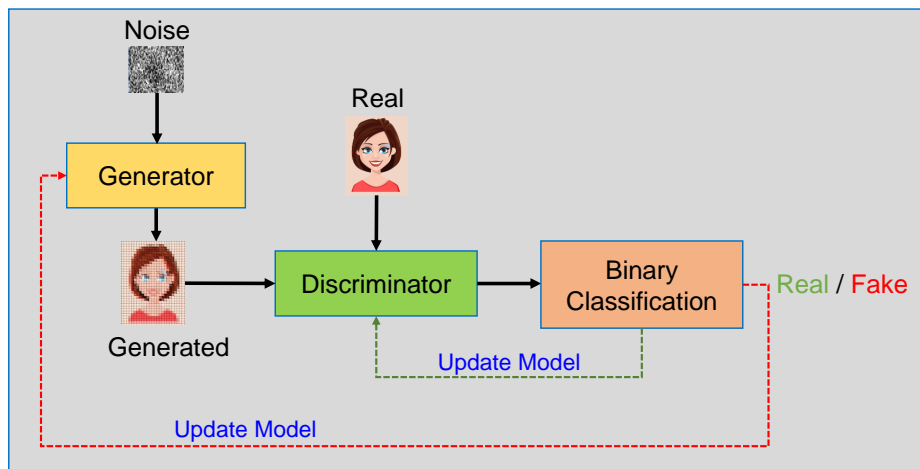


FIGURE 3.2. Deepfake creation by GAN.

3.5. Social Media Deepfake Video Detection

3.5.1. The Proposed Solution

3.5.1.1. Overview

To address the above mentioned challenges, we propose a novel technique of detecting deepfake videos in social media at any compression level applying key video frame approach and deep learning network as in Fig. 3.3. Our method involves fewer computations and is fit for deploying at edge platforms. Eventually it can be applied as a fake video detecting tool at edge.

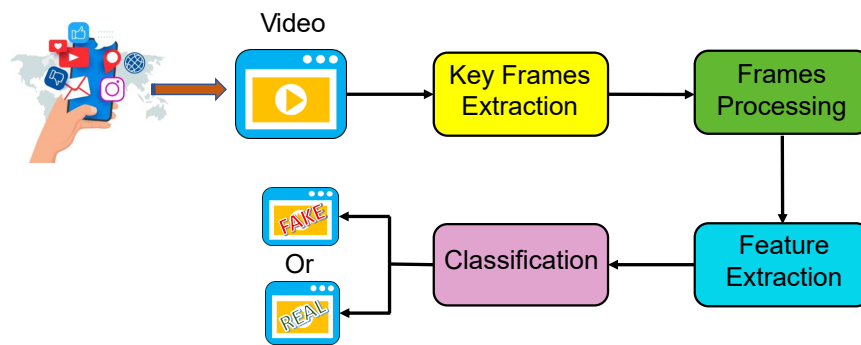


FIGURE 3.3. System level overview of the proposed network.

First, we detected each frame of the video until a fake frame is detected. Then, rather than detecting every frame, we used a key video frame extraction technique. We assume that as key frames contain all of the information of a video, it also contain visual artifacts contributed by the manipulation technique. Therefore, our detection method is based on detecting changes in visual artifacts in a frame due to deepfake forgery. This simple assumption allows us to propose an algorithm that is both accurate and computationally efficient. We reduce the amount of data by extracting key video frames from each video. It limits the number of frames from videos that must be checked for authenticity while maintaining high accuracy. Instead of checking each frame, we only check the visual artifact changes for the key video frames during detection. This reduces the number of computations. As, the detection process requires fewer data computations, this approach is a step closer to deploying our network at the edge. In contrast to the highly computationally expensive

existing works, we propose a light-weight approach. The proposed algorithm has a lower computational cost, and can be used at the edge. Limited memory of the smart phones will not be an issue. The following are our primary contributions:

- A lower complexity algorithm to detect deepfake videos.
- For feature extraction, we initially used three different CNN networks of varying sizes - (1) Xception, (2) Inception V3, and (3) Resnet50. We compared them and ultimately selected Xception network [52] as our CNN module.
- For detection, existing works use extremely complex structures such as [169, 92, 132]. Our primary objective is to reduce the computational cost of detection without sacrificing too much accuracy.

The workflow of the proposed method is shown in Fig. 3.4. It has four major parts - *key video extraction, data processing, features extraction, and classification.*

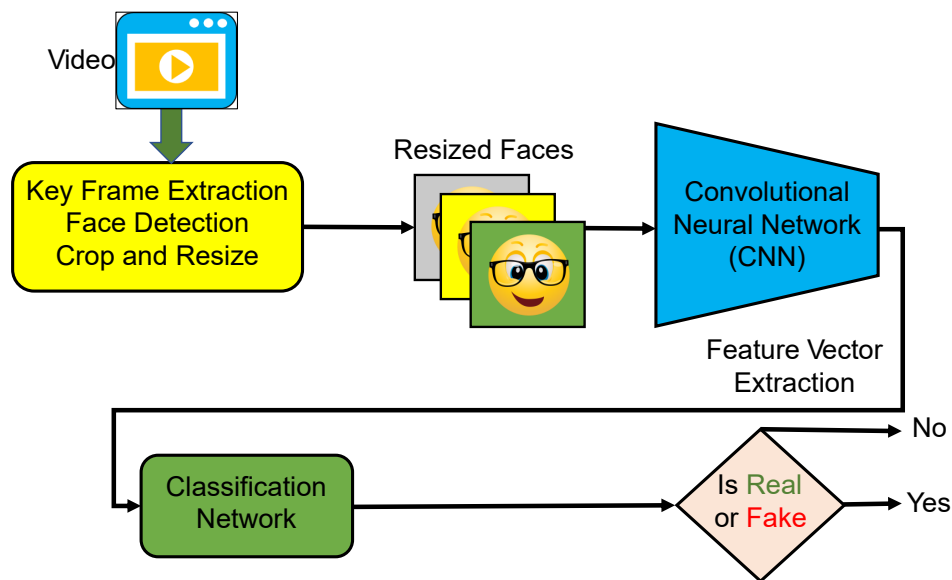


FIGURE 3.4. A detailed representation of the proposed method.

3.5.1.2. Key Video Frame Extraction

There are numerous elements in a video that do not change between frames. A key frame, intra-frame, or i-frame is a frame that signifies the beginning or conclusion of a transition. Subsequent frames only contain information that has changed. Processing

each frame and verifying its authenticity consume a great deal of resources. To reduce the computational complexity of the model, only key video frames were extracted from videos. As our research primarily focuses on visual artifacts that change with forgery, we assume that key frames alone will suffice for our model to detect a deepfake video. Fig. 3.5 displays the key frames from a 20-second video.



FIGURE 3.5. Generated key video frames from a 20 second video.

3.5.1.3. Features Extraction

A convolutional neural network (CNN) has been used as features extractor. Three different CNNs - ResNet50, InceptionV3, and XceptionNet have been experimented as features extractor. Finally, XceptionNet has been selected as the feature extractor. It extends the Inception architecture by substituting depthwise separable convolution for spatial convolution. The distinction between InceptionV3 and XceptionNet is the order of 3×3 spatial channel-wise convolution and 1×1 point-wise cross-channel correlation mapping convolutions. The original Xception network consists of 36 convolution layers organized into 14 blocks. Except for the first and last, each block has a linear residual connection. It extracts features from each frame and returns a 2048-dimensional feature vector for each. It is forwarded to the classifier network. Fig. 3.6 shows the original Xception network introduced by Chollet.

3.5.1.4. Classification

It is a classical binary classification problem. Fig. 3.7 displays the classification network. We chose a combination of layers for the classification network to improve accuracy -

a *GlobalAveragePooling* layer, followed by a *dropout* layer with a *dropout* layer of 0.5, followed by a *fully connected* layer with 0.5 dropout and *ReLU* activation, and lastly a *Softmax* layer that, in essence, classifies the detected video as either real or manipulated.

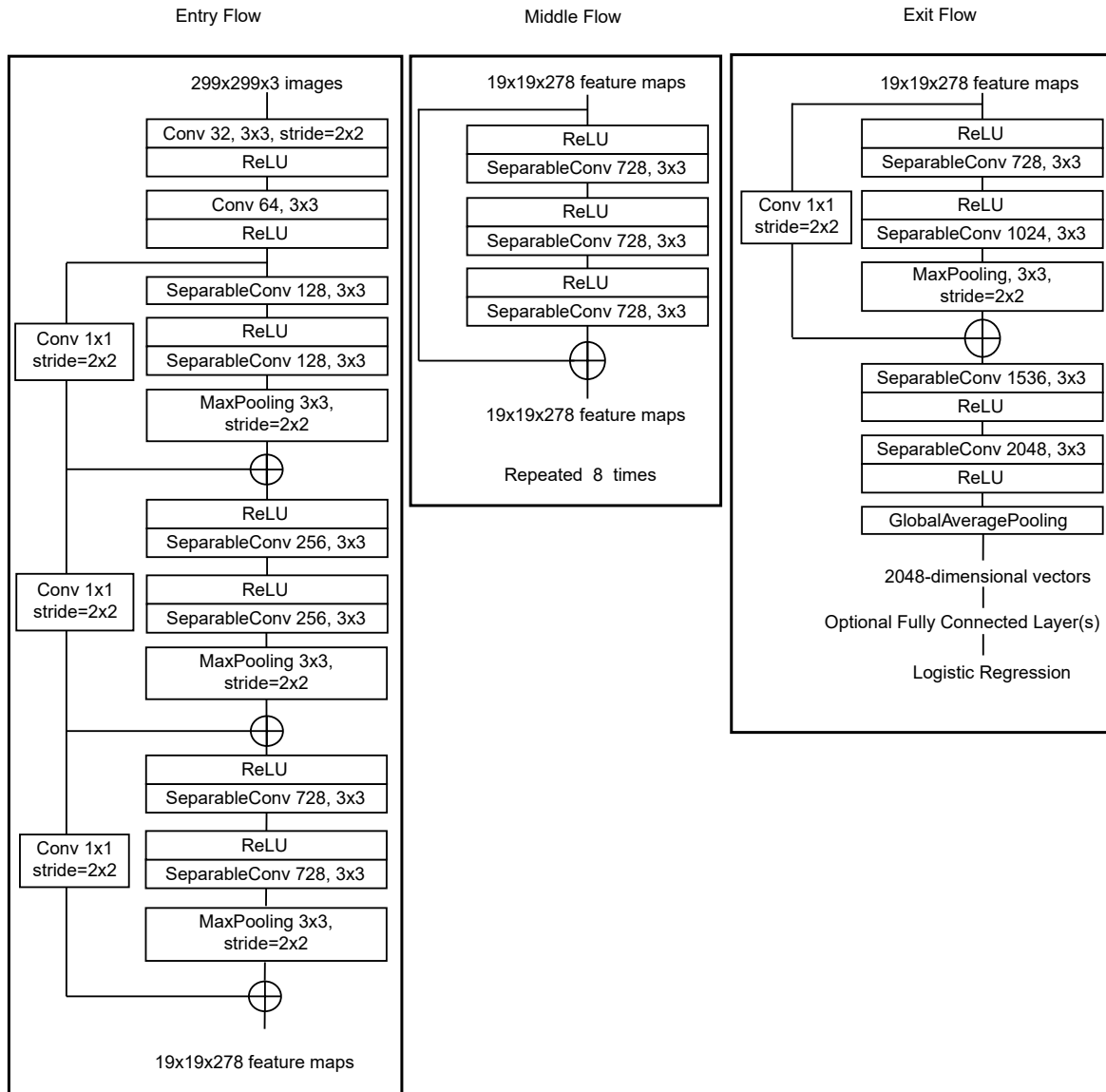


FIGURE 3.6. XceptionNet as the feature extractor.

3.5.1.5. Proposed Algorithms

Initially, deepfake videos are checked frame by frame in our initial Algo. 1 [155]. The accuracy obtained is excellent. However, we had to process a large number of frames. This

is where the final algorithm comes from. To reduce computation, we propose Algo. 2 in detecting deepfake videos.

- Our first algorithm is unique in that it reduces the complexity of detecting forged video. The time complexity is $\mathbf{O}(n)$, where n is the number of video frames extracted.
- The reason for proposing Algo. 2 is to reduce complexity even further. Because key frame extraction greatly reduces the number of extracted frames from a video, the time complexity is also greatly reduced.

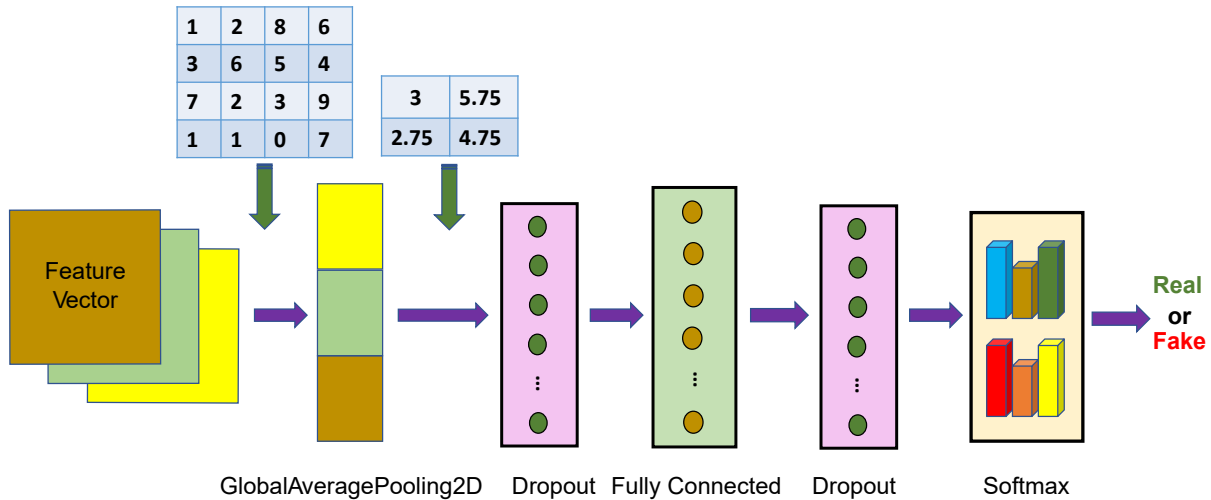


FIGURE 3.7. Classifier network.

3.5.2. Theoretical Perspective

3.5.2.1. Depthwise Separable Convolution

Three elements comprise a convolution operation:

- Input image
- Feature detector or Kernel or Filter
- Feature map

A small number matrix represents the Kernel or the Filter. The convolution operation between the filter value and the pixel value of the input image at each point (x, y) is defined

Algorithm 1 How to Detect DeepFake Video?

- 1: **Input:**Test video v , Model \tilde{M}
- 2: **Output:**Label tag
- 3: Declare and initialize $frames$, f , $face$, and $resface$ to 0
- 4: Assign total number of frames, a particular frame , cropped face respect to the frame f , and resized face respect to the face $face$ to the initialized variables respectively
- 5: Declare and initialize $realtag$ and $faketag$ to 0
- 6: Assign real probability and fake probability after prediction to these variables respectively
- 7: Set $tag = False$
- 8: Extract **all frames** from the test video v
- 9: Save the extracted frames in $frames$
- 10: **for** $f \in frames$ **do**
- 11: Detect the face for f
- 12: Crop the face and Save it in $face$
- 13: Resize the image to (299, 299) and Save it in $resface$
- 14: Load the Model \tilde{M}
- 15: Predict $resface$
- 16: Set $realtag$ to real probability of the
- 17: prediction
- 18: Set $faketag$ to fake probability of the prediction
- 19: **if** $realtag \gg faketag$ **then**
- 20: **continue**
- 21: **else**
- 22: Set $tag = True$
- 23: Consider the video as Fake
- 24: **break**
- 25: **end if**
- 26: **end for**

in Eq. (3.1) :

$$(3.1) \quad (I * h)(x, y) = \int_0^x \int_0^y I(x-i, y-j)h(i, j) didj,$$

where I is the input image and h is the kernel. If $D_F \times D_F \times M$ is the size of the input image and the filter size is $D_K \times D_K \times M$, $N \times D_G^2 \times D_K^2 \times M$ represents the complexity of the convolution operation where M is the number of channels in the input image. $D_G \times D_G \times M$ is the size of the feature matrix.

However, the complexity is decreased for *Depthwise Separable Convolution*. It separates convolution operation in two steps (1) Depthwise Convolution - Filtering stage and (2) Pointwise Convolution - Combination stage. In depthwise convolution the complexity is $M \times D_G^2 \times D_K^2$ while for pointwise convolution it is $N \times D_G^2 \times D_K^2 \times M$. The total complexity is expressed in Eq. (3.2):

$$(3.2) \quad Total\ Complexity = M \times D_G^2 \times D_K^2 + N \times D_G^2 \times D_K^2 \times M$$

The relative complexity of the two convolutions is:

$$(3.3) \quad \left(\frac{Complexity\ Depthwise\ Separable\ Conv.}{Complexity\ Standard\ Conv.} \right) = \left(\frac{1}{N} + \frac{1}{D_K^2} \right).$$

It is obvious from Eq. (3.3) that standard convolution is significantly more complex than depthwise separable convolution. It indicates that the Xception Network offers faster and more affordable convolution than conventional convolution.

3.5.2.2. GlobalAveragePooling Layer

It aids in reducing the number of parameters and, ultimately, overfitting. It performs down sampling by calculating the mean or average of the width and height input dimensions. There are no parameters to learn for the Global Average Pooling layer. It takes the average of each feature map for each category of a classification problem and generates a vector that is fed directly into the next layer. It is more resilient because it compiles spatial information.

3.5.2.3. Dropout Layer

Overfitting is very common in deep networks. The dropout layer prevents a neural network from overfitting. Eq. (3.4) expresses the least square loss for a single layer linear network with activation function $f(x) = x$, whereas least square error of that network with a dropout layer [203, 33] is defined as in Eq. (3.5) where $\delta_i \sim \text{Bernoulli}(p)$:

$$(3.4) \quad E_N = \frac{1}{2} \left(t - \sum_{i=1}^n w'_i I_i \right)^2.$$

$$(3.5) \quad E_D = \frac{1}{2} \left(t - \sum_{i=1}^n \delta_i w_i I_i \right)^2.$$

Taking the expectation of the dropout gradient:

$$(3.6) \quad \begin{aligned} E\left[\frac{\partial E_D}{\partial w_i}\right] &= -tp_i I_i + w_i p_i^2 I_i^2 + w_i \text{Var}(\delta_i) I_i^2 + \sum_{j=1, j \neq i}^n w_j p_i p_j I_i I_j \\ &= \left(\frac{\partial E_N}{\partial w_i}\right) + w_i p_i (1 - p_i) I_i^2. \end{aligned}$$

The expectation of the gradient of the Dropout Network is expressed as Eq. (3.6) where $w' = p * w$. So if $w' = p * w$, the expectation of the gradient with Dropout becomes equal to the gradient of a Regularized linear network:

$$(3.7) \quad E_R = \frac{1}{2} \left(t - \sum_{i=1}^n p_i w_i I_i \right)^2 + \sum_{i=1}^n p_i (1 - p_i) w_i^2 I_i^2.$$

For $p = 0.5$, Eq. (3.7) gives a maximum.

3.5.2.4. Softmax Layer

A softmax layer is used at the network's end to predict whether the video is pristine or manipulated. It takes a M -dimensional vector and generates another vector of the same size but with values ranging from 0 to 1, bringing the total to 1. If the probability distribution of two classes provided by the *softmax* layer is $P(y_k)$ for input y_k to the *softmax* layer, the *softmax* layer output \hat{y} can be predicted by:

$$(3.8) \quad \hat{y} = \operatorname{argmax}_{k=2} P(y_k).$$

3.5.2.5. Training Loss

During training, we minimize the *Categorical Cross Entropy Loss* to obtain optimal network parameters for best class prediction. The cross entropy loss in binary classification, as in our case, is expressed as:

$$(3.9) \quad \mathcal{L} = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})).$$

To minimize the loss stated in Eq. (3.9), we use the *Adam* optimizer. To obtain optimal parameters, one mini batch is processed at each iteration. When the loss function \mathcal{L} is optimized after several epochs, network parameters are learned to their optimal value.

3.5.3. Experimental Validation

In this section, experimental results have been reported.

3.5.3.1. Datasets

Two major datasets from two different generations of deepfake videos (Fig. 3.8) - Face Forensics++ DF (FF++ DF) [186] and Deepfake Detection Challenge (DFDC) [71] - have been used.

For selecting the feature extractor, FF++ DF data at compression level $c=23$, as shown in Table 3.1 has been used. However, once the feature extractor is selected, a mixed compression level dataset as in Table 3.2 made from the aforementioned datasets is utilized to train and evaluate the model. We concentrated on the compressed dataset because any image/video loses clarity as compression increases. FF++ DF supports two video compression levels. It represents a plausible social media scenario. When creating our dataset, we changed the compression level of the DFDC dataset. The dataset details are shown in Table 3.2.

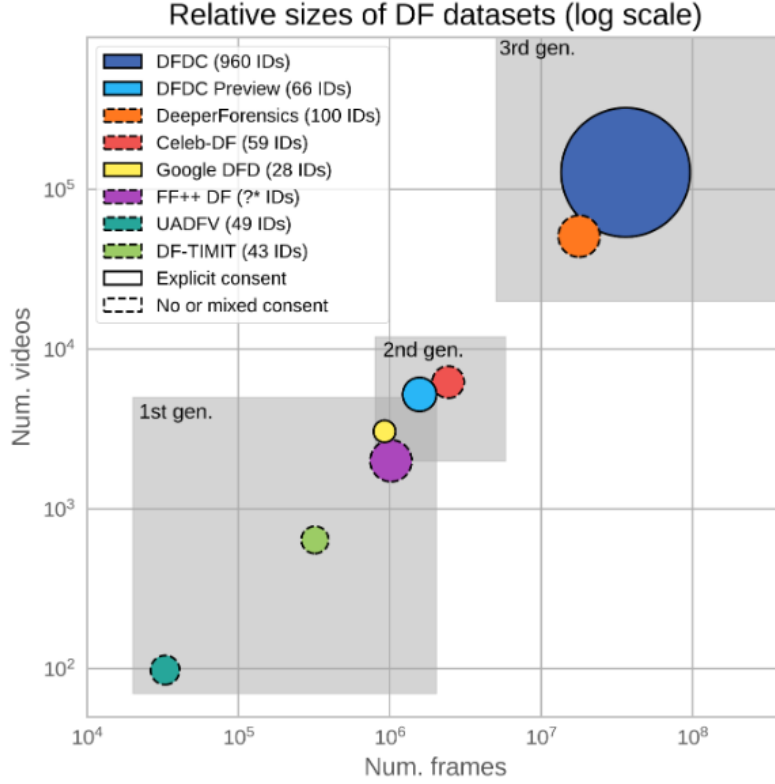


FIGURE 3.8. Different deepfake datasets [71].

TABLE 3.1. Dataset for Selecting Feature Extractor.

Dataset Division	No. of Original Videos ($c=23$)	No. of Manipulated Videos ($c=23$)
Train	800	800
Valid	100	100
Test	100	100

TABLE 3.2. Dataset Details for Deepfake Detection.

Dataset	No. of Original Videos ($c=15, c=23, c=40$)	No. of Manipulated Videos ($c=15, c=23, c=40$)
FF++	2000	2000
DFDC	5773	5765

- *FF++ DF*: 2000 deepfake videos and 2000 real videos at different compression levels from FF++ DF dataset have been used. FaceForensics++ dataset is a good representation of the social media scenario because videos uploaded on social media are compressed. We used two compression level video sets, one with a quantization parameter or constant rate factor of 23 and the other with a constant rate factor of 40.
- *DFDC*: We utilized a portion of a 470GB dataset, which included 5765 manipulated videos and 5773 original videos. We altered the compression levels to $c = 15$, $c = 23$, and $c = 40$. We limited the number of videos for low-level compression to $c = 15$. As loss increases with compression level, we desired to train our network on $c = 23$ and $c = 40$ videos rather than $c = 15$ videos. Using FFmpeg software [37, 221], we modified the compression levels of the H.264 encoder.

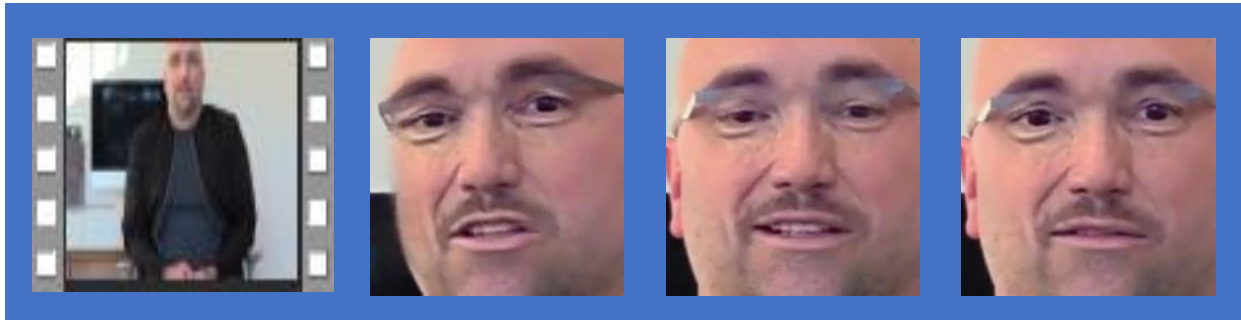
We compiled our data set from 7,773 authentic and 7,7765 fraudulent videos with varying compression levels. To eliminate any kind of data bias or preference, we kept the number of videos in each class - both manipulated and real - nearly identical. 600 mixed compression original and manipulated videos were set aside for testing the model’s accuracy, while the remaining videos were used for training and validation. Fig. 3.9(a) depicts the key video frames from a 10-second fake video in the DFDC dataset, while Fig. 3.9(b) depicts the key video frames from a 24-second fake video in the FF++ DF dataset.

3.5.3.2. Data processing

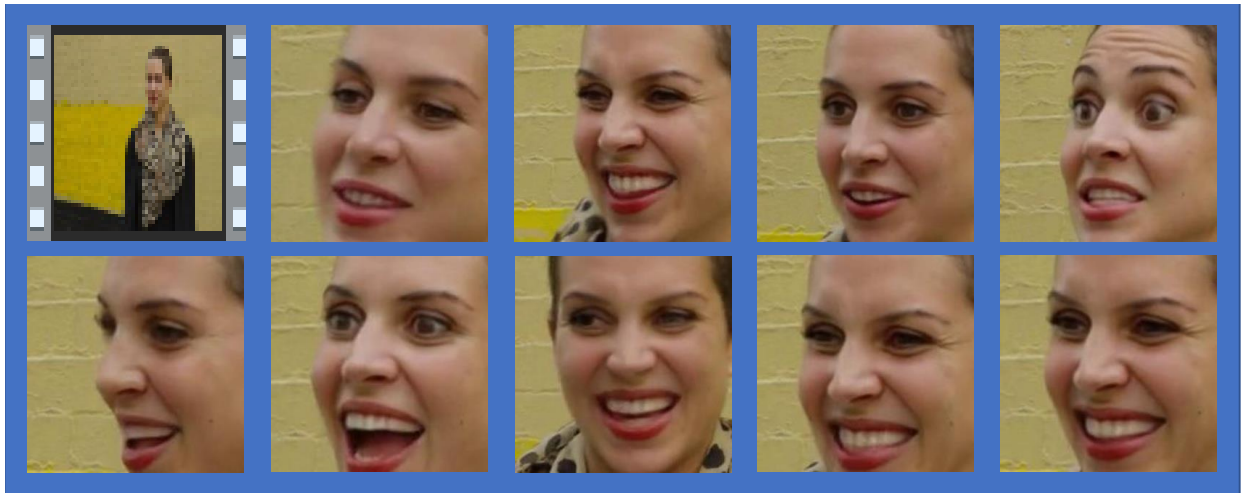
Clips of four seconds are taken from the original and deepfake manipulated videos to create a dataset of videos. Then, frames are extracted without decompression from each compressed video. The faces were then detected and cropped from each frame. All frames are then normalized and resized according to the input from the various CNN modules. For InceptionV3 and Xception net, the image size is maintained at $299 \times 299 \times 3$ and for ResNet50, it is $224 \times 224 \times 3$. Fig. 3.10 displays the data processing flowchart.

Data processing plays a significant role in our work. For the first part of our work we followed the same techniques as before [155]. For our newly proposed work, after extracting

the key video frames from each video we perform additional data processing. To increase the accuracy of the model we then detect all faces and crop the faces from each frame. Finally all frames are normalized and resized as per the input requirement of the CNN module. *Imagesize* is kept at (299, 299, 3) for InceptionV3 and XceptionNet and (224, 224, 3) for ResNet50. The data processing diagram is shown in Fig. 3.10.



(a) From a 10 sec deepfake video.



(b) From a 24 sec deepfake video.

FIGURE 3.9. Key video frames from different length videos.

3.5.3.3. Experimental Setup

The implementation setup is described in this section. The work is divided into two sections. The feature extractor was chosen in the first section, and the deepfake videos detection method was discussed in the second.

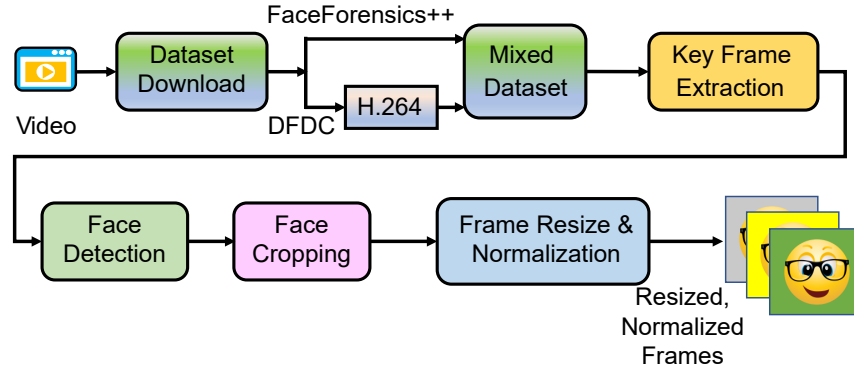


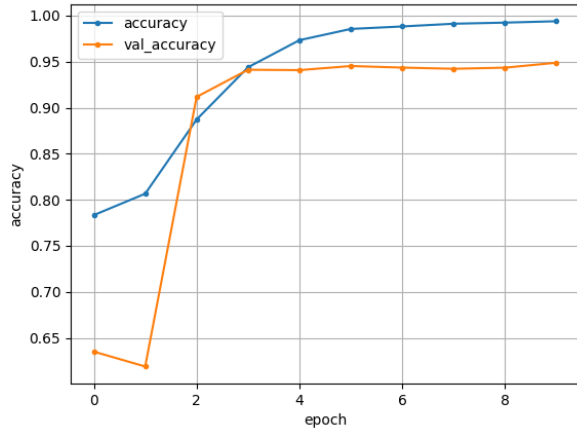
FIGURE 3.10. The proposed flow of video processing.

3.5.3.4. Training Protocol

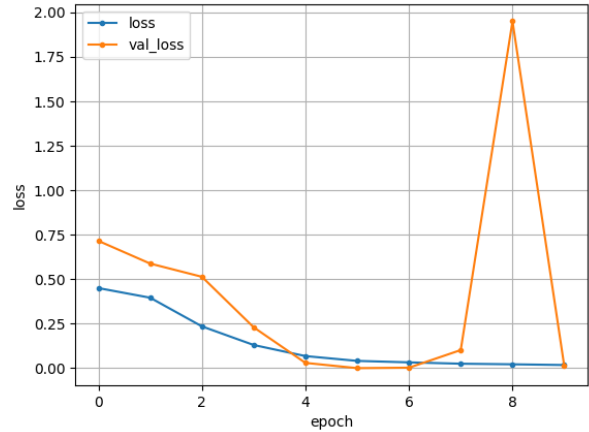
To improve precision and reduce training time, *transfer learning* has been used. A pre-trained model approach was utilized. Initially, the Imagenet trained Resnet50, InceptionV3, and XceptionNet have been separately used as feature extractor. As they were already trained on 1,000,000 images across 1000 classes, they have learned to detect fundamental and general image characteristics. Lower level layers extract simple features such as lines or edges, whereas middle and upper level layers extract more complex, abstract, and classification-defining features.

Initially we trained the whole network. The Accuracy plot and the Loss plot for *XceptionNet* and the *Classifier* are shown in the first row of the Fig. 3.11. The validation accuracy obtained in this training was close to 95%. However, when we used *transfer learning*, much better accuracy as shown in the second row of the Fig. 3.11 has been obtained. We train the classifier for 4 epochs keeping the feature extractor’s weights frozen and then fine tuned the whole network from end-to-end for 10 epochs. Use of *transfer learning* made possible for such less number of epochs for training.

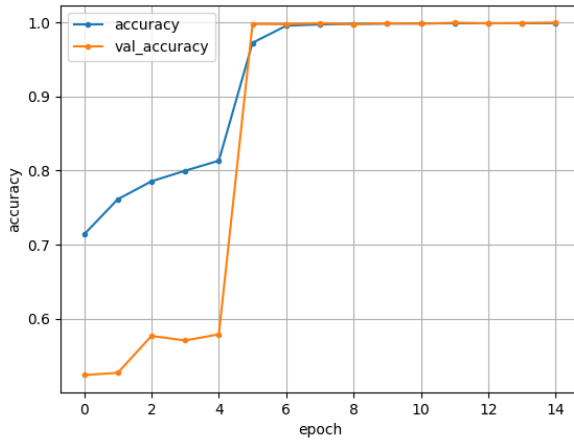
Fig. 3.13 shows the total workflow of the detection system and Fig. 3.12 shows the entire process. Table 3.3 describes the number of frames used for training and validation for feature extractor selection and deepfake detection.



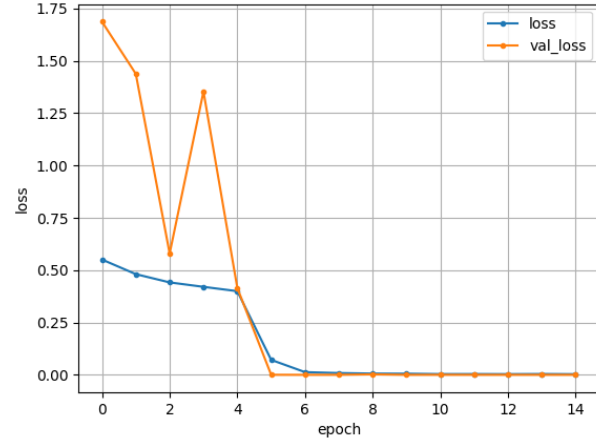
(a)



(b)



(c)



(d)

FIGURE 3.11. Accuracy and loss plots for two different scenarios. (a) and (b) are accuracy and loss plots respectively, when the end-to-end network is trained for 8 epochs. (c) and (d) are accuracy and loss plots respectively, when the classifier is first trained for 4 epochs, keeping the feature extractor’s weight frozen, and then the end-to-end network is trained for 10 epochs.

3.5.3.5. Implementation Details

We used Keras [51] with TensorFlow [26] to implement our proposed framework. FFmpeg is used to clip the videos, and the dlib package’s 68-landmarks are used for face detection. OpenCV and Katna libraries have been used for extracting key video frames. We used a Tesla T4 GPU with 64GB of memory for training. A laptop with a GeForce RTX 2060 is used to test the model.

Among the three CNN modules, XceptionNet paired has been chosen as a feature extractor as it has the highest accuracy. It has been validated using 200 videos of unseen FF++ DF dataset.

The accuracy of compression level $c=23$ is higher than that of compression level $c=40$. Once the feature extractor is selected, the model has been trained using the customized dataset made from FF++ DF and DFDC and verified it with unseen data from the FF++ and DFDC test datasets. Algo. 2 has been used to detect the deepfake video.

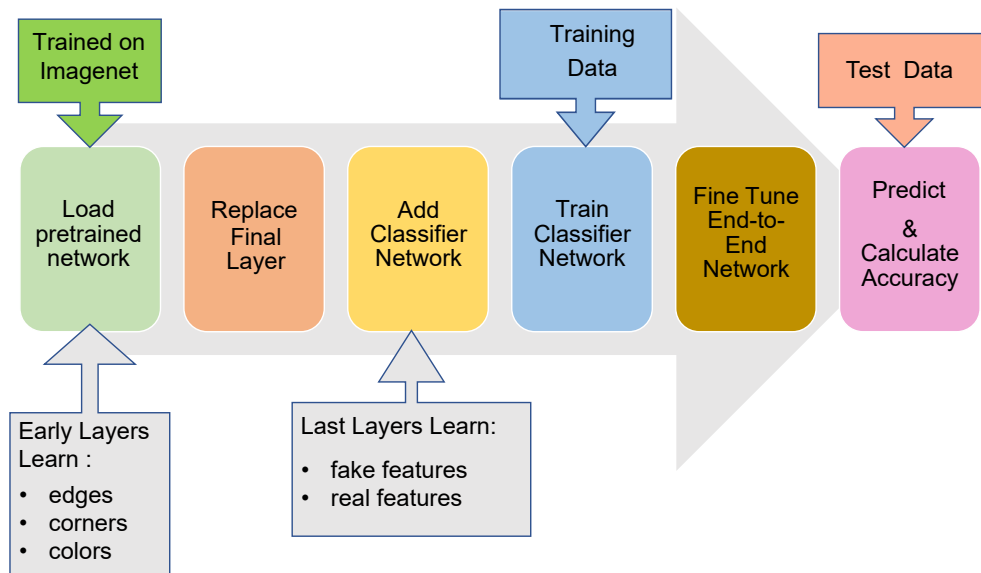


FIGURE 3.12. End-to-end workflow.

TABLE 3.3. Details of the Frames for Training and Validation.

Division	No. of Frames for Feature Extractor Selection	No. of Frames for Deepfake Detection
Train	124959	49373
Valid	31240	12345

3.5.4. Results

In this section, we discuss the results. Fig. 3.14(a) shows XceptionNet with our classifier performed better than other CNNs. Here Algo. 1 has been applied.

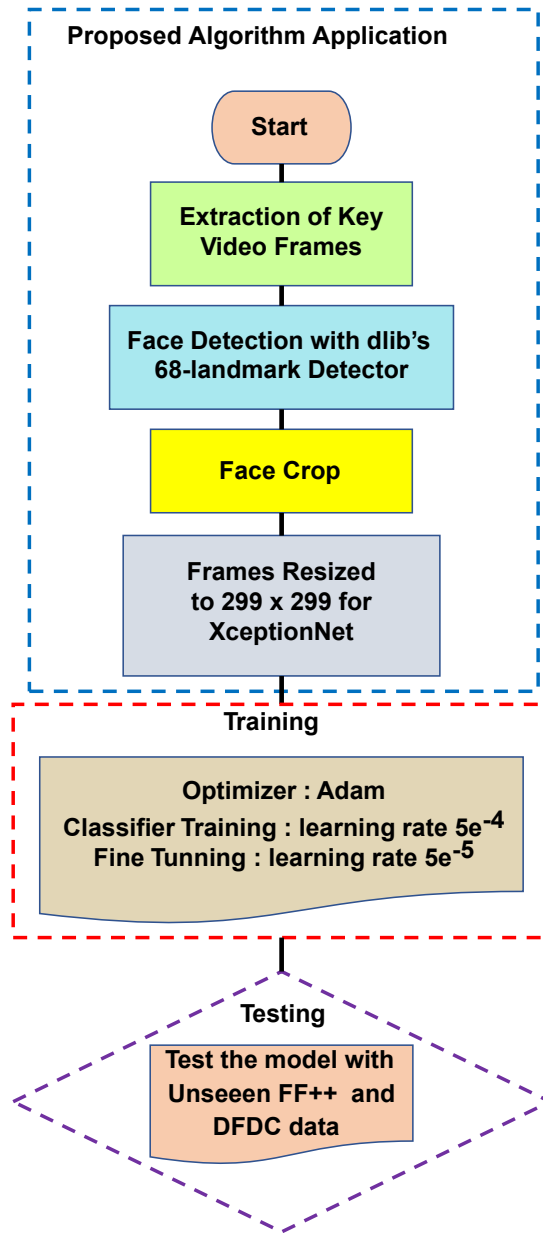
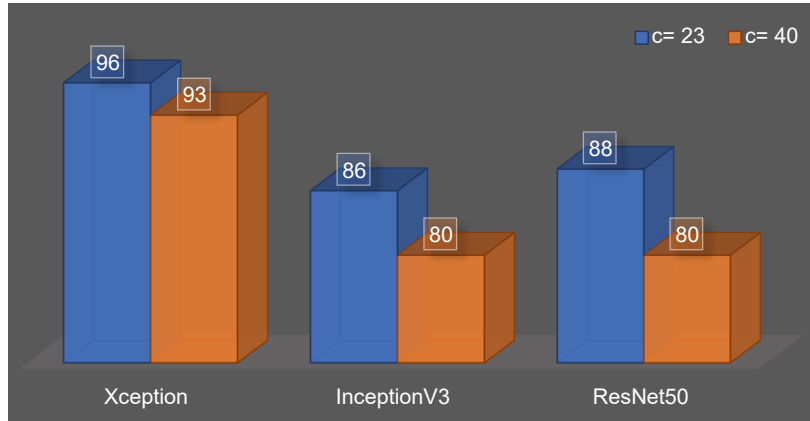
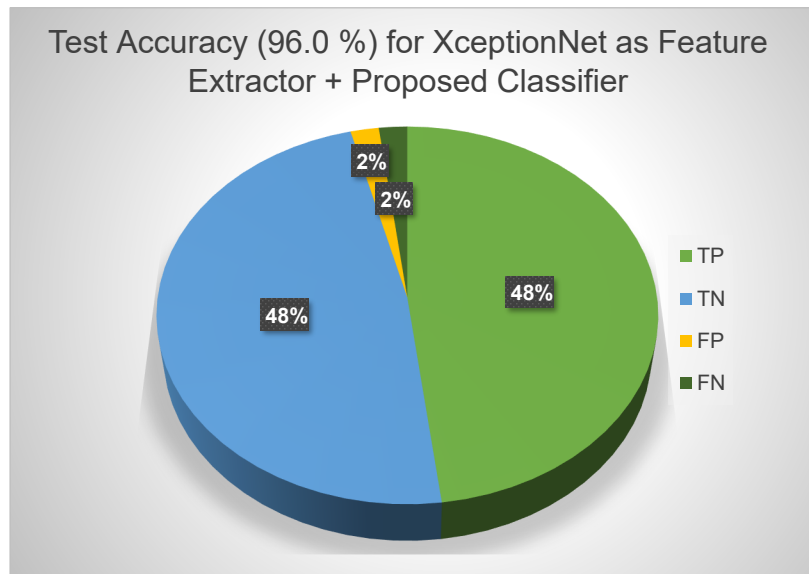


FIGURE 3.13. Developmental workflow of the detection system.

XceptionNet achieves 96% accuracy for the unseen videos with compression level $c = 23$ of FF++ DF dataset. Once the feature extractor was chosen, we moved to the final work. The model is tested in various testing scenarios with different algorithms and dataset combinations as in Fig. 3.15. 96% accuracy has been achieved applying Algo. 1 with 200 unseen FF++ DF data with compression level $c = 23$ whereas Algo. 2 has improved the accuracy to 98.5% even with mixed compression data.



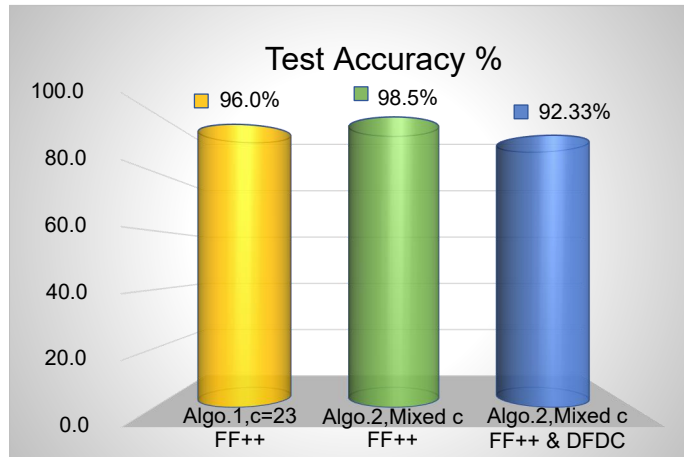
(a) Test accuracy for different CNNs as feature extractor.



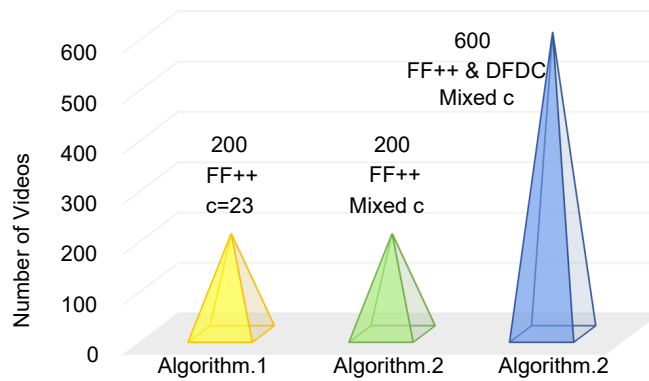
(b) Test accuracy calculation.

FIGURE 3.14. Selection of CNN as feature extractor.

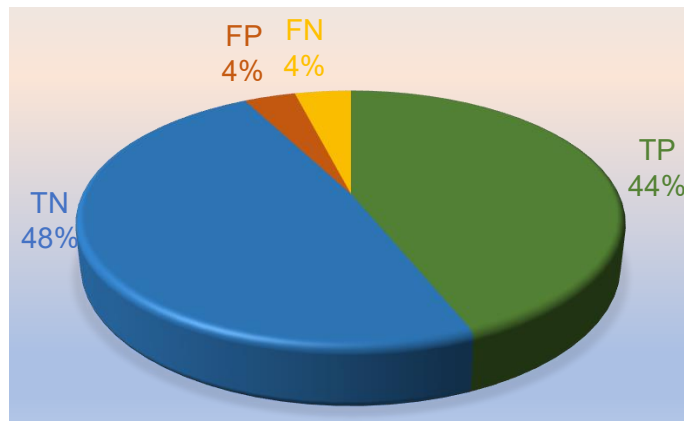
Finally, the method has been evaluated with 600 mixed compression videos from FF++ DF and DFDC datasets. Majority of them are highly compressed ($c=40$). Even with high loss videos, our method was able to achieve an accuracy of 92.33%. Features maps from different layers of XceptionNet have also been visualized to understand learning of the layers of the feature extractor. Three sample feature maps, shown in Fig. 3.16(b), 3.16(c), and 3.16(d) for the key video frame of Fig. 3.16(a), confirm the earlier layers learn simple features and latter layers learn more complex features of the input image.



(a) Test accuracy plot.



(b) Number of videos for the experiments.

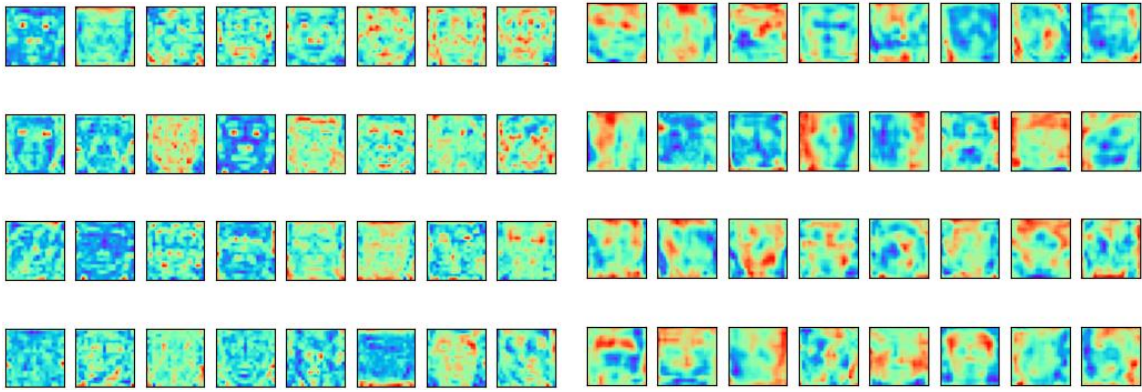


(c) Test accuracy calculation for mixed compression FF++ DF and DFDC data

FIGURE 3.15. Test accuracy for various testing scenarios.



(a) Test video frame (Fake). (b) What does layer 10 of Xception see?



(c) What does layer 45 of Xception see? (d) What does layer 90 of Xception see?

FIGURE 3.16. Sample CNN layers outputs.

3.5.4.1. Performance Metrics

As detection of deepfake video is a binary classification problem, confusion matrix is defined as in Table 3.4 to evaluate the performance of the method. As the chance of fake video in a bunch of videos is less than a real video, we define the fake video as *positive* class. Fig. 3.17 shows the derived *confusion matrix* for this problem.

Various performance metrics e.g., *accuracy* in Eq. 3.11, *precision* in Eq. 3.13, *recall* in Eq. 3.14, and *F1-score* in Eq. 3.14 have also been calculated to evaluate the performance of the model. These metrics are calculated and are shown in Table 3.5.

Our Algo. 1 is applicable to any length video. The number of key frames in most fake videos is low because only the face is changed. Even if only one key frame is extracted from

the testing video, Algo. 2 can detect fake videos, but its accuracy increases dramatically (almost all results were correct) if the video has more than one key frame.

TABLE 3.4. Confusion Matrix - Definition of TP, TN, FP and FN.

True Positive (TP): Reality : Fake(1) Model predicted : Fake(1)	False Negative (FN): Reality : Fake (1) Model predicted : Real(0)
False Positive (FP): Reality : Real(0) Model predicted : Fake(1)	True Negative (TN): Reality : Real(0) Model predicted : Real(0)

$$(3.10) \quad Accuracy = \left(\frac{TP + TN}{TP + TN + FP + FN} \right)$$

(3.11)

$$(3.12) \quad Precision = \left(\frac{TP}{TP + FP} \right)$$

$$(3.13) \quad Recall = \left(\frac{TP}{TP + FN} \right)$$

$$(3.14) \quad F1 - score = \left(\frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \right).$$

The method may not produce accurate results if the video is very hazy. The hazy fake videos account for the majority of the FN. The training and validation datasets had insufficient hazy videos for our model to learn, resulting in uncertain predictions for those videos.

Because we only trained the model with a portion of the DFDC dataset, the accuracy of the method is lower when combined with the FF++ dataset. As DFDC dataset is large, training the model with entire dataset would have improved accuracy.

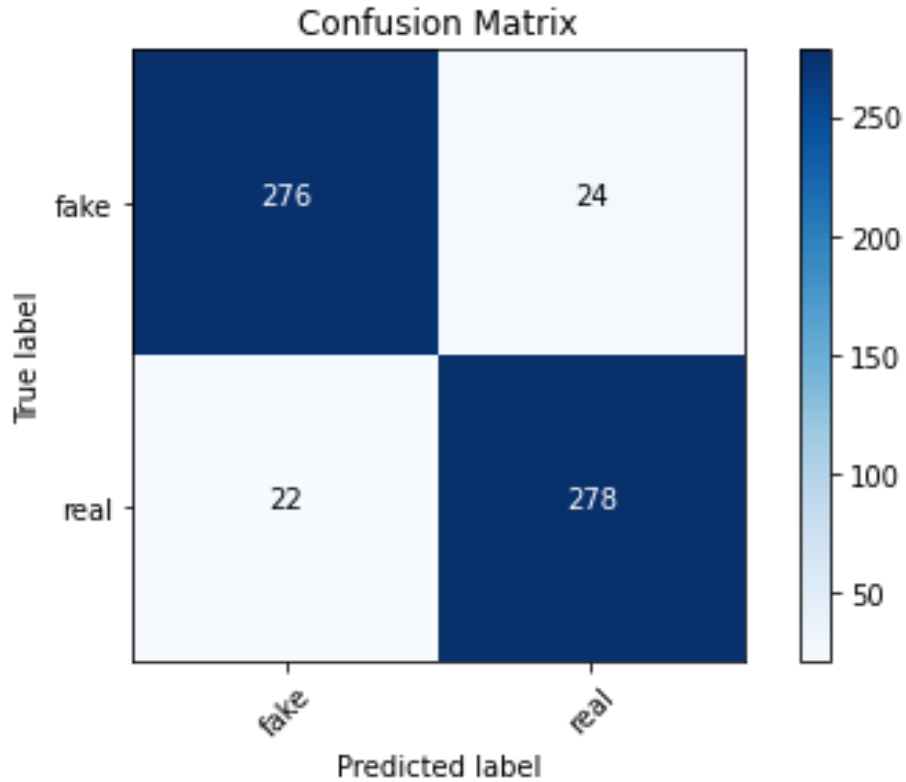


FIGURE 3.17. Confusion matrix.

3.5.4.2. Comparisons

A comparison was made between our findings and those of previous studies, such as [185, 92, 132]. Table 3.6 shows a comparative view. A comparison between our work and that of other researchers can be seen in Fig. 3.18. With our proposed algorithm, we were able to achieve an accuracy of 98.5% and 92.33% for two different test dataset videos. Performance of the proposed method will improve, if more manipulated videos with a variety of actors, different light, and noise conditions are included in the training dataset.

3.5.5. Discussions

First, we classified unaltered and altered videos using Algo. 1 that processes each video frame. We utilized a particular compression level, $c = 23$, as it is in the middle range of compression levels with low loss. It results a high degree of precision. However, the complexity is proportional to the number of extracted frames of the video.

TABLE 3.5. Performance Evaluation of the Detection Method.

Test Scenarios	Model Structure	TP	TN	FP	FN	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)
Algo.1 +	ResNet50	80	96	04	20	88.00	95.00	80.00	87.00
c=23 +	InceptionV3	84	88	12	16	86.00	89.00	88.00	88.00
200 (FF++ DF)	Xception	96	96	04	04	96.00	96.00	96.00	96.00
Algo.2 +									
Mix. c +	Xception	98	99	01	02	98.50	99.00	98.00	98.00
200 (FF++ DF)									
Algo.2 +									
Mix. c +	Xception	276	278	22	24	92.33	93.00	92.00	92.00
600 (FF++ DF& DFDC)									

But with Algo. 2 good accuracy for highly compressed and high-loss data has been achieved. As Algo. 2 significantly reduces the number of computations, it can be deployed on edge devices with the necessary modifications.

3.6. Social Media Deepfake Image Detection at Edge Platform

3.6.1. Overview

In this section, another novel method has been presented for detecting deepfake images on an edge device. These images are GAN-generated. The method is machine learning-based and uses textural features of the images. The proposed detection API performs the operation automatically. There are several reasons for taking this approach:

- The cost of computation is very low. So it is suitable for IoT environments with limited resources.
- The method is generic because it is based on gray level co-occurrence at pixel locations. It is applicable to any GAN type.

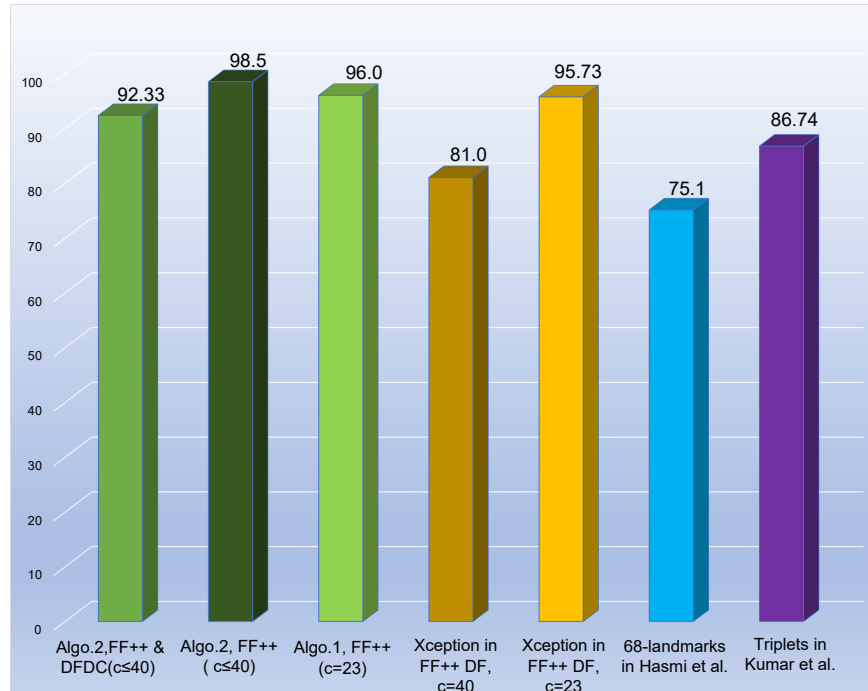


FIGURE 3.18. Accuracy comparison.

3.6.2. Detection Methodology at Edge Platform

3.6.2.1. Detection Method

Fig. 3.19 shows the overall diagram of the detection method. In this IoT setting, the edge device is linked to the end users or clients. The data is stored in the cloud. If necessary, the stored images and predictions can be used to retrain the model in the cloud at a later time. When an image from social media (uploaded by anyone with bad intentions [14]) needs to be checked for authenticity, it is sent to the edge platform through the proposed “Detection API” and the detection process starts.

The way a camera takes a picture is different than deepfake images are generated with deep neural networks. This difference prompted us to look into the textural properties of GAN generated images and finally use them in detecting deepfake images. An image’s texture is a spatial property. A well-known method for capturing the spatial dependence of gray level values is to compute the GLCM of an image. It calculates a pixel’s likelihood and its relationship to other pixels [16]. It is defined as a square matrix with elements representing the frequency of occurrence of gray-level pixels at a given distance and angle.

TABLE 3.6. Performance Comparison of Xception Network paired with Proposed Classifier.

Network	Test Dataset (Unseen)	Compression Level	Accuracy (%)
Xception + Proposed Classifier + Algo.2 + 68 landmarks	FF++ DFDC	Any (checked upto c=40)	92.33
Xception + Proposed Classifier + Algo.2 + 68 landmarks	FF++	Any (checked upto c=40)	98.50
Xception + Proposed Classifier + Algo.1 + 68 landmarks [155]	FF++	c=23	96.00
Xception in FF++ Paper [185]	FF++	c=40	81.00
		c=23	95.73
CNN + LSTM + 68-landmarks[92]	DFDC	NA	75.10
Triples(Semi-hard) [132]	FF++	c=40	86.74

To investigate the fake images further, we use the *Gray Level Cooccurrence Matrix* (GLCM) to calculate several Haralick’s Texture Features [89] - *contrast, homogeneity, energy, dissimilarity correlation*, as in Eq. 3.15, 3.16, 3.18, 3.17, and 3.19, respectively:

$$(3.15) \quad CON = \sum_{i,j=0}^{n-1} p(i,j)(i-j)^2$$

$$(3.16) \quad HOM = \sum_{i,j=0}^{n-1} \frac{p(i,j)}{(1+(i-j)^2)}$$

$$(3.17) \quad ENE = \sum_{i,j=0}^{n-1} p^2(i, j)$$

$$(3.18) \quad DIS = \sum_{i,j=0}^{n-1} p(i, j) |i - j|$$

$$(3.19) \quad COR = \sum_{i,j=0}^{n-1} p(i, j) \left[\frac{(i - \mu_i)(j - \mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right].$$

In the above expression, n denotes the number of gray levels, $p(i, j)$ is the GLCM element for the distance between gray level values i and j , and μ and σ are the mean and variance of the intensities of all gray values present, respectively.

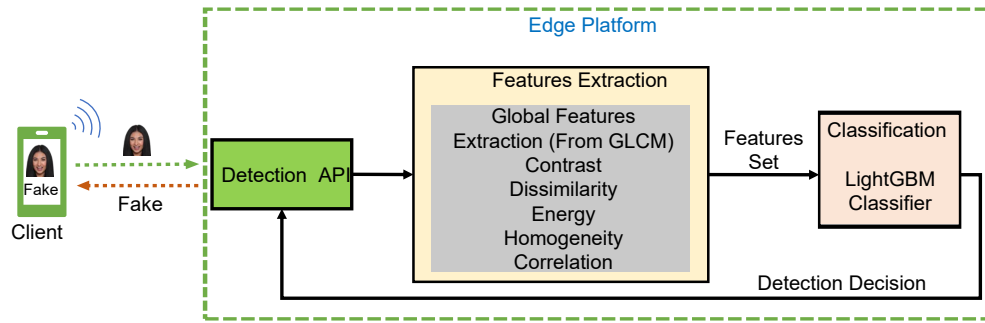


FIGURE 3.19. Overall workflow diagram at edge platform.

During detection process, global textural features of the image are calculated using Algo. 3. GLCM is calculated from the corresponding gray level image of the colored image, followed by the above mentioned Haralick's texture features. These features are combined to form a feature set. GLCM is utilized to calculate those features for four distances at $d = 1, 2, 3, 5$ and three angles $\theta = 0, \pi/4, \pi/2$ to generate the feature vector. After feature extraction, the image is classified using a machine learning algorithm. Because resources are limited in an IoT environment, *LightGBM* with the boosting type *Gradient Boosting Decision Tree (gbdt)* has been chosen. It's a tree-based method. The following are some of the benefits of using this classifier over others:

- The algorithm [119] learns using histograms. Once histograms are created, the time complexity of a histogram-based algorithm is proportional to the number of bins,

and not the data volume.

- The use of discrete bins reduces memory usage, which is a limiting factor at an edge device.
- Distributed training is incredibly fast.

Finally, detection score goes back to where it came from. Along with the image, the detection score is also saved in the cloud so that it can be used to retrain the model in future.

3.6.2.2. Phases of Detection Method

There are two phases to the overall detection procedure - training and testing.

- *Training Phase:* In the training phase, the classifier learns how to detect fake images. The details of the initial training has been described in Section 3.6.3.2. If retraining is required in the future, it can be conducted in the cloud.
- *Testing Phase:* During testing phase, unknown samples are examined. Implementation occurs at the edge device. Through our proposed API, the testing phase is executed.

3.6.2.3. Detection API

We propose an API for detection hosted on the edge device. The API's workflow is depicted in Fig. 3.20. The objective is to make the API lighter and more efficient for edge devices.

3.6.3. Experimental Validation

3.6.3.1. Dataset Details

StarGAN [50] generated images have been used to evaluate the proposed method. To create this dataset the same procedure as in [50] has been followed. Five different physical characteristics, such as hair color (black, blond, brown), gender, and age, have been chosen. No change in expression or disposition has been made. The first 6,000 images from the CelebA [148] dataset have been chosen to generate the images by StarGAN. Each real image generates five images, for a total of 30,000 images. A total of 60,000 images (30,000 fake and 30,000 real) have been used for training, validation, and testing of the detection method.

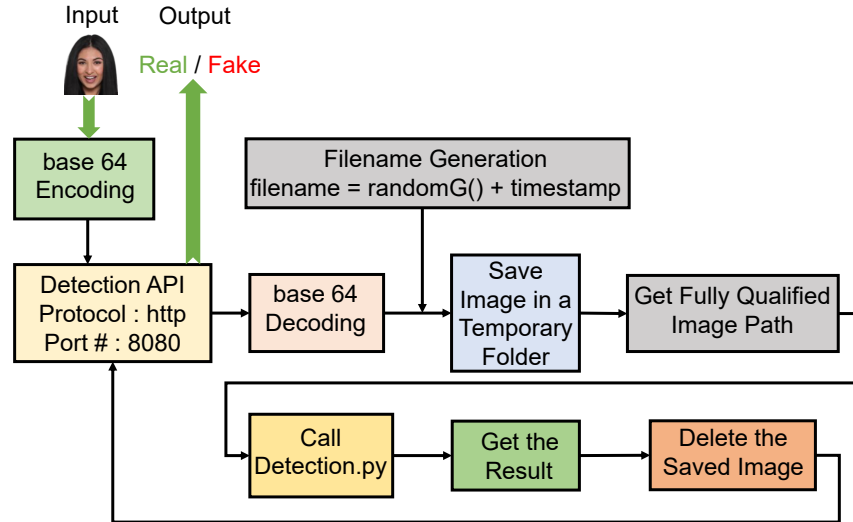


FIGURE 3.20. Detection API workflow.

To provide a balanced dataset, the minority class has been upsampled. Fig. 3.21 depicts examples of StarGAN generated images utilized in the experiment.

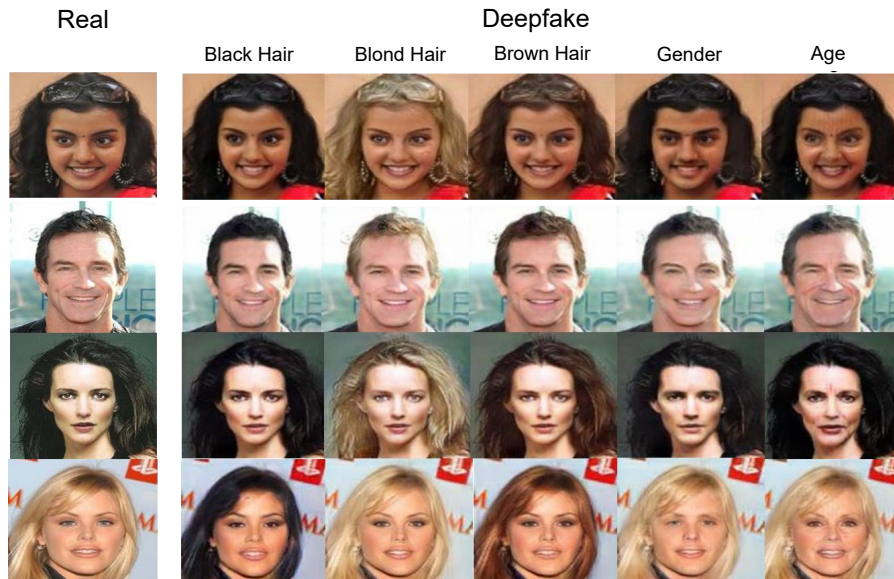


FIGURE 3.21. StarGAN generated sample images.

Fig. 3.22 shows the histograms for red, green, and blue channels of the StarGAN generated deepfake images. In bare eyes even if the images look same as the real images, the histogram color distribution are different.

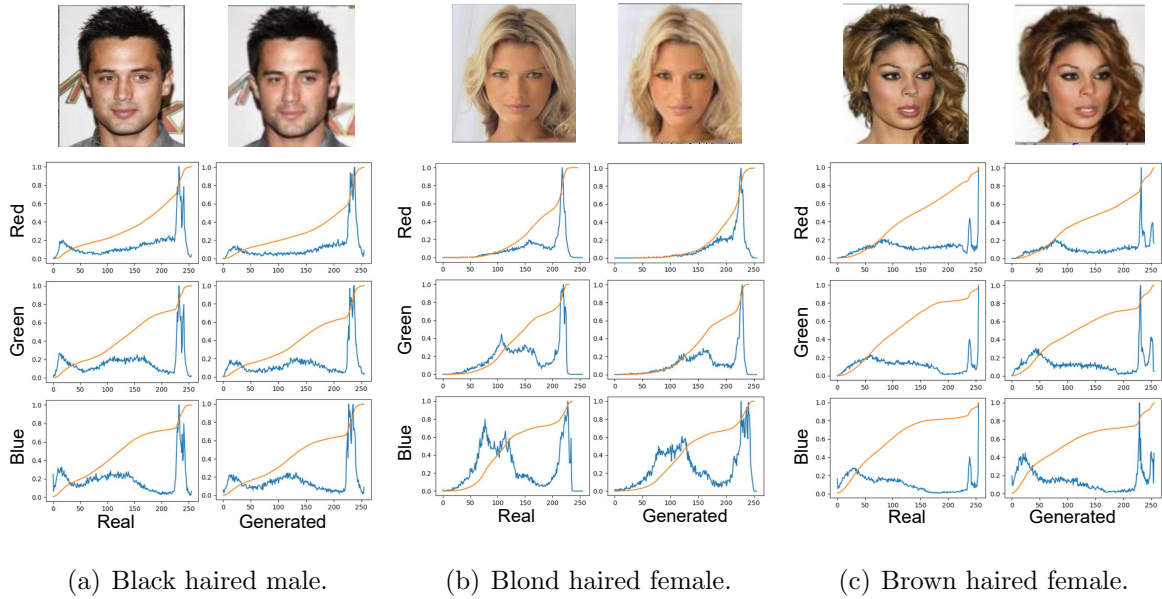


FIGURE 3.22. Histogram comparison of StarGAN generated images and real images.

3.6.3.2. Implementation of Proposed Detection Method

The detection model was implemented on Raspberry Pi 4 with 4GB of memory, a single-board computer as in Fig. 3.23. The input image was sent via the proposed Detection API, and the detection result was returned via the API.

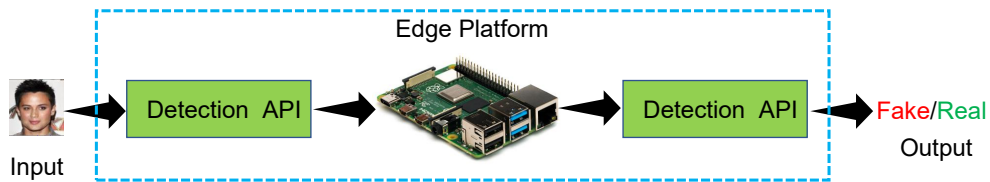


FIGURE 3.23. Implementation of EasyDeep.

Initial training was conducted on a computer with 16GB of memory and Intel Core i7-9750 processor. No GPU was employed. 48,000 images (24,000 deepfake images generated by StarGAN and 24,000 real images) were used for training, while 10,000 images were used for model validation. The remaining 2000 images have been tested. The training and validation of the model took a total of 27 minutes. Before building the features set, the image is converted to gray scale and resized to 256×256 . The features set is constructed

from Haralick’s texture features. For training data, the feature set has a size of $48,003 \times 30$. The learning rate of *LightGBM* classifier is maintained at 0.05, the maximum depth of 600 trees is maintained at 13, and the number of leaves per tree is maintained at 8,500. *Gradient Boosting Decision Tree* is chosen as the boosting algorithm. The detection portion has been implemented in Python, while the API portion has been implemented in Java.

3.6.4. Performance Evaluation

Performance of the detection method has been evaluated using Table. 3.4 and Eq. 3.11, 3.13, 3.14, and 3.14.

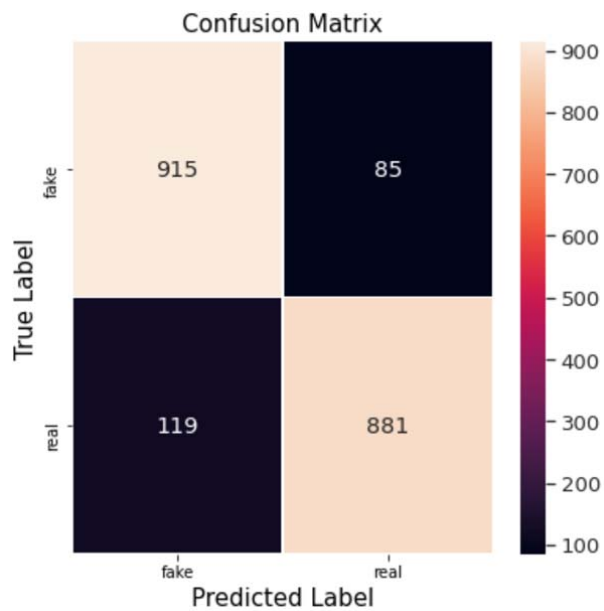
TABLE 3.7. Classification report of EasyDeep on test images.

Test Images	Precision%	Recall%	F1-score%
1000 Fake	88.0	92.0	90.0
1000 Real	91.0	88.0	90.0
Macro Average	90.0	90.0	90.0
Weighted Average	90.0	90.0	90.0
Total 2000	Accuracy %	90.0	
Total 2000	AUC Score %	96.0	

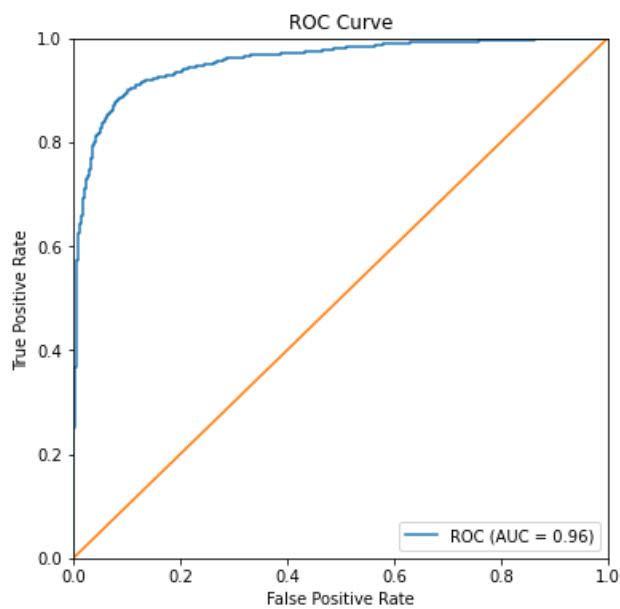
The *confusion matrix* generated from the 2,000 test images is shown in Fig. 3.24(a). *Accuracy, precision, recall, and F1-score* have also been calculated and are shown in Table. 3.7. Fig. 3.24(b) shows area under the curve (AUC) of the drawn Receiver Operating Characteristic (ROC) is 96%.

The relationship between accuracy, tree structure, boosting type algorithm, and model size is displayed in Table. 3.8. We change the parameters in order to create a model that is deployed on a Raspberry Pi and is highly accurate. The final structure is selected with 90% accuracy, 600 trees, and each tree with a maximum depth of 13 and leaf count of 8,500.

The model is trained in 27 minutes. Increasing the number of trees within an algorithm increases its precision. Maximum tree height and leaf count also influence accuracy.



(a) Confusion matrix.



(b) ROC curve.

FIGURE 3.24. Performance metrics calculation for EasyDeep.

TABLE 3.8. Accuracy Variation with Tree Structure.

Number of Trees	Max Tree Depth	Number of Leaves	Boosting Algorithm	Accuracy %	Model Size (MB)
100	8	255	dart*	79.4	3.2
100	10	1000	dart	80.4	6.7
100	11	2500	dart	81.8	12.4
100	12	4200	dart	82.1	15.8
100	13	8500	dart	82.9	19.0
100	14	17000	dart	82.7	22.2
100	13	8500	gbdt*	85.5	14.3
100	14	17000	gbdt	85.9	16.4
200	13	8500	gbdt	87.4	21.5
300	13	8500	gbdt	88.2	27.3
400	13	8500	gbdt	89.0	32.8
600	13	8500	gbdt	90.0	43.7

dart* (Dropouts meet Multiple Additive Regression Trees)

gbdt* (Gradient Boosting Decision Tree)

3.6.5. Discussions

Implementing a computation-intensive computer vision problem such as deepfake detection in an edge platform is not easy. But through this chapter, we were able to make the computation as efficient as possible. To infer accurately on a resource-constrained IoT edge device, we used 30 features per image. Increasing trees and changing the feature set enhance accuracy. More features improve model accuracy and generalization. Instead of sending the image in base64 format, binary images can be sent to improve inference time. Future work can increase accessibility through mobile application. Along with a reduction in inference time, it is possible to generalize the model and increase its precision.

Algorithm 2 How to Detect DeepFake Video using Key Video Frames Approach?

- 1: **Input:**Test video v , Model \tilde{M}
- 2: **Output:**Label tag
- 3: Declare and initialize $frames$, f , $face$, and $resface$ to 0
- 4: Assign total number of Key Video Frames, a particular key video frame , cropped face respect to the key video frame f , and resized face respect to the face $face$ to the initialized variables respectively
- 5: Declare and initialize $realtag$ and $faketag$ to 0
- 6: Assign real probability and fake probability after prediction to these variables respectively
- 7: Set $tag = False$
- 8: Extract **key video frames** from the video v .
- 9: Save the extracted frames in $frames$.
- 10: **for** $f \in frames$ **do**
- 11: Detect the face for f
- 12: Crop the face and Save it in $face$
- 13: Resize the image to (299, 299) and Save it in $resface$
- 14: Load the Model \tilde{M}
- 15: Predict $resface$
- 16: Set $realtag$ to real probability of the prediction
- 17: Set $faketag$ to fake probability of the prediction
- 18: **if** $realtag \gg faketag$ **then**
- 19: **continue**
- 20: **else**
- 21: Set $tag = True$
- 22: Consider the video as Fake
- 23: **break**
- 24: **end if**
- 25: **end for**

Algorithm 3 How to Detect Deepfake Image from Global Textural Features?

- 1: **Input:** Test image I , Model \tilde{M}
- 2: **Output:** Label tag
- 3: Declare and initialize $realtag$, $faketag$, CON , HOM , DIS , COR , and ENR to 0
- 4: Assign real probability of prediction to $realtag$ and fake probability of prediction to $faketag$
- 5: Set $tag = False$
- 6: Declare variables i , j , and an empty \mathcal{G} matrix of size $i \times j$
- 7: Declare a dataframe df
- 8: Assign gray level pixel values of point $(x1, y1)$ to i and $(x2, y2)$ to j
- 9: Declare initial and final points to $(x1, y1)$ and $(x2, y2)$ respectively
- 10: Set $dist \in 1, 2, 3, 5$
- 11: Set $angles \in 0, \frac{\pi}{4}, \frac{\pi}{2}$
- 12: Convert RGB image frame to Gray level image frame
- 13: **for** $d \in dist$ **do**
- 14: **for** $\theta \in angles$ **do** $p_{d,\theta}(i, j) = 0$
- 15: **for** $(x1, y1) \in I$ **do**
- 16: $x2 = d \times \cos \theta$
- 17: $y2 = d \times \sin \theta$
- 18: $I(x1, y1) = i$
- 19: $I(x2, y2) = j$
- 20: $p_{d,\theta}(i, j) \leftarrow p_{d,\theta}(i, j) + 1$
- 21: $\mathcal{G} \leftarrow p_{d,\theta}(i, j)$
- 22: $CON = \sum_{i,j=0}^{n-1} p(i, j)(i - j)^2$
- 23: $HOM = \sum_{i,j=0}^{n-1} \frac{p(i, j)}{(1+(i-j)^2)}$
- 24: $ENR = \sum_{i,j=0}^{n-1} p^2(i, j)$
- 25: $DIS = \sum_{i,j=0}^{n-1} p(i, j)|i - j|$
- 26: $COR = \sum_{i,j=0}^{n-1} p(i, j) \left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right]$
- 27: **end for**

28: $df[CON(d, \theta)] \leftarrow CON$
29: $df[HOM(d, \theta)] \leftarrow HOM$
30: $df[DIS(d, \theta)] \leftarrow DIS$
31: $df[COR(d, \theta)] \leftarrow COR$
32: $df[ENR(d, \theta)] \leftarrow ENR$
33: **end for**
34: Load the trained classifier M^{\sim}
35: Predict I
36: Set *realtag* to real probability of
37: the prediction
38: Set *faketag* to fake probability of the prediction
39: **end for**

CHAPTER 4

DATA FALSIFICATION: DEEPFAKE RESILIENT DIGITAL IDENTIFICATION FOR SMART CITIES

This chapter presents deepfake resilient digital id system for smart cities [157, 102, 154].

4.1. Digital ID System for Smart Cities

The world’s population is increasing at an unprecedented rate. It is estimated that 70% of the global population will be living in cities by the year 2050 [162]. Such rapid urbanization will create more carbon emissions and pollution which in turn will negatively impact the environment and people’s health. It will also create a greater demand for energy, food, and resources. Smart cities have emerged as a resilient and sustainable solution to the problems caused by rapid urbanization. They are envisioned as the future of urbanization, where residents of such cities can benefit from smart transportation, health care, energy, and other seamlessly integrated services which are all connected through the Internet of Things (IoT). Fig. 4.1 shows how smart city stakeholders are connected.

In the last twenty years, the idea of a smart city has stepped forward due to advancements in hardware and software, growth in information and communication technologies (ICT), and initiatives offered by various tech giants [25].

For a practical implementation of the smart-city digital infrastructure, citizens should have a way to easily connect to the amenities offered by the smart city. A wide range of services, such as smart healthcare, smart government and smart financial products should be accessible to the citizens. Checking bank accounts, ordering products online, using smartphones, driving electric vehicles, locking doors with smart locks, lighting homes with smart lights and paying at electronic parking meters are just some of the ways people are already transforming traditional towns into “smart cities.”

In recent years, a variety of cutting-edge facial recognition systems, various computing platform-based facial authentication systems, and threat detection approaches have been

presented in various publications. Even though there have been many creative ideas, no end-to-end face authentication system that could be used for a digital ID system in smart cities has been offered.

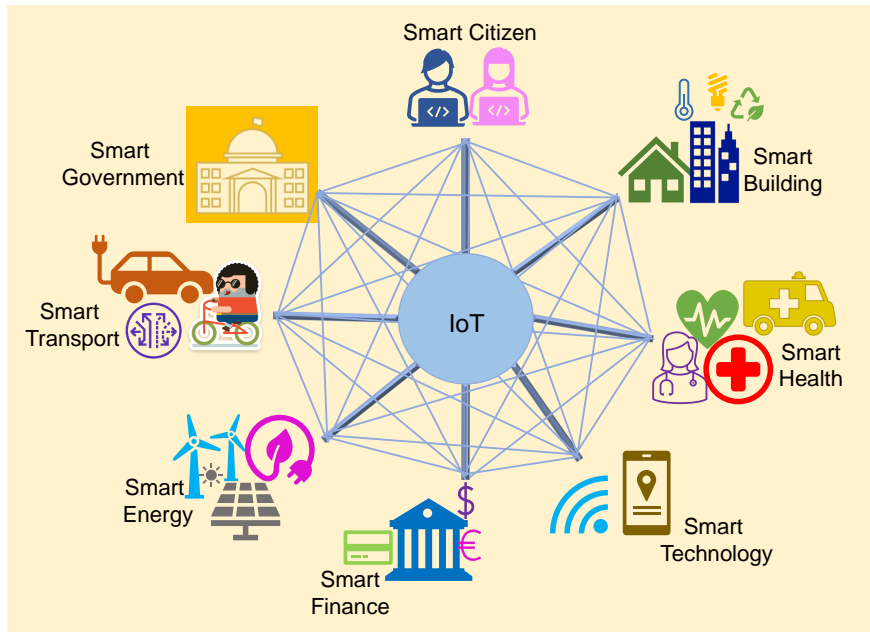


FIGURE 4.1. Components of smart city.

In this research, we present two [157, 154] universal digital identity system that can unlock the full potential of a smart city by connecting its citizens to all its amenities in a simple but efficient manner. Any facial recognition or authentication system is prone to deepfake attack. The proposed systems detect deepfake attacks with a high success rate. The final system also detects the presentation attacks. With this system, individuals may easily and effectively use the numerous smart city services that have been made available. This paper is an improved and extended version of the original work presented in [157].

4.1.1. Role of Digital ID in Smart City

In traditional cryptographic systems, secret keys are used to authenticate users. Often, users write down their secret keys, save them somewhere, share them with others, or simply forget them. Sometimes the keys are so simple or tied up with people’s life events that they are easy to predict. The authentication system collapses if the secret key is no

longer private.

On the other hand, biometric-based digital IDs are person-dependent and discrete as biometrics represent physiological or behavioral traits of a person. With such an ID, there is no need to preserve the confidentiality of the key because the users themselves are the secret keys.

Activities such as opening a bank account or availing of any age-restricted service require proper user verification. A digital ID can reduce the paperwork typically associated with such activities, giving citizens easy and efficient access to all the services provided by the smart city. The importance of using a digital ID in a smart city is multi-fold:

- (1) The digital ID system revamps the operational capacity of a city at a granular level.
- (2) Emergency medical services will be improved by introducing digital ID. If a critically sick person reaches a hospital without any traditional ID, a digital ID can save his lives. The doctor can access the patient's medical history and give proper medical care.
- (3) During any natural disaster, efficient verification of individuals' identities can be performed.
- (4) Digital IDs can facilitate improved administrative capacity for amenities around smart cities, e.g., banks, driver's licenses, retail, and transportation.
- (5) Online education is operating in parallel with the traditional brick and mortar schooling system. A Digital ID will offer a fair system with more flexibility to both students and educators.

4.1.2. Challenges of Digital Identification System

Building such a system is challenging in light of multiple considerations, including cyber security attacks, data privacy, security, and inclusion.

4.1.2.1. Vulnerability to Biometrics and Cyber Security Attacks

Biometric-based digital ID systems are prone to various biometric and cyber security attacks. As our proposed digital ID is facial biometric based, biometric attacks are our area

of concern.

- (1) *Presentation Attack:* The proposed digital ID verification system is reliant on accurate face recognition. Facial recognition software is vulnerable to face spoofing or presentation attacks. The vulnerable points of a digital identity system (DIS) are listed in Table. 4.1 [180]. The first two attacks on the table are presentation attacks.
- (2) *Deepfake Attack:* Another common vulnerability of a digital ID system is the deepfake attack. Deepfakes are AI generated fake images or videos that do not exist and can easily fool human eyes. It is a type of presentation attack [98]. It also poses a serious threat to facial recognition systems [127]. With the rapid progress of deep learning, deepfakes are gaining the potential to fool even the best facial recognition systems.

Face swapping through Face Swapping Generative Adversarial Network (FSGAN) [170] eases the creation of deepfakes as there is no need to train the FSGAN for hours with source and target images. It means deepfakes can spread more quickly and easily than ever before, as people with a basic knowledge of this technology are now capable of creating them.

- (3) *Indirect Attack:* In the Table. 4.1, attacks from row 3 to row 8 are indirect attacks. These attacks target the cyber security system directly rather than biometric attacks, as they target databases, or channels, or even the device itself. In this paper, indirect attacks have not been addressed as they fall outside the scope of this work.

4.1.2.2. Privacy and Data Security

Fig. 4.2 depicts the conditions of a biometric-based digital ID. Two major concerns are the security and privacy of information. Data should be secured and obtained only by the titled person. Data abstraction should also be followed, where people with different levels of authorization access different levels of data. As digital ID is related to someone's biometric data, storing this data safely is another important factor. Hence, the design approach needs

to be secure-and-private-by-design [161].

TABLE 4.1. Vulnerable Points in DIS.

Serial No.	Type of Attacks	Real World Scenarios
1	Present fraud face biometrics at the camera.	Using a 3D face mask, 2D photo, a video clip of the attacked face.
2	Submit saved digital photo of face instead of using camera.	Resubmit earlier photo.
3	Trojan Attack during feature extraction.	Selection of predefined features by hacker.
4	Alter feature set after extraction.	When face matching is done at a different place than feature extraction, change of some packets in TCP/IP stack remotely.
5	Attack the matcher.	Matcher shows intruder defined scores.
6	Alter the database.	Any entry of the database can be changed.
7	Attack the channel carrying data from the database.	Intercept the channel and alter the data before it reaches matcher.
8	Change the final score.	Hacker can change the final result failing the FRS.

4.1.2.3. Exclusion

Ideally, everyone in the smart city should be enrolled in the digital ID system. However, there will be scenarios where, e.g., people may not want to be enrolled in the system; people may not be capable of providing the required biometrics due to physical disabilities; or they may not be digitally aware. In those scenarios, alternative traditional identification, e.g., a paper ID, is required.

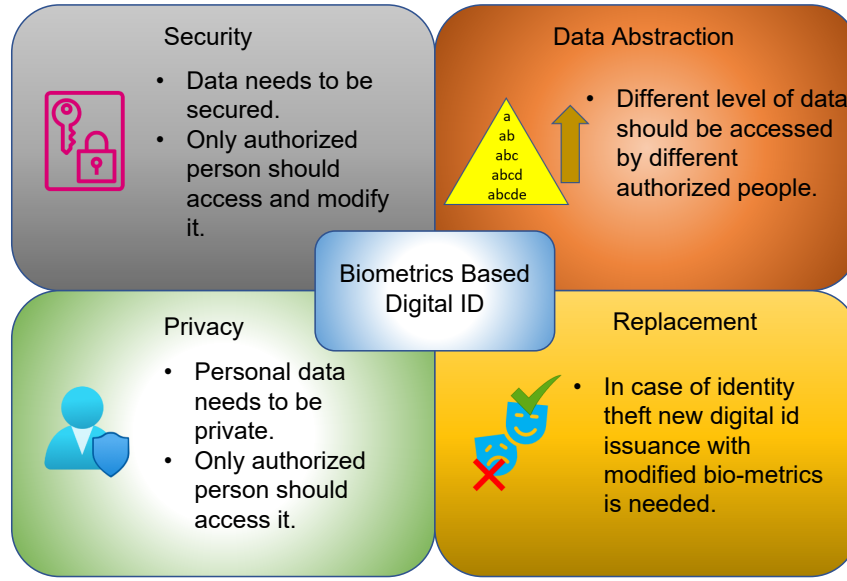


FIGURE 4.2. Mandatory requirements for digital ID in a smart city.

4.1.2.4. Data Privacy Regulation Policy

Data privacy regulation policies are different across the globe. Europe has the General Data Protection Regulation (GDPR), whereas North America has different regulations like HIPAA, FERPA, COPPA, and FCRA for different types of consumer data. There are many more such regulations worldwide, and any physical digital ID system for a city would have to ensure that it adheres to all rules and regulations surrounding consumer data privacy. As the approach presented in this paper is simply a proof of concept for a facial authentication based digital ID system, the application of data privacy regulations for the hypothetical smart city falls outside the scope of this paper.

4.2. Background

In the past few years, research on computer vision and pattern recognition has been boosted by significant advancements in deep learning techniques, new ways of thinking about parallel computing, and monumental developments in hardware. Because facial authentication is at the heart of iFace and iFace 1.1, this section discusses some of the most recent work on the main features of the proposed systems.

Our iFace system is based on biometric key on edge devices of an IoT system, we

summarize papers which address cryptographic key generation from face features and also biometric authentication systems in the IoT. For the iFace 1.1 system, three more areas - the FR system, face features-based authentication, and morphing attack detection (MAD)- have also been studied.

4.2.1. Cryptographic Key Generation

An entropy based method has been explored in [46] to regenerate a cryptographic key. Features are extracted from the images using entropy based method and Reed-Solomon ECC has been performed to generate the bio-key. Lookup tables have been created to regenerate the original key. In our work, we also use lookup tables, but the use and scope are different. An eigenfunction based face recognition method has been mentioned in [224]. It does not generate any key but tracks the user's head. Finally, it recognizes the face by comparing the traits between the user and data stored. A 128 bit key has been generated from a principal component analysis feature vector using thresholding and distinguishable bits with a right sequence number are updated in a lookup table [240]. Finally, the Reed-Solomon algorithm has been used to create an error correcting code (ECC). Symmetric DES and the generated key are used to encrypt any message. In the decryption stage, the reverse procedure is done. A key is computed by connecting several multi-bit keys generated from various threshold value [251]. An optimal threshold value has been chosen to reduce the authentication error. The methods mentioned above generate bio key based on face features but have not been implemented in limited-resource IoT devices.

4.2.2. Biometric Authentication Systems in IoT

A detailed survey has been made for face verification and authentication for IoT mobile devices in [77]. Another low complexity deep learning based face recognition method has been implemented in an embedded device [172]. A secure biometrics based end-to-end IoT solution has been mentioned in [100]. To increase the security, pairing-based cryptography has been used. A face recognition system, implemented in FPGA for digital forensics application, has been presented in [175]. A deep learning based method has been described in

[149] for an IoT-cloud setting with a tree-based cloud model for face verification. The edge part is optional for processing and filtering images. Our work fits in the same setting as this paper but with versatile scope, as our proposed method addresses the security part of the facial authentication system by computing the bio-key at the edge and by using an encoding key. During authentication, bio keys are compared in our method instead of images. This makes our method more robust.

4.2.3. FR Systems

In the early days of facial recognition, a holistic approach [233] was used, such as image projection on low-dimensional space [35], Laplacianface [97], and sparse representation [239]. In the early 2000s, more local features-based face detectors [142, 28] were introduced. However, from 2012 onward high-accuracy deep learning-based techniques have been predominant.

In 2014, DeepFace [212] transformed the direction of facial recognition techniques. An accuracy of 97.35%, close to human accuracy, was obtained with this 9-layer deep neural model on the LFW dataset [107].

Deep neural network-based techniques have used diverse architectures, different loss functions, and various image processing techniques. In the same year as DeepFace [212], another paper [207] performed face verification using high-level features and deep ConvNets. Face features from different face parts helped the model to achieve a higher accuracy of 97.45% on the LFW dataset [107]. Both works used a cross-entropy-based softmax loss. However, the later versions [101] and [103] of DeepID used a Euclidean distance-based loss named contrastive loss. Here, absolute distances between image pairs are calculated.

Another face recognition model is FaceNet [196], where a new loss, triplet loss, was used for feature learning and clustering. The method had an accuracy of 99.63%. Triplet loss is another Euclidean distance-based loss where the relative difference in distances between matching pairs is considered. Another important facial recognition system was proposed in [65]. Here, marginal loss was proposed for deep face recognition with a comparable accuracy of 99.48% on the benchmark LFW dataset. It minimizes the intra-class variation

and maximizes the inter-class distances simultaneously.

A pose invariant facial recognition technique was proposed using Disentangled Representation Learning GAN in [222]. This FR system has been evaluated for various illumination and angular positions of the face. Another competitive FR system is *CosFace* [232]. Large margin cosine loss (LMCL) has been introduced by redefining *Softmax loss* as a *cosine loss*. One of the state-of-the-art FR systems, *ArcFace* is presented in [64]. An Additive Angular Margin Loss has been proposed for face recognition. It is a highly accurate system. It achieved the highest accuracy of all the discussed FR systems on the LFW dataset.

After studying several of the aforementioned state-of-the-art FR systems, FaceNet [196] has been chosen as the FR system in the proposed digital ID. FaceNet [196] offers high accuracy as well as ease of application on the edge platform.

4.2.4. Facial Features-based Authentication Systems

In the last few years, various cloud-based, cloud-edge-based, and IoT mobile device-based face authentication systems have been researched in literature.

A detailed survey has been made for face verification and authentication for IoT mobile devices in [77]. In [214], a face verification system for a mobile device, from face registration to face verification, was proposed using light normalization and information fusion. However, no security measures were undertaken. Another work [100] proposed biometric-based security for IoT infrastructure using pairing-based cryptography.

A face verification technique for mobile devices is presented in [215]. The Viola-Jones detector has been used for face detection and subspace metrics for authentication. It has a low error rate. But no security measures were implemented. Another facial feature based active authentication technique for mobile phones has been proposed in [75]. A short video is used as the input of the face verification system. The detection rate showed high accuracy when authentication and enrollment were done from the same session videos. It is not suitable for real-time use where different session data needs to be compared. Similarly, a face authentication system for mobile devices is implemented in [87]. Here, face detection has been performed with Haar-like features and the AdaBoost algorithm. Face authentication,

on the other hand, has been done with a local binary pattern.

Another mobile-friendly deep learning based face detector has been proposed in [194]. Various illuminations and extreme poses were considered. Without CUDA, mobile GPUs have been used to implement deep neural network models. [208] presents a fingerprint and face template based method. The face verification is SVM based, whereas the fingerprint verification is minutiae based. According to the authors, the “Secure sketch” cryptography and geometric translation make the method forgery-free. An enhanced biometric capsule-based authentication method was proposed in [177]. MTCNN [250] has been used for face detection and FaceNet [196] has been used for face feature extraction. A deep learning-based face verification method has been proposed in [149] for an IoT-cloud setting. The face verification part is done by a tree-based cloud model. The edge part is optional for processing and filtering images.

The majority of the aforesaid facial authentication systems lack security measures. However, for any facial authentication system, security and privacy are the two critical criteria that need to be fulfilled. These two factors have been prioritized in the design of the proposed digital ID.

4.2.5. Attack Detection

In this subsection, we discuss papers addressing various attacks on facial recognition systems. User liveness has been addressed in a majority of the papers.

For face authentication, an acoustic sensor based liveness detection method has been proposed in [47] with an accuracy of 96.02% and a false alarm rate of 3.97%. It uses the unevenness of the stereo structure of a real face to check the liveness of the user. Another liveness detection method has been proposed in [48] using photoplethysmograms of two simultaneous videos of the face and fingertips.

Some papers also focus on deepfake detection systems. In our earlier papers [155] and [156] deepfake videos have been detected. These systems, which are based on convolutional networks, have a high level of accuracy. Dynamic lip movement analysis has been done in [244] to detect deepfake attacks. In [69], the potential threat of face swapping to electronic

Know Your Customer (eKYC) has been discussed, and a detection system has been proposed. Another IoT-friendly deepfake detection method has been described in [104]. The LightGBM classifier has been used to classify the images based on features from the Gray Level Co-occurrence Matrix (GLCM). In [243], an anti-spoofing facial recognition system has been proposed. COTS RFID tag array has been used to extract biometric features of the face and 3D geometry. 95.7% success rate is achieved with 4.4% EER. Most of the papers in this area also focus on a specific attack, but not in the context of facial authentication systems.

The discussion above shows that in the last few years, a number of state-of-the-art FR systems, different cloud/edge-cloud/mobile based facial authentication methods, and different attack detection techniques have been developed. Regardless of all the state-of-the-art methods, no end-to-end facial authentication system that can be used for implementing a digital ID system for smart cities has been proposed. Our objective is to propose a proof-of-concept of a viable but simple facial authentication based digital ID system for smart cities with high success rates in detecting attacks and authenticating smart city citizens.

4.3. iFace: Proposed Biometric-based Digital ID for Smart Cities

4.3.1. End-to-End System Level Architecture

The proposed biometric based digital ID system consists of a layered architecture which is distributed among edge and cloud computing platforms, starting from the end user to the cloud server, as shown in Fig.4.3. This four layered structure consists of the following:

- (1) Layer-1: It consists of the Smart Citizen with digital ID, various types of cameras from different smart city stake holders, and an input device to provide user ID. When digital ID verification of a person is necessary, these cameras take a photo and send it along with the username to Layer-2. Cameras can be smart phone cameras or any cameras installed as the end device. The user ID can be inserted using the keypads at the end device itself.
- (2) Layer-2: Edge Computing Platform works as Layer-2. The photo and username from layer 1 come to this layer. As both layer 1 and layer 2 are at the same location,

no transmission of biometric data or username happens over open channels at this stage. This alleviates the necessity of encryption of the biometric data at this point. Most processing and computing steps are performed here. Bio keys are generated and encoded here.

- (3) Layer-3: It comprises of a cloud computing platform. It is connected to edge devices through various long range technologies like *4G*, *LTE*, etc. The data is encoded and sent to the cloud. Layer-3 is mainly used for storing large amount of bio-metric data and usernames.
- (4) Layer-4: Smart city stakeholders are the key components of layer 4. Once the bio key is authenticated in layer 2 with the information from layer 3, the digital ID is verified and smart city application is accessed through its API.



FIGURE 4.3. End-to-end system level framework for digital ID system in a smart city.

4.3.2. System Overview

We propose a digital ID system for smart cities which consists of two phases:

- Enrollment or registration of new users.
- Authentication of existing users.

The process of enrollment of a new user is shown in Fig. 4.4. In the enrollment phase, a unique username is issued to the new user after verifying the existing government issued ID. Bio-metric facial features of the person are extracted from a neutral frontal face (NFF) image taken by the end device camera. Bio keys are then generated from the image, are encoded, and saved in the cloud server along with the username.

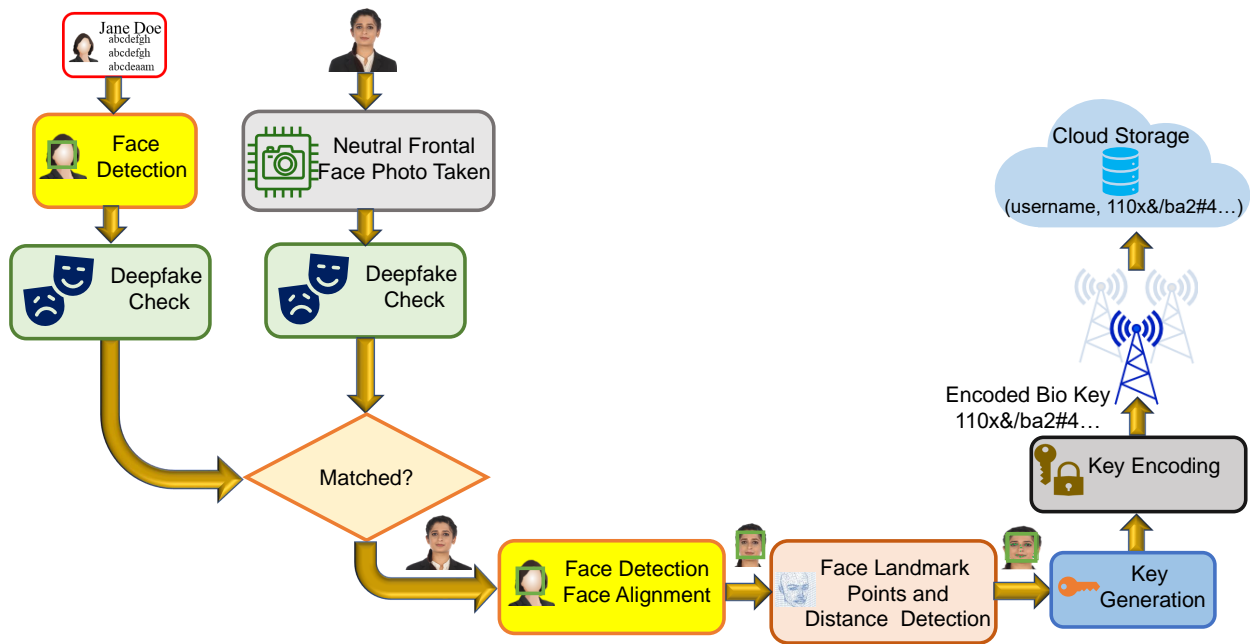


FIGURE 4.4. Registration of a new user in iFace.

During the authentication phase shown in Fig. 4.5, the user provides his username and an NFF photo is shot by the end device. Once the photo is taken, bio keys are generated. To avoid any discrepancy in input data, NFF images are taken in both phases. But, as these photos are taken in unconstrained environments, it is highly unlikely that a person will have the exact same photo for all occurrences. If two photos are not exactly the same, they will generate different bio keys. Those bio keys are not completely different but little variations

will be present. Our system can accommodate a certain level of such modifications while generating bio keys using Reed Solomon error correcting codes (ECC). If the system can generate bio key at registration from the bio key at authentication phase, it verifies the person through digital ID and gives access to the smart city facility that the user wants to access. If the two faces do not match, the user does not get access.

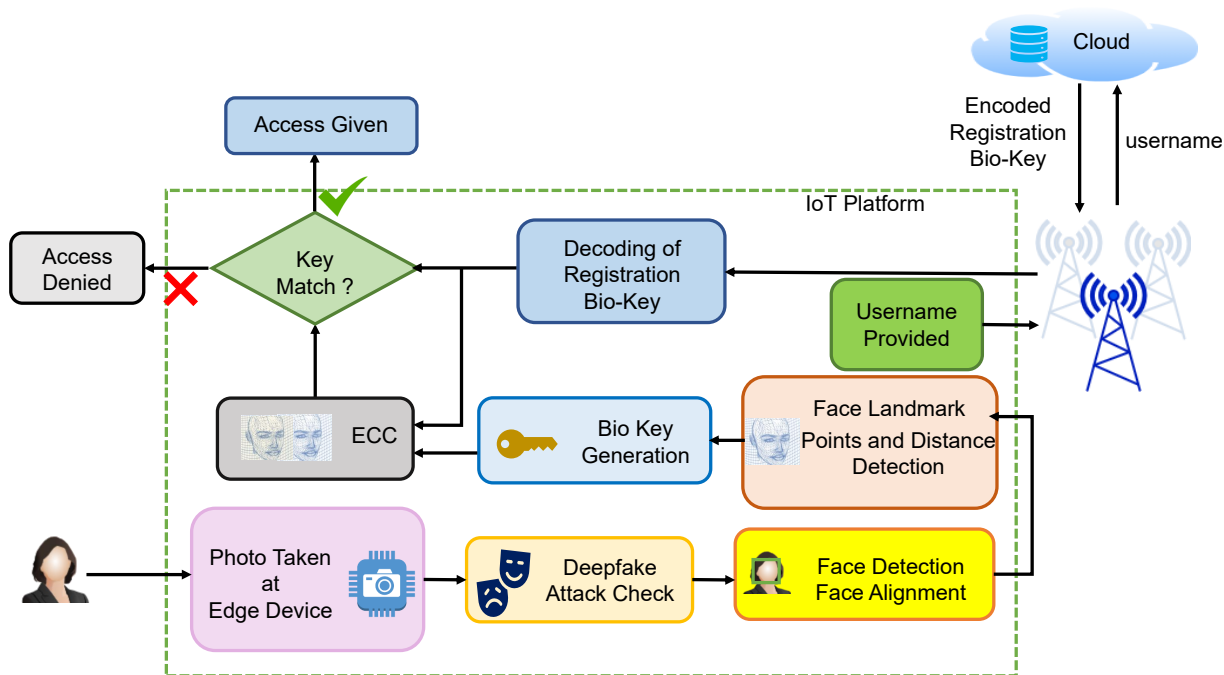


FIGURE 4.5. Authentication of existing user in iFace.

4.3.3. Deepfake Attack Detection

Each time an user uses a facility a NFF photo is taken and is checked for any deepfake fraud.

4.3.3.1. Method

An NFF photo of the user is taken through the camera attached to the edge device. The taken photo is in the RGB color space. The photo is checked for deepfake detection attacks. We follow the procedure from our previous work [155]. However here we use MobileNetV2 [193] as the feature extractor and a softmax layer as the classifier.

To detect the deepfake images, a CNN based method as shown in Fig. 4.6 is used. An existing CNN, suitable for facial recognition system in edge devices of smart cities, has been used as the feature extractor of the network. The used CNN is based on depthwise separable convolution.

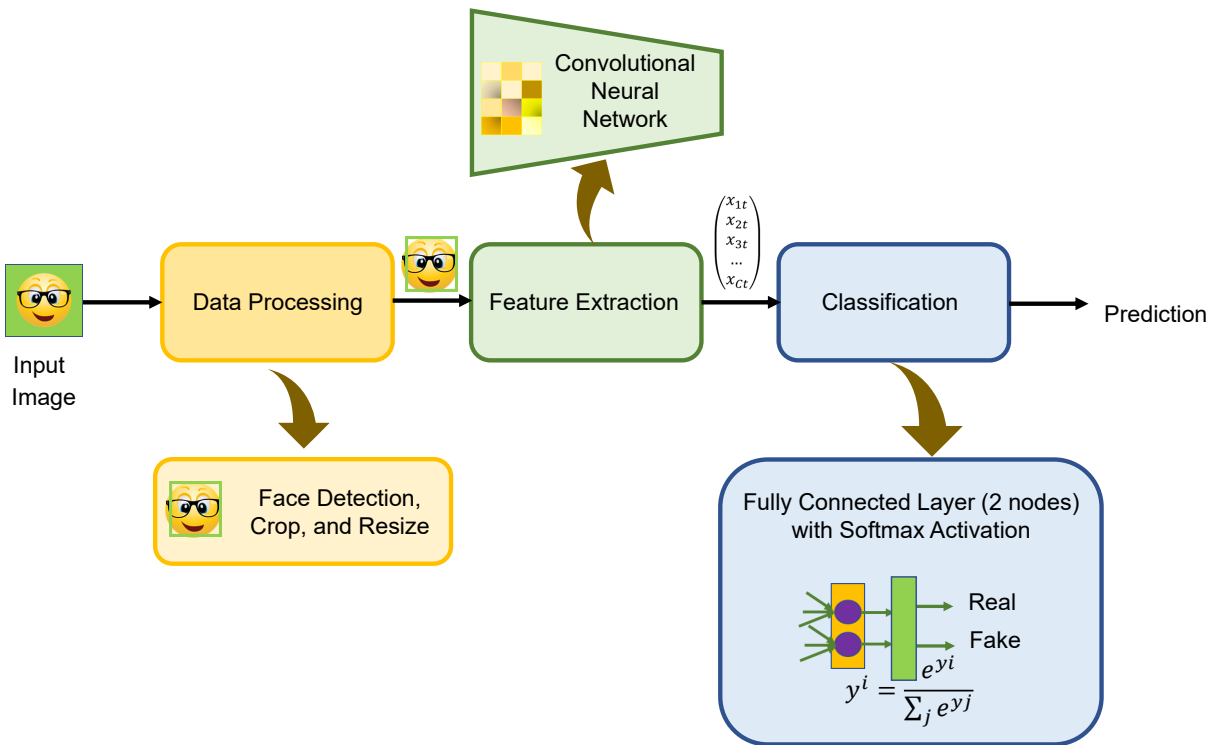


FIGURE 4.6. Deepfake image detection method.

Depthwise separable convolution leverages the depthwise and pointwise filters, by performing depthwise convolution prior to pointwise convolution. The cost of depthwise separable convolution is much smaller than standard convolution. Each depthwise convolution filter is applied on each input channel or depth. A linear combination of depthwise filter output is computed by applying pointwise convolution filters. Here we use MobileNetV2 [193] as the feature extractor and a softmax layer as the classifier. MobileNetV2 employs depthwise separable convolution. Here linear bottlenecks have been used between the layers. Shortcuts connect the bottleneck layers instead of non-bottleneck layers as in ResNets [95] and make it faster and accurate. The last layer of the CNN is removed. A fully connected layer with *softmax* activation and 2 nodes has been used as the classifier. We fine tune a

pre-trained MobileNetV2.

Before training our network, we processed the data as per our requirements. First, we detected the face using dlib’s 68-landmarks detector [124]. Then faces are cropped from the image frame, resized to 224×224 , and normalized.

4.3.3.2. Experimental Validation

- (1) *Dataset:* Part of the FaceForensics++ [185] DF dataset has been used to train and evaluate the detection method. It has videos of different compression levels. We used the deepfake videos of compression level $c = 40$ for training and evaluating our model, as it represents a realistic scenario for social media. The dataset details are given in Table 4.2.

4 sec videos are clipped from the original and manipulated videos with compression level $c = 40$. Then frames are extracted from each compressed video with no decompression using FFmpeg [37]. We extracted image frames from those videos at a rate of $24fps$. We then detected the faces and cropped the faces from each frame. Finally All the frames are normalized and resized as per the input of MobileNetV2.

TABLE 4.2. Dataset for Deepfake Detection in iFace System.

Division	Real	Fake
Train	61,506	63,452
Valid	7,688	7,931
Test	7689	7933

- (2) *Training Protocol:* To train the network following protocol has been followed.
- *Transfer learning* has been used. We fine tuned a pre-trained MobileNetV2. We trained the last 40 convolutional layers.
 - The network has been optimized with Adam [125] optimizer of learning rate 0.0001 and other parameters set to default values.
 - The network has been trained for 30 epochs.

- No data augmentation has been done.

(3) *Implementation:* We implemented our work in Keras [51] with the TensorFlow [26] backend. FFmpeg [37] is used to clip the videos. For training we used a Tesla T4 GPU with 25GB RAM. A GeForce RTX 2060 laptop is used to evaluate the model.

4.3.3.3. Results

Table. 4.3 shows the classification report for the deepfake detection method of iFace. It is evident from the result that there is room for modification in the method as it shows 91% accuracy. Also, here we trained the last 40 layers of the model which might be a problem during generalization of the method. Hence, in the improved iFace 1.1, we took a different strategy to train the model.

TABLE 4.3. Classification Report for Deepfake Detection of iFace.

Test Images	Precision (%)	Recall (%)	F1-score (%)
7689 Real	95.0	88.0	91.0
7933 Fake	89.0	95.0	91.0
Macro Average	91.0	91.0	91.0
Weighted Average	91.0	91.0	91.0
Total 15,622	Accuracy (%)	91.0	

4.3.4. Digital ID System

Once the photo is passed through the deepfake check, it is ready for biometric feature extraction.

4.3.4.1. Methods

(1) *Biometric Features Extraction:* The process of biometric face features extraction is shown in Fig.4.7 and Fig.4.8. First the face is detected from the NFF photo with the

dlib [124] library using Histogram of Oriented Gradient (HOG) and linear Support Vector Machine (SVM) in Fig.4.7. The reasons behind choosing HOG based dlib library are the following:

- As we detect the face at the edge where resources are limited, HOG based dlib face detector is the best choice.
- It is the fastest and lightest model among face detectors and suitable for edge environment.
- Frontal photo is considered for digital ID. We do not expect a person, when verifying with digital ID, will show a side view of the face. So, HOG based dlib works perfectly well in our used case scenario.
- This model works better with CPU. No GPU is required to detect faces. This characteristic makes it a good fit for using at an edge device.

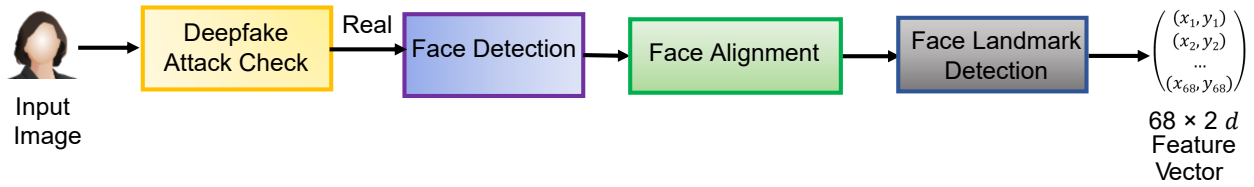


FIGURE 4.7. Facial landmark points detection workflow.

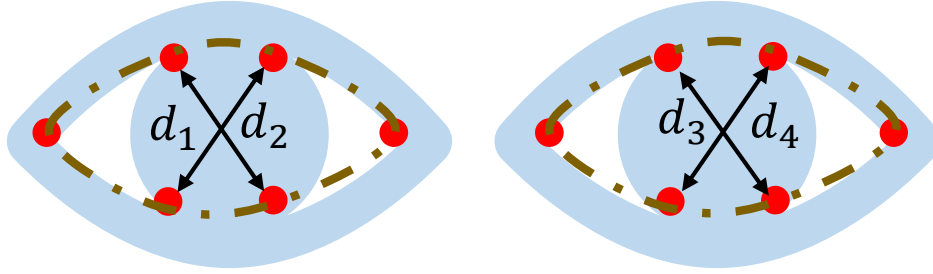
Once the face is detected, it is aligned with OpenCV. It will limit some positional discrepancies of two photos taken at different times. 68 facial landmarks points related to jaw, both eyebrows and eyes, nose, and mouth are then detected using the HOG based dlib face detector [124]. The (x, y) coordinates of 68 facial landmark points make a 68×2 feature vector $\mathcal{F}1$ in Fig. 4.7.

Another feature vector $\mathcal{F}2$ of dimension 1×6 is formed with $d_1, d_2, d_3, d_4, d_5,$ and d_6 as in Eq. 4.1:

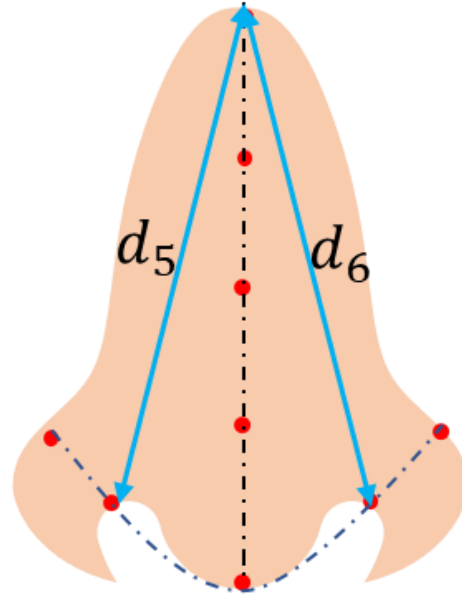
$$(4.1) \quad \begin{aligned} d_j &= d_{1j} \\ d_{1j} &\in \mathcal{F}2^{(1 \times 6)}. \end{aligned}$$

d_j are calculated as in Fig. 4.8, by calculating the diagonal distance of both iris and

the nose length. As the bio-metric data is unique for each individual, the feature vector is *sui generis* too.



(a) Calculation of diagonal distance of iris



(b) Mean distance calculation

FIGURE 4.8. Facial distance calculation.

(2) Biometric Key Generation: In this section, a novel bio key generation technique is proposed. First, a binary key is generated from the feature vector $\mathcal{F}1$, as shown in Fig. 4.9. The process consists of three steps:

- First, the 68×2 dimensional feature vector is reshaped to 1×136 as $(x_1, y_1, x_2, y_2, \dots, x_{68}, y_{68})$.
- Second, a unique threshold value d_m is calculated using Eq. 4.2, as shown in

Fig. 8(b):

$$(4.2) \quad d_m = d_5 + d_6.$$

- Binarization of the feature vector $\mathcal{F}1_b$ is performed by comparing each element f_i of the feature vector $\mathcal{F}1$ to the threshold value d_m following Eq. 4.3:

$$(4.3) \quad f_{bi} = \begin{cases} 0, & \text{if } f_i < d_m \\ 1, & \text{if } f_i \geq d_m \end{cases}$$

Finally, $\mathcal{F}1_b$ and $\mathcal{F}2$ are concatenated to form the unique feature vector or final bio key \mathcal{F}_{io} following Eq. 4.4:

$$(4.4) \quad \mathcal{F}_{io} = \mathcal{F}1_b + \mathcal{F}2.$$

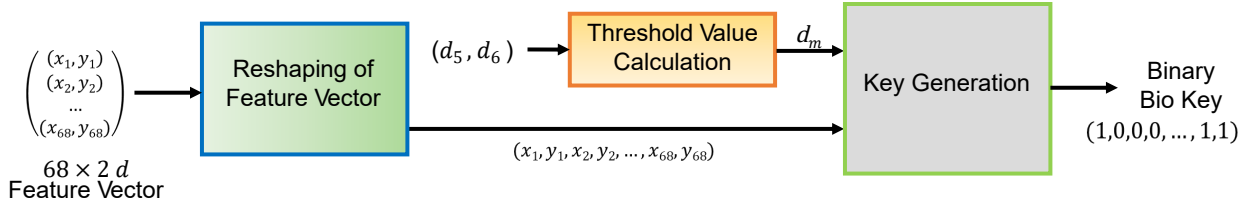


FIGURE 4.9. Binary key generation from face landmark points.

- (3) Error Correction: In our proposed digital ID system, we use \mathcal{F}_{io} as the biometric feature to authenticate a person. \mathcal{F}_{io} is robust against lighting and is also unique to an individual. However, getting the same picture of a person at various times and by various cameras of a smart city is almost impossible. These variations in pictures can alter the bio key at a certain percent. To accommodate these variations and avoid false rejection ratio, we use Reed Solomon (RS) codes [182] to correct the errors. The face matching process is shown in Fig. 10(a) and Fig. 10(b).

In the registration phase, \mathcal{F}_{io} is encoded with Reed Solomon codes and saved in a lookup table in the cloud server. The lookup table comprises of two columns - username U and encoded bio key \mathfrak{F}_{io} as shown in Fig. 10(a).

During the authentication phase, as shown in Fig. 10(b), the user provides the username which finds the corresponding encoded bio key \mathfrak{F}_{io} in the lookup table. Then we split \mathfrak{F}_{io} in original input \mathcal{F}_{io} and error correcting code ECC . ECC is then combined with \mathcal{F}_{mod} , collected at this stage from the authentication photo. It generates encoded authentication \mathfrak{F}_{mod} . If the photo at this stage differs from the photo taken at registration, it gets corrected \mathcal{F}'_{io} with the Reed Solomon decoding module. If the decoding module is able to generate the original \mathcal{F}_{io} at this point then the faces are matched and the user gets access to the specific facility of smart city where he used his digital ID.

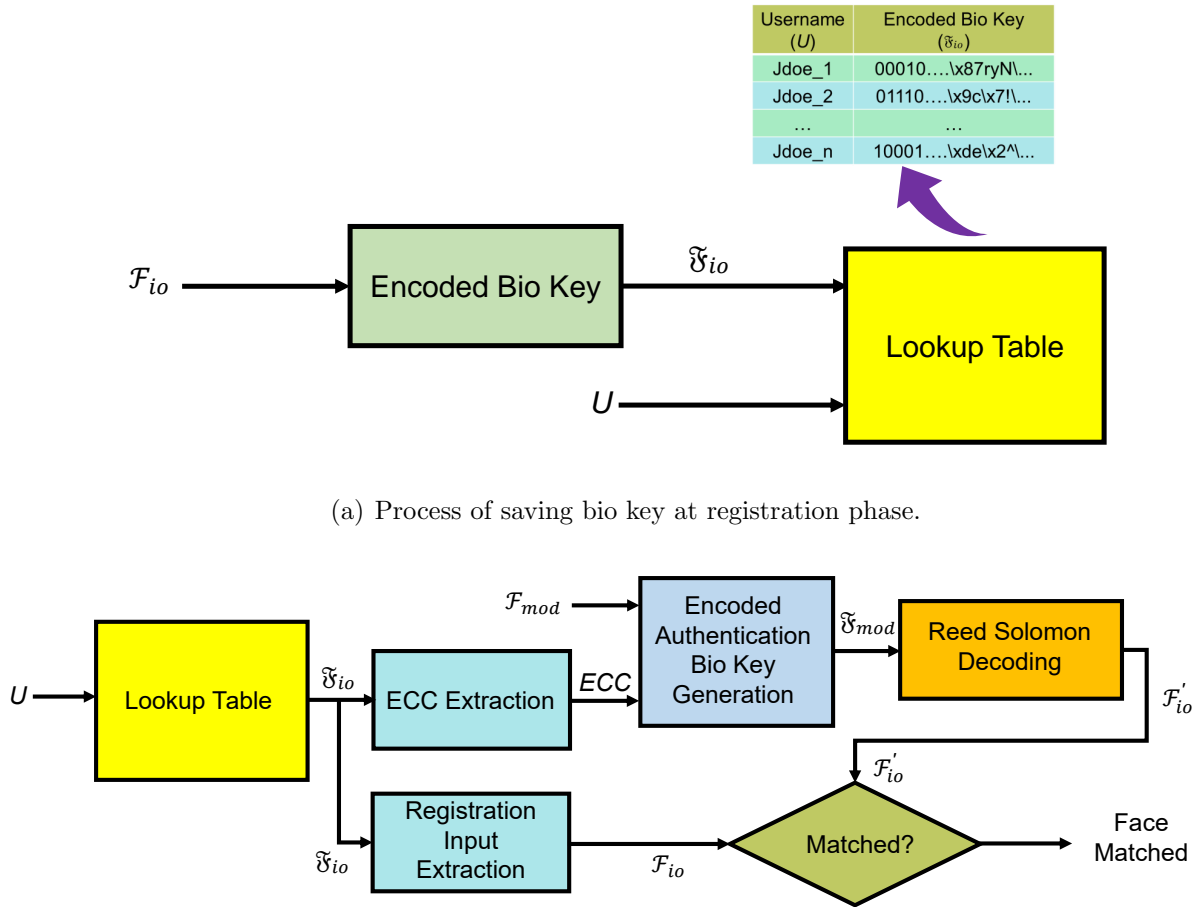


FIGURE 4.10. Face matching workflow.

An attacker can get the encoded bio key from the lookup table in cloud, but they will not be able to impersonate any person as the authentication process is being performed at the edge with the presence of user. That makes the system robust.

4.3.4.2. Experimental Validation:

- (1) *Dataset:* For evaluation of our proposed digital ID system, three different datasets have been used. The dataset details are summarized in Table 4.4. Neutral frontal face (NFF) without any occlusion is required for our system. Various frontal face datasets are publicly available but very few datasets contain NFF. For this reason, we take 250 neutral faces or close to neutral faces from CelebA [148] dataset to form *Dataset-1*. The second dataset *Dataset-2* is a neutral face dataset from Kaggle [164]. However, both datasets contain only one NFF image for each individual. It does not fully evaluate our digital ID system. These datasets result in 0% False Acceptance Rate (FAR) and 0% False Rejection Rate (FRR) but this is not a fully correct evaluation of our method. A rightful user with a different image taken at authentication should be tested too. Finally, we test the performance of our digital ID system with a customized dataset *Dataset-3* of total 60 images of 30 individuals, mostly celebrities and political figures collected from Google search. Two images are collected for each person - one image for enrollment and the other image for authentication. Most of the tested images are neutral but some are with a slight smile or with mouth open. For each registered user, the other 29 users' images have been used as impostor images resulting in 900 tested combinations during authentication.
- (2) *Implementation:* We implement our proposed digital ID system in Python using a GeForce RTX 2060 laptop with a 6GB shared memory of total 16GB memory. We evaluate our system with the three datasets mentioned above. For deepfake detection we use [185] for both training and testing purpose. The message length during error correction of encoded message is 148. 4 bit RS codec has been used to avoid intruders.

TABLE 4.4. Dataset Details for iFace Verification.

Dataset Name	Source of the Dataset	No. of Images
Dataset-1	CelebA[148]	250
Dataset-2	Frontal Faces Neutral Expression 95 Landmarks[164]	240
Dataset-3	Internet	60

4.3.4.3. Results:

The results of the experiments for three different datasets are shown in Table. 4.5 and Fig. 4.11.

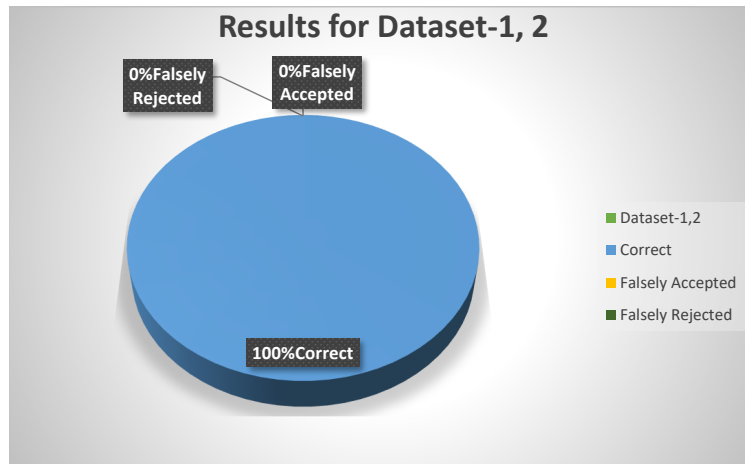
TABLE 4.5. Performance Evaluation of iFace.

Dataset	No. of Testing Images	No. of Cases Authenticated		
		Correct	Falsely Accepted	Falsely Rejected
Dataset-1	1000	1000	0	0
Dataset-2	1000	1000	0	0
Dataset-3	900	875	0	25

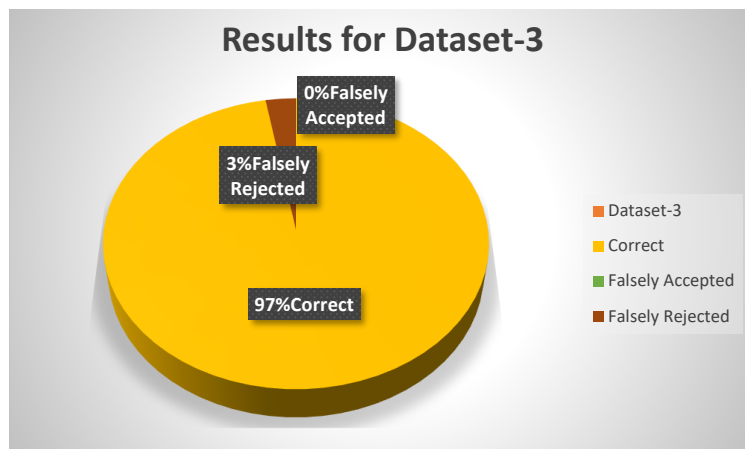
For Datasets 1 and 2, as there is only one image per user, the same image has been used for registration and authentication. As a result, FRR is 0%. For a specific user, we declare an impostor set containing other users' images. Here 0% FAR means no one other than the user is given access for that specific user ID. So, no impostor can access the user data. These results are included in this paper to demonstrate that impostors cannot access the tested user's account.

For Dataset 3, we use different images for registration and authentication. Not all images fulfill the exact criterion of neutral frontal face (NFF). As a result we see 25 images

are falsely rejected even if they are rightful people. This generates an FRR of 2.77%. The accuracy of the deepfake attack check in our case is 91%.



(a) For Dataset 1 and Dataset 2.



(b) For Dataset 3.

FIGURE 4.11. Performance evaluation of iFace for different datasets.

There are certain scenarios which have not been addressed in this work but will be considered in future work:

- If the person looks considerably different from the photo taken at registration, the system can not authenticate.
- Heavy eye make up like smokey eyes can generate a false rejection.
- Identical twins scenario has not been considered.

4.3.5. Discussions

In this work, we proposed iFace, a biometric-based end-to-end digital ID system of smart cities. Our system can detect certain deepfake attacks. It does not allow impostors to access users data. However, the accuracy of the deepfake detection method is not so high. *Presentation attacks* have also not been addressed. Authentication of an user involves computation spread in an edge-cloud environment.

4.4. iFace 1.1: Biometric-based Improved Digital ID System for Smart Cities

iFace 1.1 is the improved version of iFace, the digital ID system for smart cities. Once facial authentication is performed in iFace 1.1, residents of the smart city will be able to use different city services.

4.4.1. System Overview

iFace 1.1 works in two phases as iFace does:

- Registration/Enrollment Phase
- Authentication Phase

4.4.1.1. Enrollment Phase

The registration process of a new applicant is shown in Fig. 4.12. In this phase, an existing government-issued ID is checked to avoid any forgery, and then a unique username is provided to the new applicant.

An edge device camera takes a neutral frontal face (NFF) image. Next, the face is detected in the image by the face detection (FD) module. The photo is then checked for deepfake and presentation attacks in the Attack Detection (AD) module. If the AD module does not verify the photo, law enforcement authorities are notified of possible fraud. For legitimate applicants, a facial embedding is extracted from the detected face in the photo. The embedding, along with the username, is added to a *lookup table* on the cloud server of the smart city. Data encryption may be added at this step.

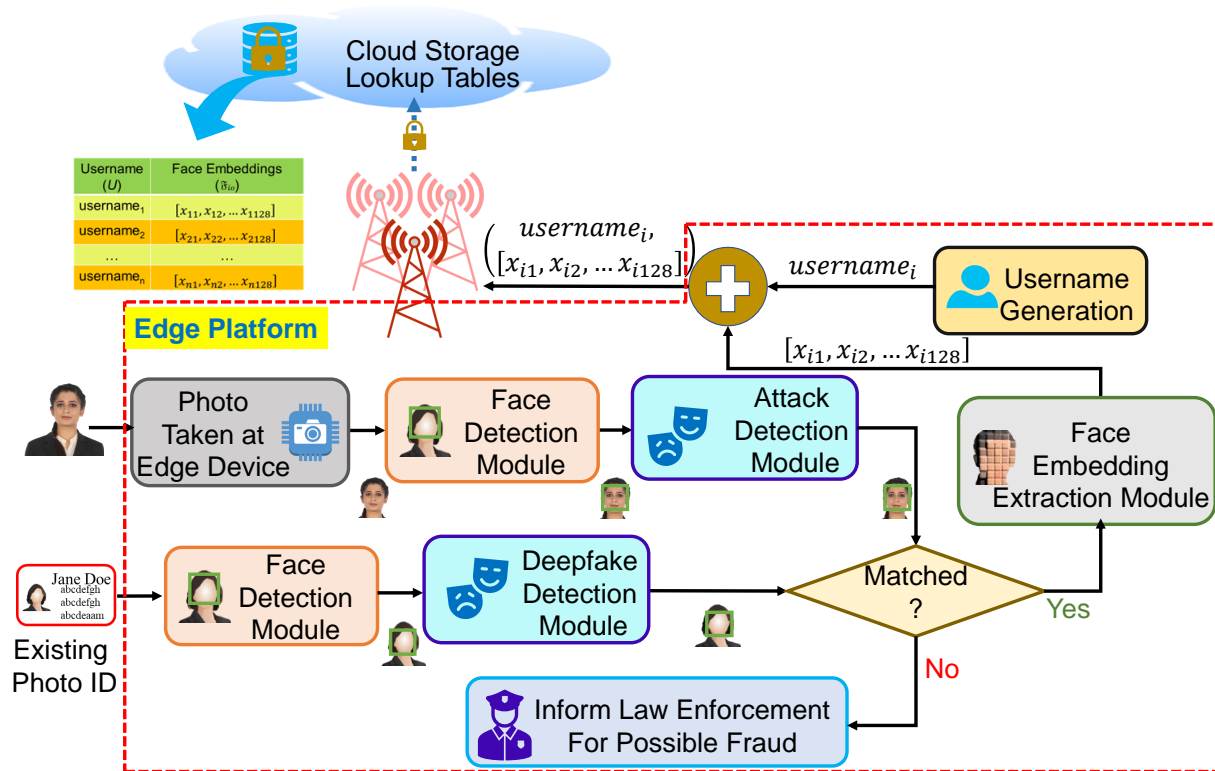


FIGURE 4.12. New user registration in iFace 1.1.

4.4.1.2. Authentication Phase

Authentication is done on the edge, as in Fig. 4.13. The user provides their username, and a photo is captured by the edge device. Once the face is detected in the FD module, it is checked for any possible attack in the AD module. Next, the facial embedding is extracted, and the system predicts the username associated with the face embedding. If the predicted username matches the input username, access to the facility is granted; otherwise, it is denied. The authentication process is performed on the edge device. No facial embedding data leaves the edge device, making the authentication process more secure.

4.4.2. System Modules

The system pipeline consists of five modules. Each module serves a specific purpose. In this section, various modules are discussed in details.

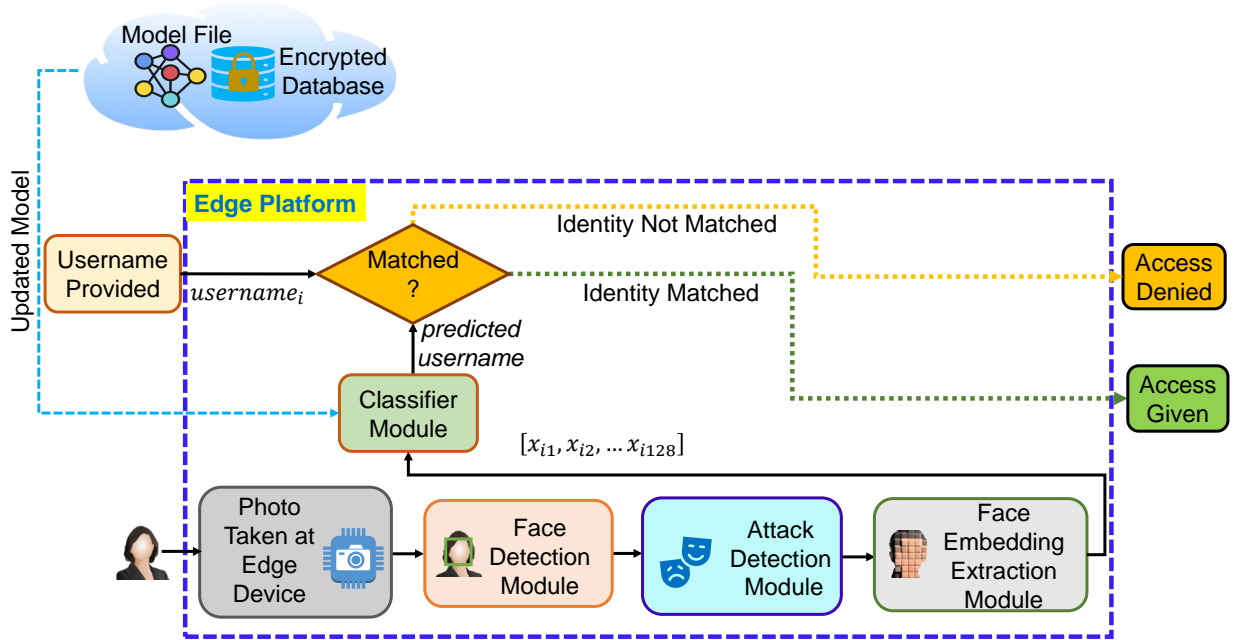


FIGURE 4.13. Authentication at edge in iFace 1.1.

4.4.2.1. Username Generation (UG) Module

The UG module is an important module where unique and user-specific usernames are generated during registration. Fig. 4.14 shows a sample username generation module workflow. During registration, the applicant provides their name. A unique username is generated as per the sample workflow.

During enrollment, the generated username and facial embedding extracted from the photo are sent to the cloud data storage and added to the *lookup table*. Data encryption should be used in each step to make the process more secure. Fig. 4.15 shows a sample *lookup table* before encryption. The stored data is used to train the classifier from time to time.

4.4.2.2. Face Detection (FD) Module

Accurate face detection is the first step of face authentication. There are mainly two types of state-of-the-art face detection methods: deep neural network based and handcrafted features based or machine learning-based. Deep neural network based face detectors, e.g., Multitask Cascaded CNN [250], RetinaFace [63], Fast RCNN [82], Faster RCNN [184], Mask

RCNN [94], YOLO [181], and SSD [146], have emerged as successful face detectors. They are more accurate and robust. However, the majority of them are heavy and poorly suited for deployment on an embedded device.

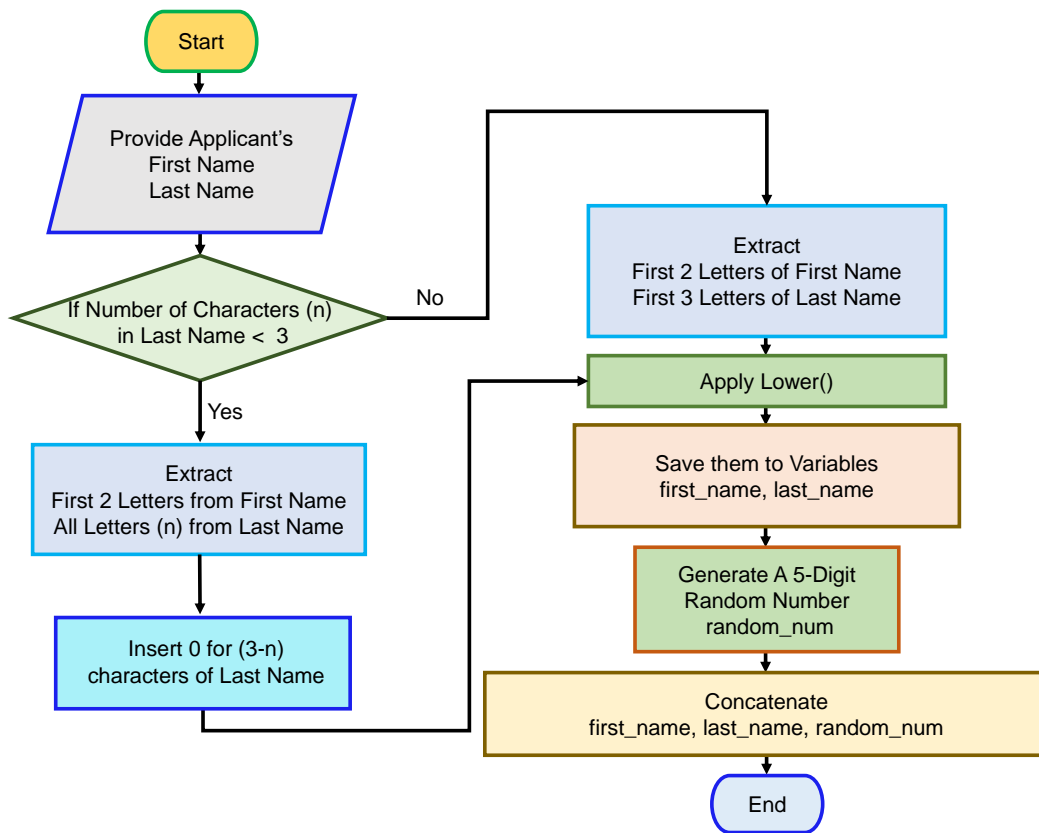


FIGURE 4.14. Username generation module workflow.

In our work, a lightweight, fast, and efficient model is required. Hence, we chose a machine learning-based face detector over a deep neural network-based one. For security and data integrity purposes, the face detector should not work under any face occlusion. Face detection from the frontal face photo is required. Hence, the Viola-Jones Haar Cascade face detector [229] has been chosen as the face detection module from [28, 229, 230, 59]. It detects the face from the photo. It is tiny in size ($\sim 1\text{MB}$) so a good fit for an IoT environment where resources are limited. It does not work under conditions where the face is occluded, which is a mandatory condition of the system to avoid any fraudulent activity. The Dlib HoG [124] face detector also works with mostly frontal faces.

Lookup Table	
Username	Face Embedding
abcde02846	$[x_{11}, x_{12}, \dots, x_{1128}]$
fghij96784	$[x_{21}, x_{22}, \dots, x_{2128}]$
...	...
klmn069001	$[x_{n1}, x_{n2}, \dots, x_{n128}]$

FIGURE 4.15. Sample lookup table.

But, the extracted face by Dlib HoG mostly excludes the forehead of the detected face, as shown in Fig. 4.16. It is not desired for face authentication purposes. Therefore, a frontal face image is necessary each time for security reasons. But for practical implementation, an alternative state-of-the-art suitable method can be used.

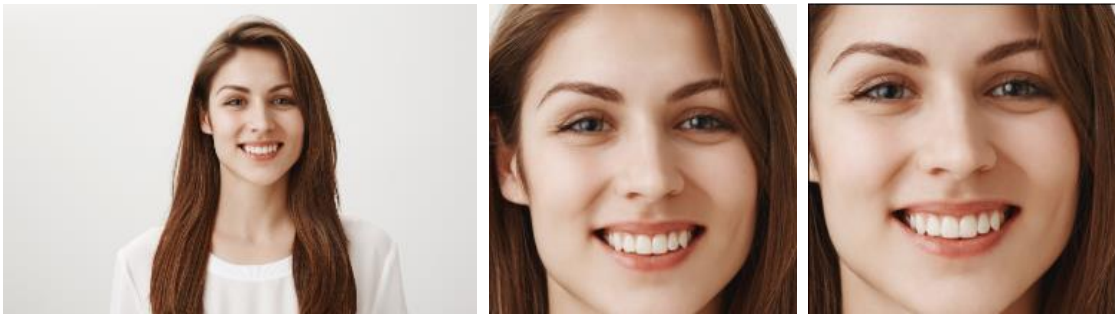
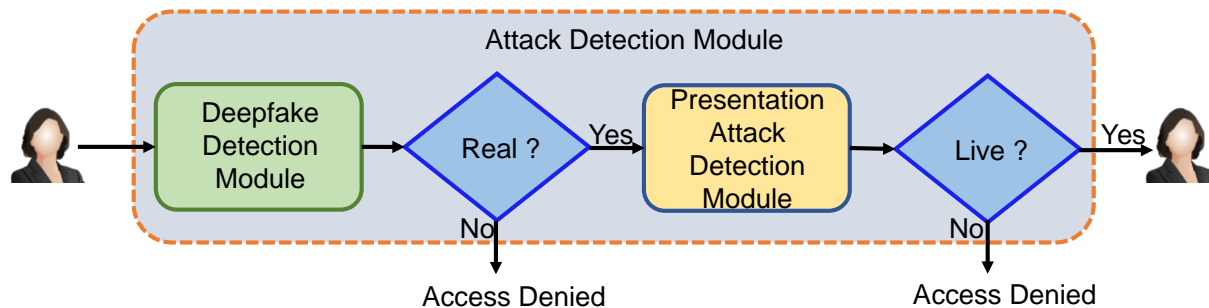


FIGURE 4.16. Extracted faces using Haar Cascade (middle) and Dlib HoG (right most) face detectors from original face (left most) (photo courtesy: Microsoft Power Point).

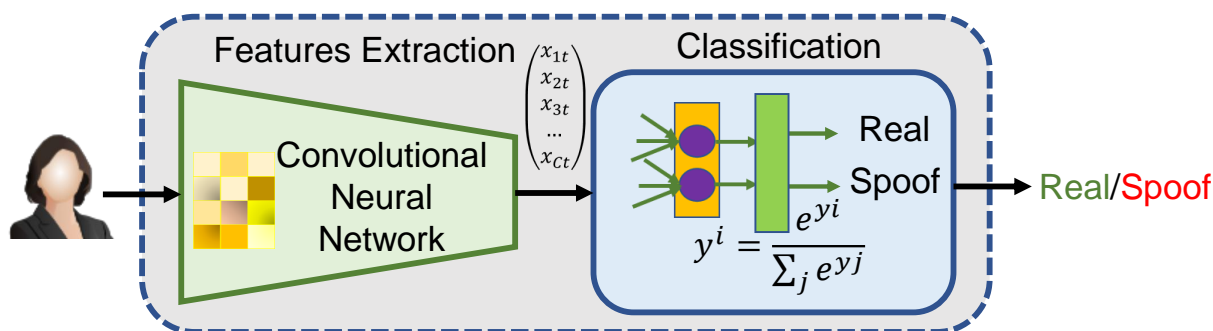
4.4.2.3. Attack Detection (AD) Module

The attack detection module is the next module in the digital ID system pipeline. It checks for various presentation attacks. A person with ill intention can gain access to any facility in a smart city by fooling the facial recognition systems. A spoof image or non-living object can be presented to the camera of the edge device instead of a live person. To detect

such an attack, the neutral frontal face photo of the user, taken through the edge device camera, is passed through the AD module as in Fig. 17(a). The AD module comprises of two subsystems—the Deepfake Detection (DD) and Presentation Attack Detection (PAD) modules.



(a) System level diagram of attack detection module.



(b) Detection process.

FIGURE 4.17. Attack detection module.

- (1) *Deepfake Detection (DD) Module:* The deepfake detection (DD) module is used to detect if there is a deepfake attack on the system. Ideally, the module should be trained to detect any type of deepfake. However, for the purpose of the simplicity of iFace 1.1, the model has been trained to only detect deepfakes, generated by face swapping techniques by FSGAN. More research on deepfakes is needed to propose a generalized deepfake detection module.

Before using any photo taken during registration and authentication of the users, deepfake attack is checked for avoiding any forgery.

- (a) *Method:* An NFF photo of the user is taken through the camera attached to the edge device. The taken photo is in the RGB color space. The photo is checked for deepfake detection attacks as in [102]. Here, also we use the same CNN structure, MobileNetV2 [193] as the feature extractor and a softmax layer as the classifier. *Transfer learning* has been used. However, instead of training last 40 layers of MobileNetV2, different protocol has been followed. Before training our network, we processed the data as per our requirements. First, we detected the face using dlib’s 68-landmarks detector [124] from the image frames and then faces are cropped and resized to 224×224 . Finally the face frames are normalized.
- (b) *Dataset:* Two public datasets have been used to evaluate the detection method. For fake images, we used the DeepfakeTIMIT (DF-TIMIT) [127], [128] dataset. The deepfake videos in this dataset have been generated using videos of 32 subjects from the VidTIMIT [192] dataset with FSGAN [170]. A total of 620 videos are generated using a lower quality (LQ) with 64×64 input/output sized model and a higher quality (HQ) 128×128 input/output sized model. We extracted fake image frames from those videos at a rate of $25fps$. For real images, the VidTIMIT [192] dataset has been used. It is a dataset with 43 individuals - 24 males and 19 females. The videos were shot in 3 different settings. There are 10 videos for each subject. The videos are stored as frames in the dataset. For our work, we used the same subject videos as DF-TIMIT [128]. Dataset details are mentioned in Table 4.6.
- (c) *Training Protocol:* To train the network we follow the below protocol with the aforementioned dataset.
- Transfer learning has also been used here to reduce the training time. We chose an ImageNet trained MobileNetV2 [193] network as the feature extractor by replacing the last 1000 node fully connected layer with a 2 node fully connected layer. To set up the classifier layer, it is initially

trained for 10 epochs keeping the weights of the feature extractor frozen. Then, the whole network has been trained for 15 more epochs from end-to-end. Finally the best model is chosen depending on the validation accuracy.

- Training data has been augmented to improve performance. Rotation, width shift, zoom, horizontal flip, and brightness have been changed to generate new images.
- The model has been trained on both datasets DF-TIMIT HQ and DF-TIMIT LQ separately. In both cases we use the same VidTIMIT dataset [192] for real images. During evaluation, the same dataset and cross dataset evaluation have been performed. For the same dataset evaluation, unseen data from the dataset is utilized for testing. For cross evaluation, one dataset is used for training and the other for testing. The results are presented in Sec. 5.7.
- For both datasets, $\sim 48,000$ images for training, $\sim 12,000$ for validation, and $\sim 4,000$ images for testing have been used as shown in Table 4.7.
- The network has been optimized with the Adam [125] optimizer of learning rate 0.0002 and other parameters to default values.

We implemented the detection method in Keras [51]. The model has been trained in a GeForce RTX 2060 laptop with 6GB shared and 16GB total memory.

TABLE 4.6. Dataset for Deepfake Detection of iFace 1.1.

Real		Fake	
Dataset Source	No. of Images	Dataset Source	No. of Images
VidTIMIT	34,004	DeepfakeTIMIT (HQ)	33,988
VidTIMIT	34,004	DeepfakeTIMIT (LQ)	34,025

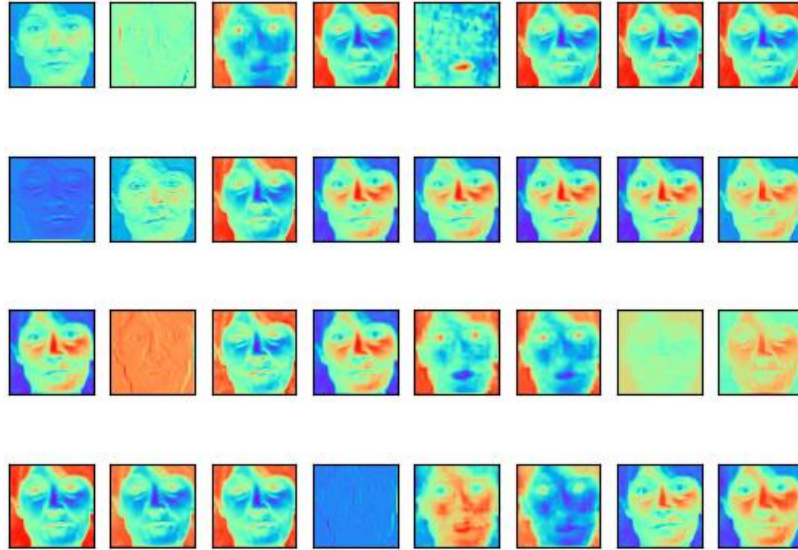
TABLE 4.7. Dataset Division for both DF-TIMIT HQ and LQ Dataset.

Data	Number of Images		
	Real	Fake	
		DF-TIMIT HQ	DF-TIMIT LQ
Train	23,873	23,939	23965
Validation	6135	6000	6010
Test	3996	4049	4050

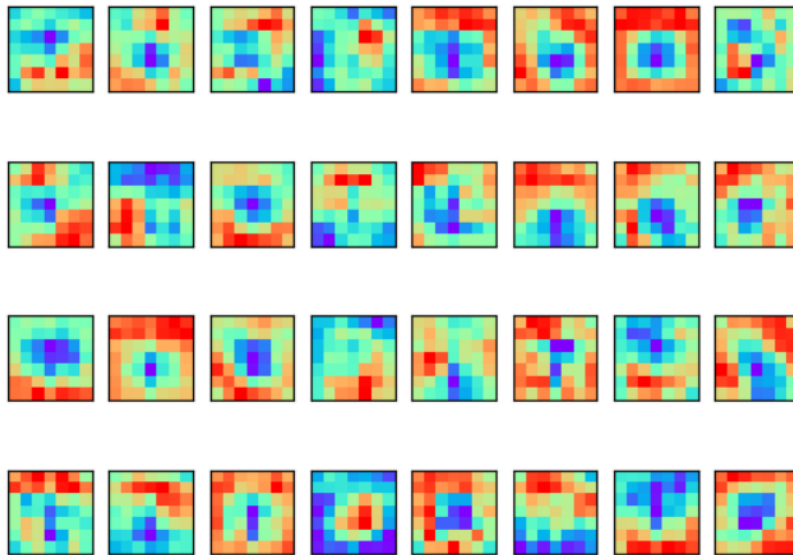
(d) *Results:*

Fig. 4.18 shows the features extracted by two different layers of MobileNetV2 [193]. Features extracted by the first convolutional layer are shown in Fig. 18(a). All the filters of this layer are activated at each part of input. As we go deeper in the CNN, the layers extract more complex features. Features extracted by 32 filters out of 1280 filters of the last convolutional layer are shown in Fig. 18(b). Fig. 19(a) shows the class activation heatmap of feature extractor MobileNetV2 [193] pretrained on ImageNet using GRAD-CAM [197] and Fig. 19(b) is that of MobileNetV2 [193] trained on DF-TIMIT HQ dataset. Fig. 4.19 shows MobileNetV2 [193], trained on DF-TIMIT HQ, correctly classifies real and fake images whereas Imagenet [62] trained MobileNetV2 [193] fails to classify. Table. 4.8 shows accuracy and inference times of different evaluation scenarios for the Deepfake Detection module. Perfect accuracy is achieved when the model is trained and evaluated on the DF-TIMIT (LQ) and VidTIMIT datasets. But in a real scenario, we counter with high-quality fraud. When the testing data is high quality and close to reality, we get a more realistic accuracy of 94.83%, which means that our system can find 94.83% of face-swapped deepfake images. The accuracy is really poor when trained on low quality images and tested on high quality images, which is expected. Inference time is also similar in the first three cases and high at the last case, as expected.

When the model is trained on low quality images, it considers high quality fake images as real. To evaluate classification performance of the model, the confusion matrix has been generated for a scenario when training and testing are performed on DF-TIMIT HQ dataset [128] as in Fig. 4.20.

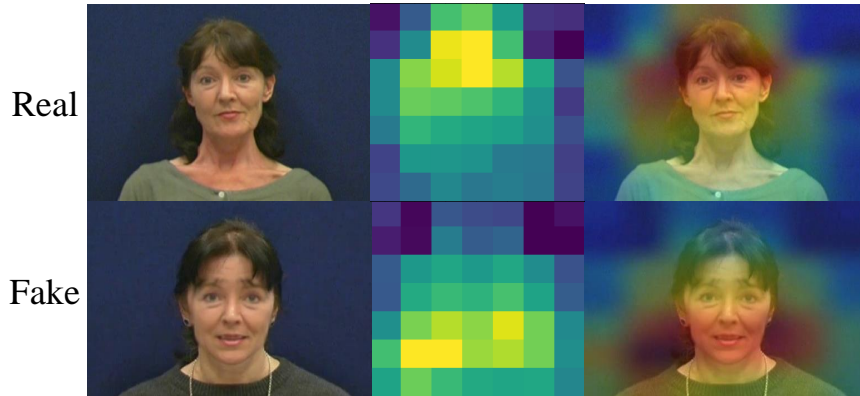


(a) Output of 32 filters of the first convolutional layer.

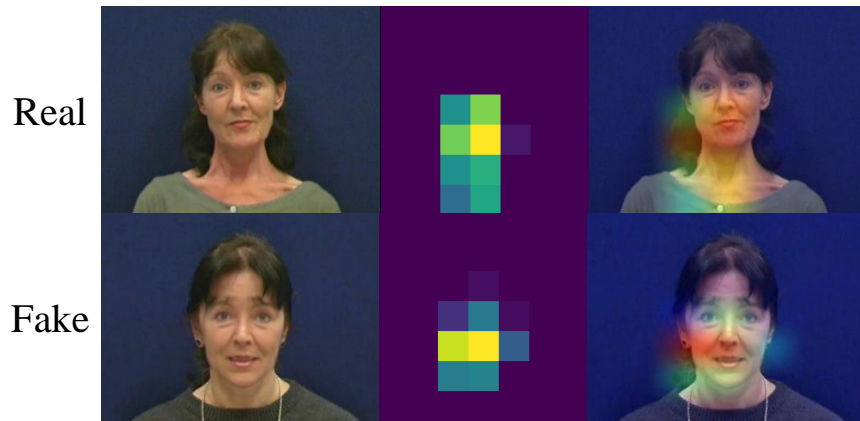


(b) Output of 32 filters out of 1280 filters of the last convolutional layer.

FIGURE 4.18. Feature visualization of MobileNetV2 for sample layers.



(a) Pre-trained on ImageNet (predicted wrong).



(b) Trained on DF-TIMIT HQ dataset (predicted correct).

FIGURE 4.19. Class activation map visualization using GRAD-CAM for MobileNetV2.

Precision, *Recall*, *Accuracy*, and *F1-score* have been calculated in Table 4.9 from the confusion matrix. Table 4.10 compares our proposed method with other existing solutions. We achieve much higher accuracy using MobileNetV2 [193] as feature extractor and training the network with full face images.

(2) *Presentation Attack Detection (PAD) Module*: To detect presentation attacks, the same pipeline shown in Fig. 17(b) is followed. Here we use EfficientNet B0 [213] as the feature extractor as it shows better results. A *GlobalAveragePooling* layer, followed by a *dense* layer of 2 nodes and a *Softmax* activation function, has been

used as the classifier. As presentation attacks have been approached as a binary classification problem, *Binary Cross Entropy* loss has been used. Transfer learning is also used here to shorten training time and improve accuracy.

TABLE 4.8. Accuracy and Inference Time for Different Evaluation Scenarios.

Training Dataset	Testing Dataset	Accuracy (%)	Inference Time (mS)
DF-TIMIT(HQ)	DF-TIMIT(HQ)	94.83	3.67
DF-TIMIT(LQ)	DF-TIMIT(LQ)	100.00	3.76
DF-TIMIT(HQ)	DF-TIMIT(LQ)	96.91	3.81
DF-TIMIT(LQ)	DF-TIMIT(HQ)	57.38	4.45

* For real images \rightarrow VidTIMIT dataset.

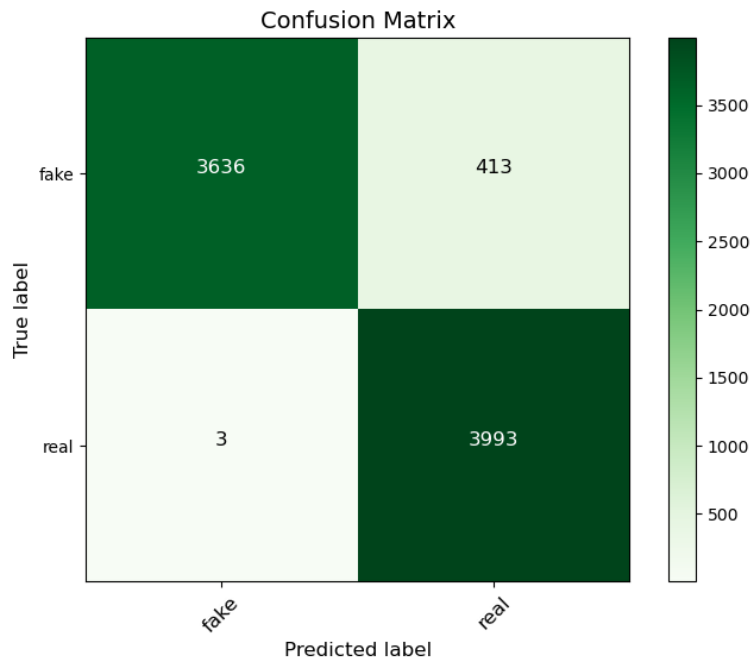


FIGURE 4.20. Confusion matrix - trained and tested on DF-TIMIT HQ.

The Replay Attack dataset [49] has been used to train and evaluate the Presentation Attack Detection module. The dataset is an imbalanced one. For our work,

we partially used the spoof part of the dataset to make a balanced one. The dataset details used in our work have been stated in Table 4.11. The number of frames extracted from test videos is fewer than that from training videos, as the duration of the test videos is shorter than the training videos in the dataset. Frames have been extracted using ffmpeg [37].

TABLE 4.9. Classification report-trained and tested on DF-TIMIT HQ.

Test Images	Precision (%)	Recall (%)	F1-score (%)
3996 Real	100.0	90.0	95.0
4048 Fake	91.0	100.0	95.0
Macro Average	95.0	95.0	95.0
Weighted Average	95.0	95.0	95.0
Total 8044	Accuracy (%)	95.0	

Here, the initial number of epochs for the classifier training is kept to 5, and end-to-end model training to 10. No data augmentation has been done. The model has been evaluated with the test section of Replay Attack [49] dataset.

Our system can detect 93.0% of presentation attacks. The performance of the module is evaluated through *confusion matrix* in Fig. 4.21. *Precision*, *recall*, *F1-score*, and *accuracy* have been calculated in Table. 4.12.

4.4.2.4. Face Features Extraction (FFE) Module

One of the major modules of the system is the feature extraction module, as it extracts the facial features from an image. We wanted to select a simple but highly accurate method as the Face Features Extraction (FFE) module. Hence, we followed the process as in FaceNet [196] instead of other new state-of-the-art methods to extract facial features. In image classification, state-of-the-art accuracy has been obtained in Google’s EfficientNets.

TABLE 4.10. Performance Comparison of Deepfake Detection Module of iFace 1.1 with State-of-the-Art Solutions.

Study	Year	Performance(%)	
		DF-TIMIT LQ	DF-TIMIT HQ
Matern et al. [150]	2019	AUC= 77.00	AUC=77.30
Yang et al. [247]	2019	AUC= 55.10	AUC= 53.20
Afchar et al. [27]	2018	AUC= 87.80	AUC=68.40
Zhou et al. [254]	2018	AUC= 83.50	AUC=73.50
Nguyen et al. [169]	2019	AUC= 78.40	AUC=74.40
Proposed Method	2021	ACC = 100.00	ACC=94.83

*ACC → Accuracy ; *AUC → Area Under the Curve

TABLE 4.11. Dataset for Presentation Attack Detection Module.

Dataset	Type	Image Type	No. of Images	Remarks
Replay	Train	Real	899	60 original train videos
		Spoof	891	120 spoof videos
Attack	Test	Real	80	80 original test videos
		Spoof	191	200 spoof test videos

Model size and computational complexity are notably low. Hence, instead of using the original deep learning network of FaceNet, we use EfficientNet B0 as the feature extractor [86]. Fig. 4.22 shows the workflow diagram of the FFE module at training. We use EfficientNet B0 [213], pretrained on ImageNet [62] as the backbone feature extractor, and connect a *GlobalAveragePooling* layer followed by a *dense* layer of 128 nodes without any

activation function. *L2 normalization* is used to extract face embedding as in [196]. The module extracts facial features from the face image. These features are expressed in terms of a 128-dimensional feature vector.

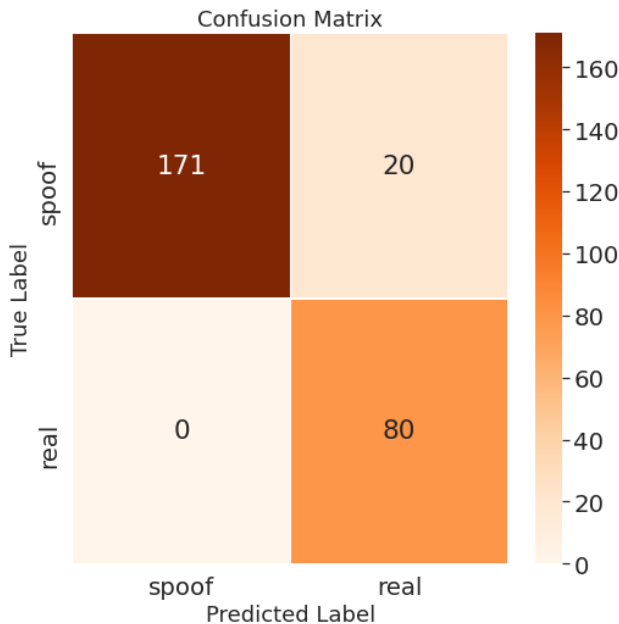


FIGURE 4.21. Confusion matrix for presentation attack module.

TABLE 4.12. Classification report of presentation attack module -trained and tested on Replay Attack dataset.

Test Images	Precision (%)	Recall (%)	F1-score (%)
191 Spoof	100.0	90.0	94.0
80 Real	80.0	100.0	89.0
Macro Average	90.0	95.0	92.0
Weighted Average	94.0	93.0	93.0
Total 271	Accuracy (%)	93.0	

The Triplet Loss function [196] has been used to train the feature extraction module. During training, the network learns to calculate the optimum *Euclidean* distance among images through embedding.

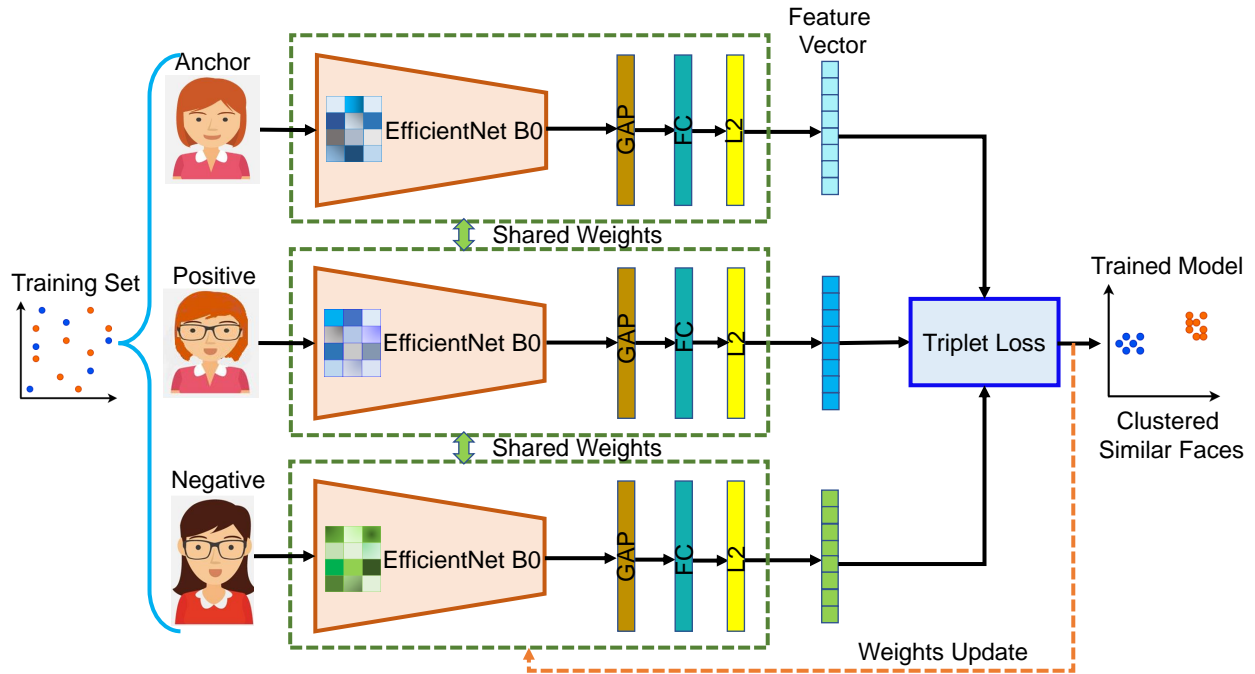


FIGURE 4.22. Training of face features extraction module.

Once the feature extractor has been trained, the trained model is used to get face embeddings from the test image.

To train the face features extraction module, we downloaded images of the main characters of the American sitcom ‘Friends’ using the Bing Search API and created a customized dataset. The dataset details are mentioned in Table. 4.13. We used images without any occlusion in the face.

This module is trained for 100 epochs. A set of triplets, i.e. anchor image, positive image, and negative image, are generated before training. To generate the triplets, the below procedure has been followed:

- (1) For our dataset, we have 6 classes (6 characters). 2 classes are chosen randomly.
- (2) 2 images are chosen from one class, and 1 image is selected from the other class. This process is also random. From 2 images, one is chosen randomly as the anchor image and the other as the positive image, whereas the single image from the other class is chosen as the negative image.
- (3) For each anchor image, we chose 10 positive and 10 negative random images. So for

our 247 images, 24,700 combinations of triplets were generated.

Fig. 4.23 shows the Principal Component Analysis (PCA) plot of the embeddings of our training dataset for the FFE module. Here, 2 principal components are used to reduce the dimensionality of the embedding vectors. Fig. 23(a) shows the embeddings before training the module, and Fig. 23(b) shows the same embeddings, clustered after the training.

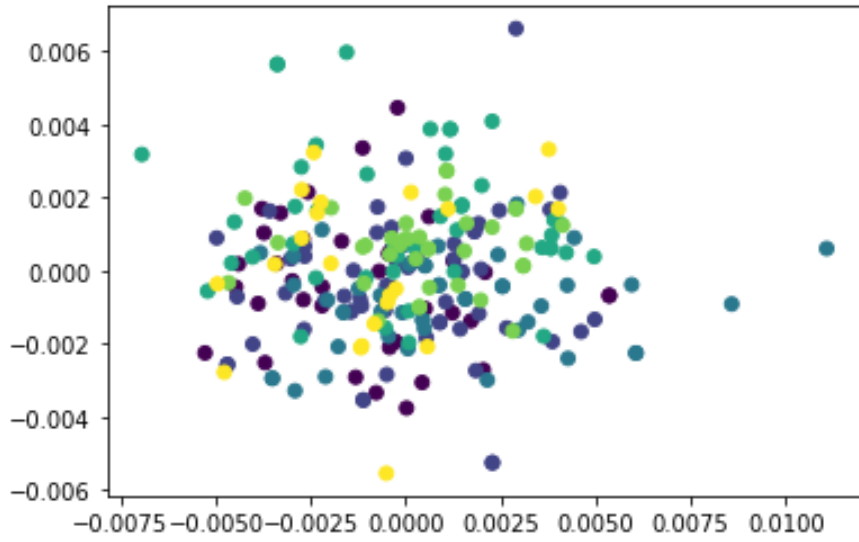
TABLE 4.13. Customized Dataset for Face Features Extraction Module and Classification Module.

Dataset	Character Names	No. of Images
Main 6 Characters of American sitcom “Friends”	Chandler	45
	Joey	50
	Monica	47
	Phoebe	43
	Rachel	36
	Ross	30

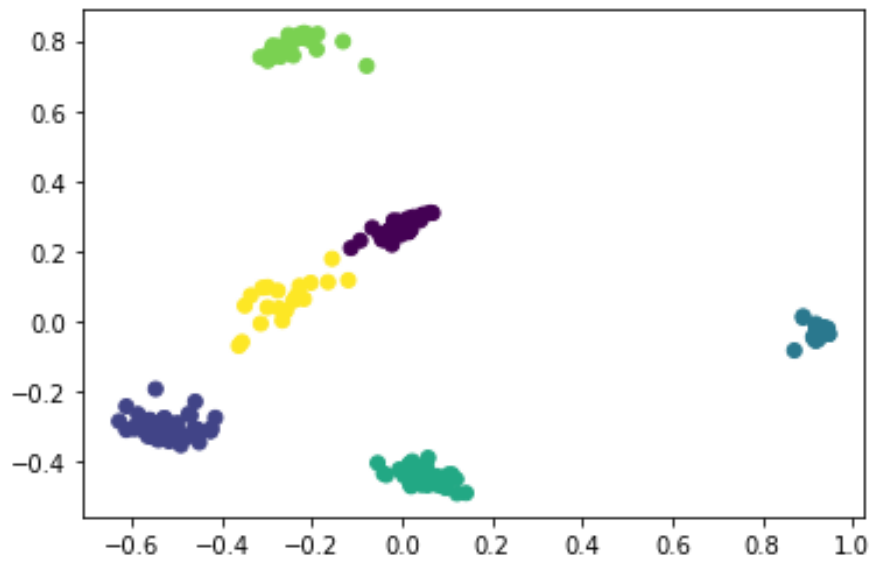
4.4.2.5. Classifier Module

Once the FFE module extracts the features, a classifier is used to predict the identity of the image. In our work, a k-Nearest Neighbor (kNN) classifier with 3 neighbors, an auto-tree algorithm, and 30 leaves is used to make the prediction. The distance metric of the classifier is chosen as *Euclidean*. We train the classifier with the face embedding extracted from the trained FFE module. It authenticates or denies a face by measuring the distance to k nearest neighbors and finally decides by taking a majority voting. Fig. 4.24 shows the classifier’s training. The classifier predicts the username corresponding to the face embedding.

Embeddings from the FFE module are used as the training data for classification. The dataset details are mentioned in Table. 4.13.



(a) Before training.



(b) After training.

FIGURE 4.23. Embedding plots of six main characters of American sitcom “Friends”.

Embeddings from the trained FFE module are used as the input of the classifier. So, the number of training images of the classifier is equal to the number of original images i.e. 247.

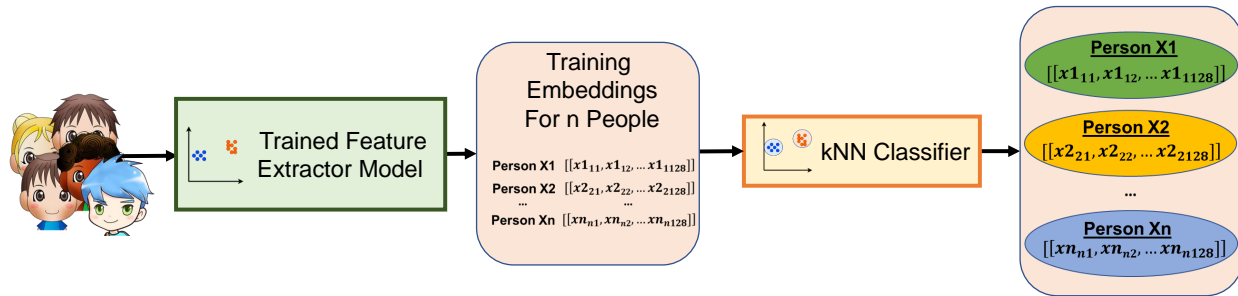


FIGURE 4.24. Classifier training.

4.4.3. Performance Evaluation of the Proposed Digital ID System

In this section, the performance of iFace 1.1 is described, along with a comparison to existing works.

The user is authenticated in an intuitive way by comparing the input username with the predicted username from the face embeddings extracted during authentication on the smart device. This authentication is conducted on the edge.

Fig. 4.25 shows the authentication process. If the person is already enrolled in the system and the predicted username matches with the input username, the classifier authenticates as in Fig. 4.25(a). The person gets access to the facility. But if the person is not enrolled in the system or if the predicted username does not match with the input username, the system does not give access to the facility to the person as in Fig. 4.25(b).

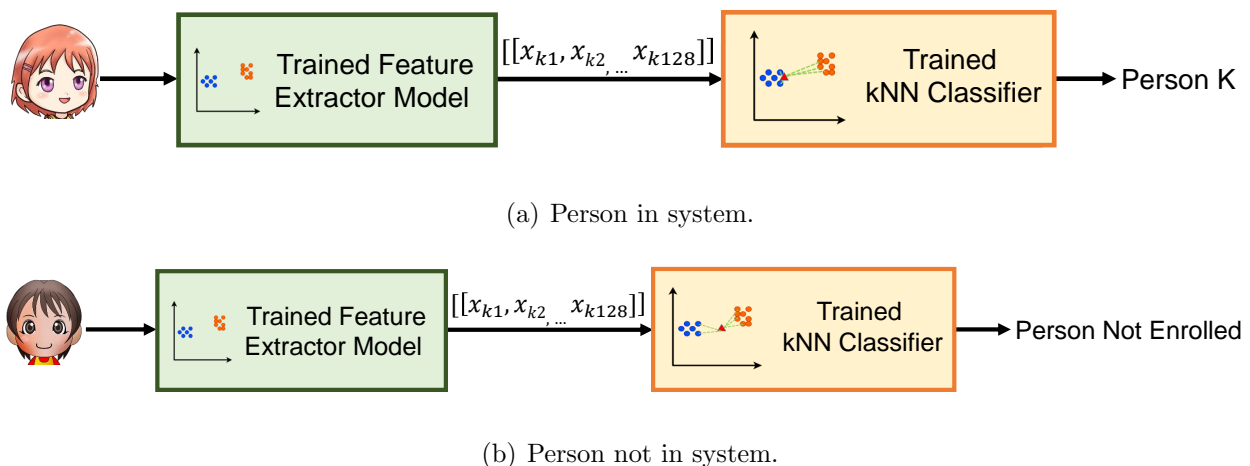


FIGURE 4.25. Authentication process of iFace 1.1.

During authentication, data collection, processing, feature extraction, and prediction are all done on the edge platform. As user verification is conducted each time the user accesses the facilities, the odds of impersonating a person are very low. This provision makes the system robust. Encryption during data transfer and storage further secures the system.

When a face needs to be authenticated in the system, the embedding is extracted from the image through the trained FFE module. Two such scenarios are plotted in PCA plots, as in Fig. 4.26 along with training data embeddings. If the person is already enrolled in the system, the PCA plot is as in Fig. 4.26(a) and if the user is not yet enrolled in the system, Fig. 4.26(b) depicts that scenario. 171 spoofs have been correctly detected among 191 spoof images.

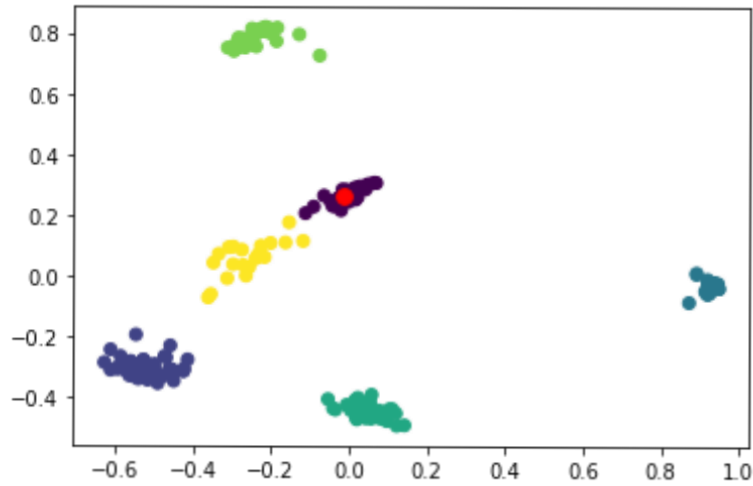
The system has been evaluated with 50 unseen frontal images of 6 enrolled people and 500 frontal images of the CelebA [148] dataset. The first set is also used as the authorized people to measure the false rejection ratio (FRR) and the second set as the intruders to measure the false acceptance ratio (FAR).

The performance of the facial recognition system of the proposed digital ID is shown in Table. 4.14 and Fig. 4.27. FRR is calculated with 50 images of enrolled people. One of the 50 images was falsely rejected, generating an FRR of 2%. We evaluate the system with 500 images from the CelebA [148] dataset. 15 images were falsely accepted. FAR is calculated to 3%.

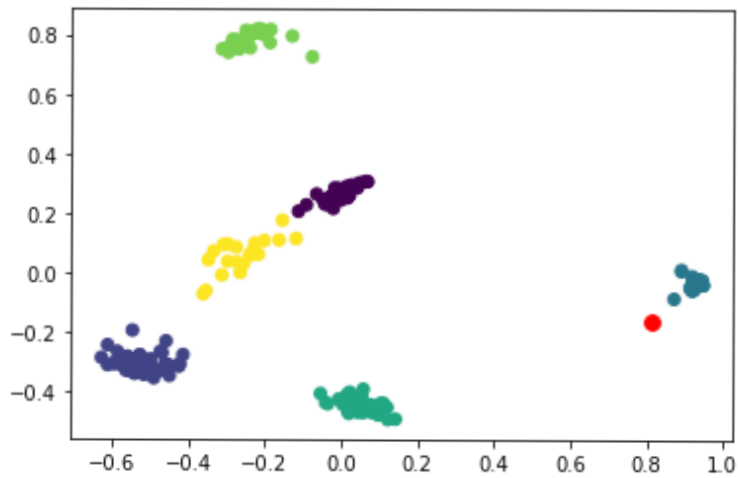
The evaluated images are not all frontal faces. The foreheads or cheeks of some faces are occluded with hair. Those images are the cause of the false acceptance ratio. There is a variation in age for certain ‘Friends’ characters in our training and testing dataset. This was intentionally chosen to show the effect of the aging of a person in his/her digital ID.

Various metrics like *precision*, *recall*, *F1-score*, *accuracy*, *FRR*, and *FAR* have been calculated in Table. 4.15 to evaluate the digital ID system. Table. 4.16 compares the proposed iFace 1.1 with the existing face authentication systems. The methods that are applicable to mobile devices or on the edge platform are stated there. It is clear from Table.

4.16 that different metrics have been used to evaluate the performance of these systems. Most of the systems perform face authentication without providing any security measures. They are mainly standalone facial authentication or verification systems. On the other hand, our paper presents an end-to-end facial authentication-based digital ID system that can detect both presentation attacks and deepfake attacks.



(a) For enrolled person.



(b) For not enrolled person.

FIGURE 4.26. Embedding plot of sample person in the clustered training dataset.

TABLE 4.14. Performance of the Proposed Digital ID System iFace 1.1.

Dataset	Type	No. of Test Images	No. of Authentication		
			Correct	False Acceptance	False Rejection
Own Data	Enrolled User	50	49	0	1
CelebA	Not Enrolled User	500	485	15	0

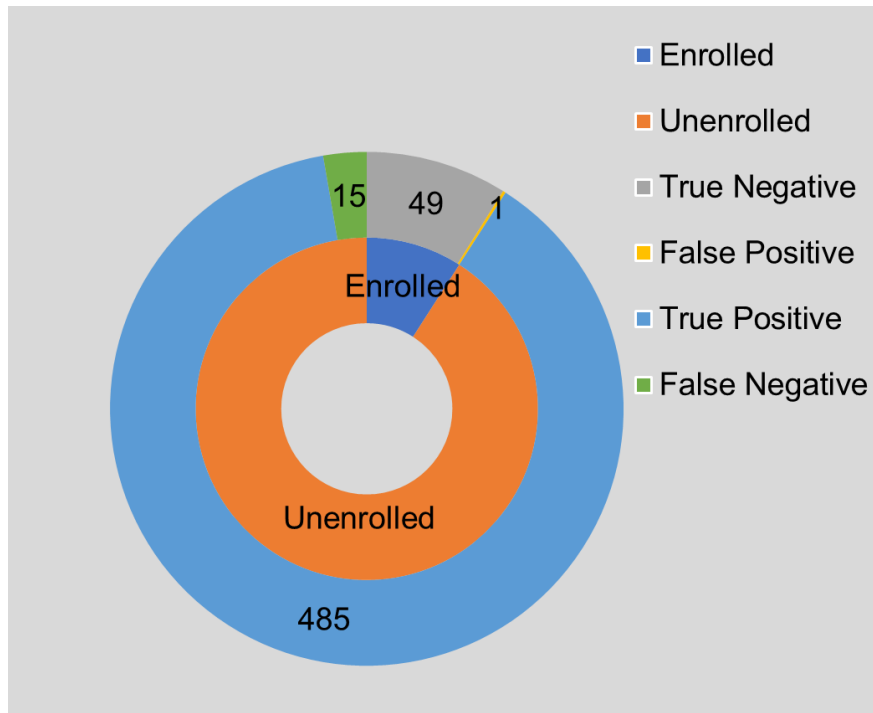


FIGURE 4.27. Digital ID performance.

4.4.4. Conclusions and Future Work

In this chapter, a proof-of-concept of an end-to-end facial authentication-based digital ID system for smart cities has been proposed. Several things have been accomplished here:

- Our system is capable of detecting the various intruder attacks mentioned in Sections

1 and 2. The system is resilient to deepmorph deepfake attacks, mentioned in Section 2. It shows high accuracy even with high quality face swapped GAN generated images.

- It can detect presentation attacks, mentioned in Section 1, with an accuracy of 93%.
- The false acceptance ratio and false rejection ratio of the system are fairly low.
- The face authentication system has an accuracy of 97%. As facial authentication has been done at the edge, the risk of security compromise is reduced.
- No photos are stored anywhere in the system. Face features are stored in the cloud in terms of a numerical value. Hence, it is not possible to reverse engineer the photos from these numbers, which makes the process secure.
- Biometric data is stored without any identifiable information about the user, eliminating the data privacy regulation issues.

TABLE 4.15. Performance Metrics of Facial Authentication System.

Test Images	Precision (%)	Recall (%)	F1-score
500 Impostors	100.0	70.0	98.0
50 Users	77.0	98.0	86.0
Macro Average	88.0	97.0	92.0
Weighted Average	98.0	97.0	97.0
Total 550	Accuracy (%)	97.0	
	FRR	2.0	
	FAR	3.0	

Currently, the deepfake detection module detects the face-swapped images. However, a comprehensive deepfake detection module for other sources of deepfakes will be added. The PAD module upgrade is another area where more experiments will be performed. Our proposed system has demonstrated promise. However, more study and experimentation are needed before smart city deployment.

TABLE 4.16. Performance Comparison of the Proposed Facial Authentication System with Existing Papers.

Papers	Performance Metrics	Face Authentication	Attacks Resiliency
Tao et al. [214]	EER 2%	Yes	No
Tao et al. [215]	EER 1.2%	Yes	No
Hadid et al. [87]	Acc. varies with image size	Yes	No
Sarkar et al. [194]	Acc. 88%	No	No
Masud et al. [149]	Acc. > 95%	Yes	No
Mitra et al. [157]	FRR 2.77%	Yes	Yes
Current Paper	Acc. 97%, FRR 2%	Yes	Yes

EER → Equal Error Rate.

Acc. → Accuracy.

FRR → False Rejection Ratio.

As for future work, a more efficient digital ID system can be achieved by addressing the following areas:

- A re-enrollment of facial features is needed to accommodate the age-related changes if the person significantly ages.
- Data deletion processes need to be included in the system after someone's death.
- A data update option needs to be there if there is any facial change due to any accident or cosmetic surgery. The provision for deletion of data and re-enrollment of the user with a new user id should be incorporated in the case of identity theft.
- The modules and the facial authentication system need to be upgraded to state-of-the-art systems.
- A large dataset of human faces is required with images generated by various GANs to obtain a generalized deepfake detection module. People of different demographics,

races, colors, genders, and ages should be included in the systems. People with glasses, piercings, head coverings, hearing aids, and braces should also be included in the training dataset. The hardware should support this lengthy and resource-intensive training.

- The existing PAD module detects spoofs from the face photo. The PAD module must be tested with different head poses [34], lighting conditions [228], inside, outside, day, and night settings.
- It can be improved by using challenge response techniques with random instructions for motion, e.g., head and eye movement, opening and closing the mouth, or reading aloud any random sentences.
- Voice verification can also be added to enhance one more level of verification.
- The system should be capable of authenticating people with facial occlusions such as sunglasses, masks, and any facial piercings.
- It will be exciting to see if the identical twin scenario can be addressed.

To the authors' best knowledge, iFace 1.1 is the first viable end-to-end proof-of-concept that addresses the main challenges of smart-city digital ID. We hope to see more research in this direction and finally an implemented iFace system in a smart city.

4.5. Discussions

In this chapter, we proposed a deepfake resilient digital id system for smart cities. Our proposed iFace 1.1 system is a digital id system which is based on facial biometric features authentication. It is robust against various presentation attacks too. The success rate of detecting deepfake images is high. It can detect highly sophisticated and easy to generate state-of-the-art deepfake images.

CHAPTER 5

DATA SCARCITY: A NOVEL FRAMEWORK FOR AUTOMATIC CROP DAMAGE ESTIMATION

In this chapter, ML-based especially deep neural network-based solution for a data scarcity problem in agriculture domain has been proposed [159].

5.1. Introduction

The climate of the Earth has altered over time. The majority of these shifts were caused by minor variations in Earth's orbit, which vary the amount of solar energy absorbed by our planet [1]. However, since the mid-20th century, the current climate era, which began 11,700 years ago and marked the beginning of modern civilization, has largely changed due to human actions. It is worsening at an unprecedented rate [15].

A steady climate is essential for human civilization, especially agriculture [74]. Human actions have warmed the earth's oceans, biosphere, and atmosphere. Excessive heat, drought, fires, floods and intense storms are among of the observed impacts [241]. Some of the implications of climate change are seen in Fig. 5.1.

Extreme weather events and calamities caused by climate change harm agriculture. Climate change impacts agricultural growth and yield, putting pressure on the food supply chain [73]. The damage can occur from early planting to harvesting. Crop damage costs a country's agriculture industry billions of dollars. Hence, a protection umbrella is required. Crop insurance protects farmers' finances.

To avoid such losses, farmers contact their insurer. The crop insurance company dispatches a loss adjuster. The adjuster collects photos, reviews meteorological data, and talks to neighbors to determine damage [17]. Data from a damaged sample of an HDZ are extrapolated to the entire land. The loss adjuster selects the homogenous damage zone manually. However, identifying a homogeneous damage zone without knowing the level of damage is difficult on broad lands [202]. Large lands have primarily heterogeneous damage.



FIGURE 5.1. Effects of climate change - drought, wildfire, ice melting at the poles, flood, and storms

Extrapolation in that scenario is flawed. However, because insurance money alleviates some stressors for farmers, the insurance claim procedure must be simple, seamless, and precise.

5.2. Addressed Research Problem

Despite recent advances in AI, its use in agriculture is still in its infancy. To forecast effectively, deep neural networks require vast training datasets. In practice, acquiring huge datasets for deep learning networks is difficult or impossible. Such failures result in deep neural network errors. The lack of relevant datasets is one of the many causes for the sluggish digital transformation of agriculture, even while datasets are available in concentrated study areas including plant disease [18], soil health [21], groundwater nitrate contamination [12], and disaster analysis [10]. Due to data scarcity, agriculture has failed to fully benefit from AI. Crop damage due to natural causes is one of these topics -

- Comprehensive public dataset is unavailable. Hence, the traditional deep learning methods do not produce any high accuracy result.
- At various stages of crop growth, crop or plant can be damaged by natural events. Hence, vast amount of data for different growth level of crops/plants are necessary to present an AI based model.
- Data scarcity limits using vast data-based AI solutions for estimating crop damage.

USA is the largest corn producer of the world and according to a new NASA study corn yield will be reduced 24% due to climate change by 2030 [73]. Fig. 5.2 shows the projection of corn fields in 2070 where red color represents highly decreased corn production. Corn production in parts of both Americas, West Africa, Central Europe, and India and China in Asia will be severely affected [73]. As in near future, corn will be one of the most impacted crops, we use corn as a case study for evaluating our method. However the method should be portable for it to be applied to any crop. In this paper, the problem of data scarcity in estimating crop damage caused by natural disaster has been addressed in context of corn production.

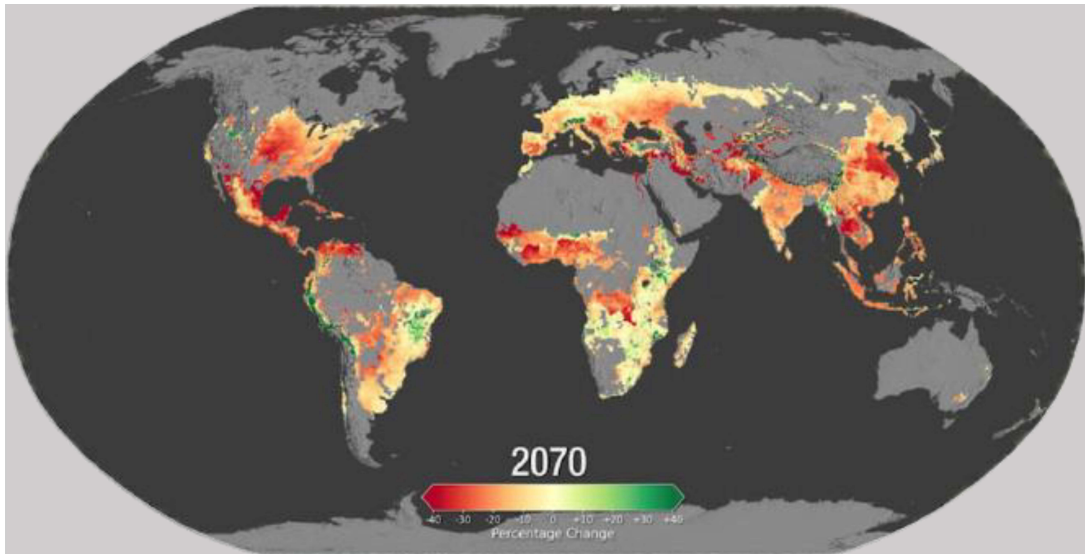


FIGURE 5.2. Corn yield projection in 2070 [73]. In the color gradient scale, red means the mostly affected whereas green means not affected.

5.3. Proposed Solution: eCrop

In this paper, we propose a **proof-of-concept** of a grid-based method **eCrop** for estimating the crop damage across a large crop field. It is a part of the proposed **agro cyber physical system (A-CPS)**.

The problem of crop damage detection is difficult to address as there is no public dataset. However *eCrop* addresses the above data scarcity challenge and provides a solution to estimate the crop damage caused by natural events for a large crop field. A Siamese network-based method has been proposed to detect crop damage and evaluated through a case study.

When the loss adjuster from the insurance company comes to the damaged field to inspect crop status, our *eCrop* method assists in detecting the crop damage accurately and automatically. Adaptation of our proposed method in processing the insurance claim will make the payment process easier, automated, and error free.

5.4. eCrop: A Novel Method to Evaluate the Extent of Crop Damage

5.4.1. Proposed Agro Cyber Physical System (A-CPS)

A cyber physical system is the integration of physical systems and computational resources. When the Internet-of-Things (IoT) is implemented in a physical system, it forms a cyber physical system. As CPS comprises of various heterogeneous objects, connection and communication among these devices play a key role. CPS increases the efficiency, scalability, and usability of any system. Application of CPS in any industry not only reduces the cost [4], it also makes the system more adaptable and seamless. It advances the industrial growth towards automation.

A-CPS is the cyber physical system in agriculture. It integrates the Internet-of-Agricultural-Things and computing elements. Our proposed A-CPS is shown in Fig. 5.3. A-CPS is the foundation stone of smart agriculture. It increases the efficiency of agricultural systems, predicts the yield precisely, estimates the damage automatically, and presents solutions for sustainable agriculture.

Our proposed A-CPS in Fig. 5.3 presents different granularity of data at different levels. Different stakeholders e.g., farmers, horticulturists, environmental scientists, and insurance providers access different levels of data. It makes the process more secure and robust.

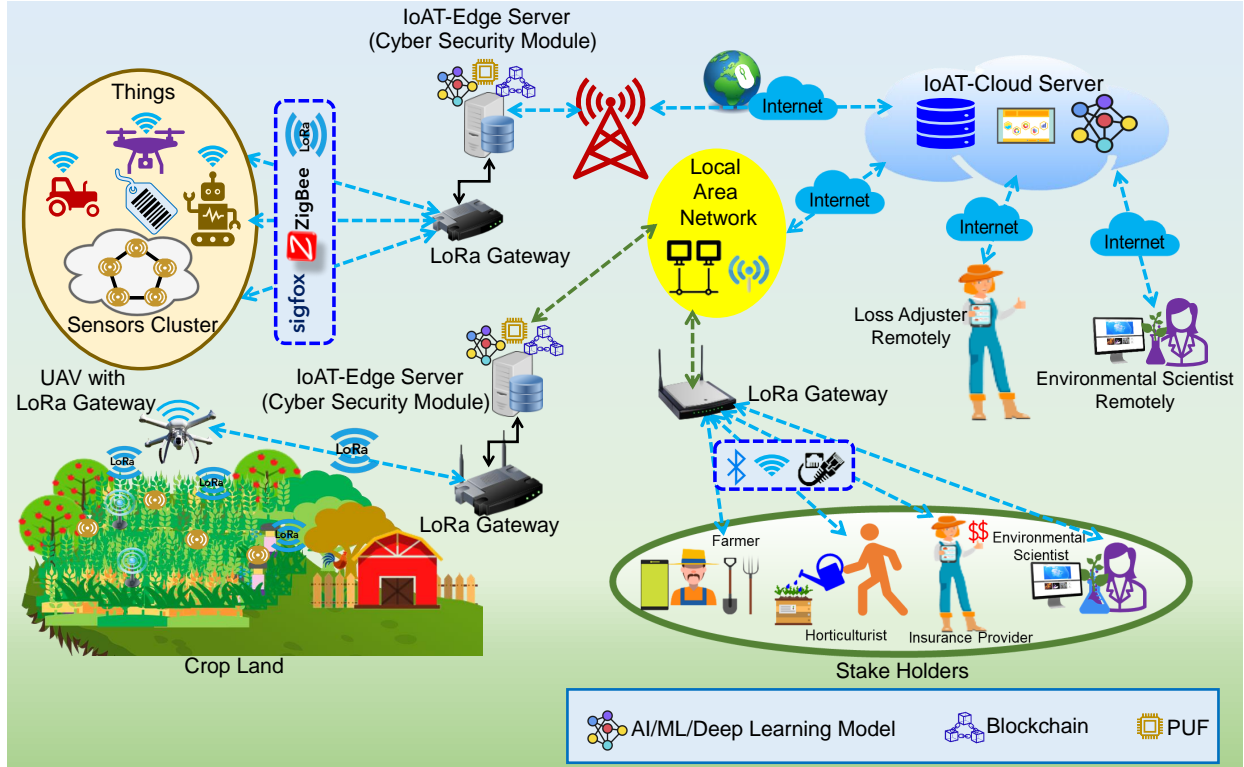


FIGURE 5.3. Proposed agro cyber physical system (A-CPS). eCrop is a part of the proposed A-CPS.

The proposed A-CPS has mainly 3 layers : agro layer, edge layer, and cloud layer. They are connected through the connectivity layer [153] Internet of Agriculture Things (IoAT) from agro layer. Sensors, robots, UAVs [129], and RFID tags are the IoAT devices. The agro layer is connected to the edge layer via a connectivity layer e.g., near range ZigBee or long range SigFox and LoRaWAN. Edge layer processes the data and computes time sensitive operations. This layer comprises of IoAT edge servers. These edge servers are equipped with hardware accelerators. Machine learning models run in these devices. For authentication of these devices Physical Unclonable Functions (PUF) are used. Blockchains

are used for data integrity purposes. Agro layer IoAT devices are connected to the IoAT edge servers through LoRa Gateways. Finally, all data are stored at the IoAT cloud servers for future use.

Crop damage estimation is part of this A-CPS. Here, farmers and insurance providers are the related stakeholders, UAV is the IoAT device, IoAT edge server computes the damage, and finally IoAT cloud server saves the data.

5.4.2. Proof-of-Concept of eCrop

In this section, we propose the proof-of-concept of the *eCrop* method for evaluating the extent of damage. It is a grid-based method. Fig. 5.4 shows the grid and Fig. 5.5 shows the overall *eCrop* pipeline.

1 1	2 1	3 1	4 1
5 1	6 1	7 1	8 0
9 1	10 1	11 1	12 0
13 1	... 1	... 0	N 0

FIGURE 5.4. eCrop grid for evaluating the extent of crop damage.

In the event of crop damage due to natural causes, crop damage estimation is performed at different times and crop growth stages for different damage type. For hail and wind damage, damage estimation is done using eCrop just after the disaster whereas for heat, drought, frost, and fungal diseases, estimation is done near to the harvest time.

A UAV is sent to take photos through out the large field following the proposed *eCrop* method. The data collected by the UAV is then sent to the IoAT edge server which processes

the data and estimates the damage.

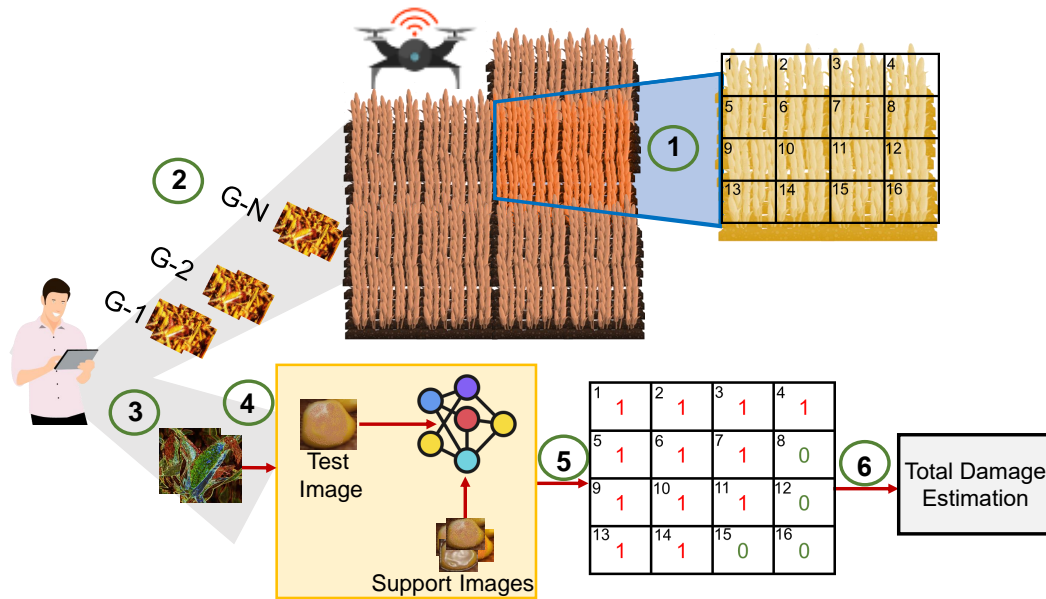


FIGURE 5.5. eCrop system overview.

5.4.3. eCrop Grid

- (1) The large crop field is divided into a grid system as in step-1 of Fig. 5.5.
- (2) For each grid, the adjuster takes several photos of the crop through a UAV, as shown in step-2 of Fig. 5.5.
- (3) Any existing Machine Learning (ML) method can detect the damaged area as in step-3 of Fig. 5.5.
- (4) Damage type is detected using our proposed damage detection method discussed in Sec. 5.5 and shown in Fig. 5.6. The process is repeated for all the images of the grid as in step-4 of Fig. 5.5. (In case of crop kernel level approach, 50% of the damaged area is sufficient to check the damage type. But for smaller crops, head or panicle level approach is advisable.)
- (5) Final damage type for the grid is calculated from average similarity score of that grid.
- (6) If any damage type is identified, the grid is updated with 1 as in step-5 of Fig. 5.5.

(7) The whole process is repeated for all grids and an overall estimate is calculated for the damage as in step-6 of Fig. 5.5.

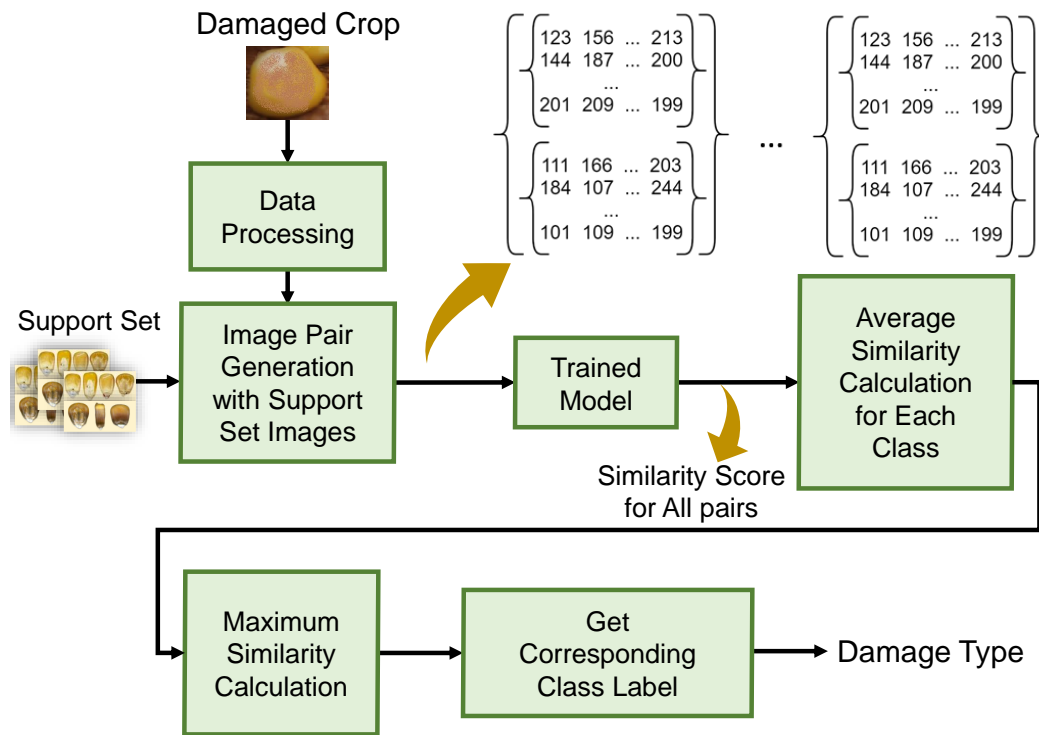


FIGURE 5.6. Automatic detection of crop damage for each grid.

The method is scalable to any crop. For more general method, a crop selection module with various stages of crop growth such as planting, growth, and harvesting can be added in the eCrop system. To avoid identifying deformed grains as damaged grains, an elimination module can be added too. The module will detect the deformed grains and not allow the system to detect the damage on those grains.

5.4.4. eCrop Grid Generation

The first step of estimating crop damage is *eCrop* grid generation. Fig. 5.7 describes the proposed *eCrop* grid generation method for detecting the crop damage. First, the map of the crop field is uploaded in the eCrop system. The (*latitude*, *longitude*) of the four corners of the land are retrieved. The distance between the corners are calculated using the *Haversine formula*.

If the (*latitude, longitude*) of two points P and Q are denoted as (ϕ_1, λ_1) and (ϕ_2, λ_2) respectively, the *great-circle* distance between them is calculated using *Haversine formula* as in Eq. 5.1 assuming the points are on a sphere:

$$(5.1) \quad d_{P,Q} = 2R \arcsin \left(\sin^2 \left(\frac{\phi_1 - \phi_2}{2} \right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2 \left(\frac{\lambda_1 - \lambda_2}{2} \right) \right)^{1/2}.$$

In the above expression, $d_{P,Q}$ is expressed in km, the radius of the earth R is 6371 km, and the (*latitude, longitude*) angles are in radians.

Once the distances are calculated, Algorithm 4 is followed. Then, $N = n \times m$ number of grids, each of size (100×100) sq.meter, are drawn and photos are captured. Finally, crop damage is detected for the entire grid.

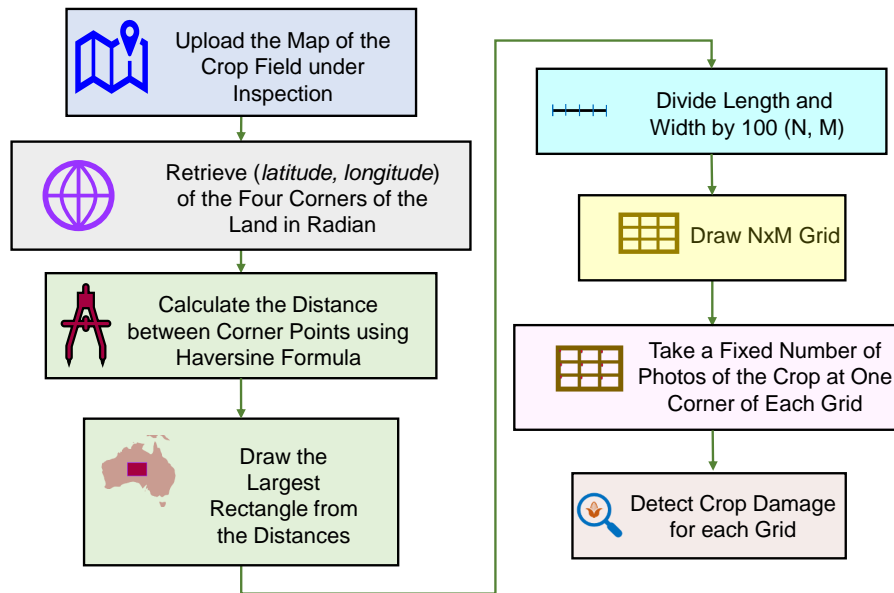


FIGURE 5.7. Grid generation of eCrop system for detecting crop damage.

5.4.5. Extent of Damage Calculation

For each (100×100) sq.meter grid, several images of the crop are taken through UAV. Once the damaged area in each image is detected, the average damage type is identified. Grid score value is updated with 1 if any damage is detected, otherwise it is 0. The damage type is also noted. This process is repeated for each grid. As the images are taken through

Algorithm 4 Procedure to Detect Crop Damage for the Entire Grid.

```
1: function ECROP(length, width)
2:   Declare variables n, m, count, damagetype, similarityscore, and griddamagetype, grid
   and initialize them to 0
3:   Declare a variable imagecount and set a value for it
4:   Draw a rectangle with sides length, width
5:   Set l  $\leftarrow$  round( $\frac{length}{100}$ )
6:   Set w  $\leftarrow$  round( $\frac{width}{100}$ )
7:   Draw  $l \times w$  grids on that rectangle
8:   Set row to 0
9:   for  $m \in w$  do
10:     Set column to 0
11:     for  $n \in l$  do
12:       for  $count \in imagecount$  do
13:         Take photo of the crop at position (row, column)
14:         Detect damagetype and note similarityscore
15:         Save damagetype, similarityscore, and count
16:       end for
17:       Update griddamagetype from average similarityscore
18:       Save griddamagetype and value of n and m in grid  $column \leftarrow column + 100$ 
19:     end for
20:     row  $\leftarrow row + 100$ 
21:   end for
22: end function
```

UAV, the process of imaging is easy. If the number of damaged grids out of total N grids is u , then the extent of damage e_{damage} is given by Eq. 5.2:

$$(5.2) \quad e_{damage} = \left(\frac{u}{N}\right).$$

The claim value M is calculated using Eq. 5.3:

$$(5.3) \quad \begin{aligned} M &= c \times e_{damage} \times N \times 10,000 \\ &= c \times u \times 10,000. \end{aligned}$$

In the above expression, c is the insured claim value in dollars per sq. meter of the field.

A hypothetical case is assumed to present the effectiveness of our proposed grid method. The average size of the crop lands vary across the globe. In USA, the average cropland varies from coast to coast. Most of the fields located west of the Mississippi are $(\frac{1}{2} \times \frac{1}{2})$ sq.mile and they were combined with time. Some are of size (1×1) sq.mile and separated by roads. According to USDA report [226], the average cropland size was 444 acres or 1.79 sq. km as per 2020 data whereas in Argentina the average crop land size is 500 acres [202]. We take a value 1.69 km square size with sides of (1300×1300) sq.meter, close to USA data as an example. It is divided into $13 \times 13 = 169$ unit grids, each of size (100×100) sq.meter. If 150 out of 169 grids have damaged crops, then e_{damage} will be 0.89 and the total money claimed will be $\$0.89X$ instead of $\$X$ where $\$X$ is the original claimed insured money for that damaged crop land. This grid method will help the insurance company to calculate the claimed money accurately and automatically.

5.5. Meta Learning-based Detection of Crop Damage for Each Grid

Here, we present the architecture and the learning protocol of our proposed crop damage detection method. We apply this method to detect the crop damage for each grid of a crop land.

5.5.1. Architecture

We propose a Convolutional Siamese Network-based framework as damage detection network. A Siamese network [43] is composed of two co-joined twin networks. The twin networks are called sister networks. They are identical and they share weights and network parameters. They are joined with an energy function at the top but accept different inputs

or an image pair. The energy function can be a Euclidean distance or cosine similarity. They calculate the similarity score between the two images. In our work, a shallow convolutional neural network (CNN) has been used to extract the features from the images. Fig. 5.8 shows the proposed CNN structure used in the sister networks of damage detection system.

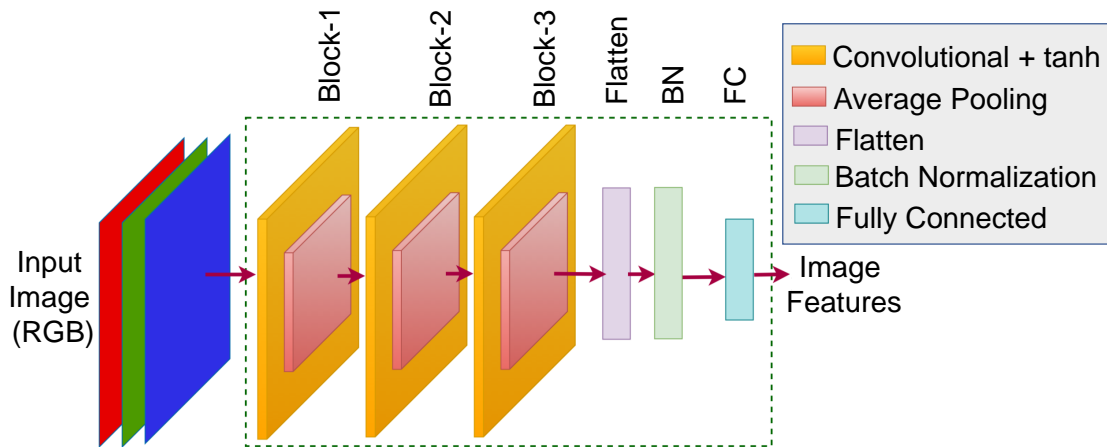


FIGURE 5.8. CNN structure used in sister networks of damage detection system.

It has 3 *convolutional* blocks. Each block consists of a *Convolutional* layer with tanh activation and *Average Pooling* layer. The pooling layer reduces the spatial dimensions. The number of filters varies in each block but the kernel sizes are kept the same. Finally a *Flatten* layer, followed by a *Batch Normalization* layer with default parameters, and followed by a *Fully Connected* layer form the sub-network. The parameters of the layers are presented in Table. 5.1. Two of these structures have been used as the sister networks of the Siamese network as in Fig. 5.9 and they share weights between each other. The total number of trainable parameters are 4,514. The network accepts inputs as a pair. Each sister network accepts an RGB input image of size 28×28 . The output of each sub-network is a 16-dimensional feature vector.

5.5.2. Data Pair Generation

As a Siamese network accepts a pair of images as inputs, data pair generation plays a significant role for the training of this network. If the two images are from the same class,

TABLE 5.1. Sister Network Architecture Details.

Layers	Parameters	Output Shape
Conv2D	f=16, k=3, s=1, p=1	(28,28,16)
Averagepooling2D	k=2, s=2	(14,14,16)
Conv2D	f=8, k=3, s=1, p=1	(14,14,8)
Averagepooling2D	k=2, s=2	(7,7,8)
Conv2D	f=8, k=3, s=1, p=1	(7,7,8)
Averagepooling2D	k=2, s=2	(4,4,8)
Flatten	-	(128,)
BatchNormalization	-	(128,)
Dense	u=16	(16,)

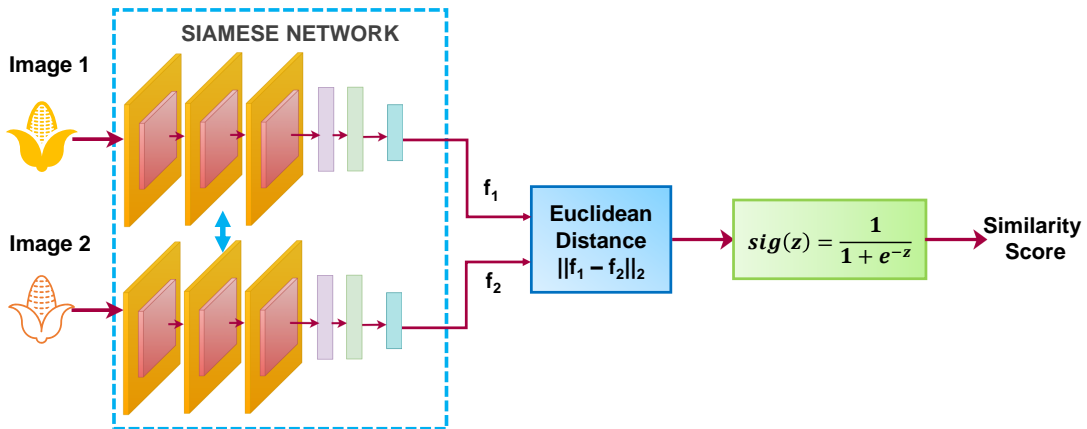


FIGURE 5.9. eCrop network consisting of CNN structure in Fig. 5.8 as sister networks.

they form a positive or similar pair and if the images are from different classes, a negative or dissimilar pair is formed. Each pair is labeled. The label value of the pair is denoted by Y . It is expressed in Eq. 5.4:

$$(5.4) \quad Y = \begin{cases} 1, & \text{for similar images} \\ 0, & \text{for dissimilar images} \end{cases}$$

Before making the data pair, RGB images are resized to 28×28 and normalized. This step has been done to reduce the computational time.

5.5.3. Energy Function and Similarity Score

To know the similarity score between the image pair, the *Euclidean distance* has been used as the energy function. It measures the distance between two images in feature space. If the feature space is w dimensional and p_i and q_i are two points in that space, then the *Euclidean distance* D_w between those two points is given by Eq.5.5:

$$(5.5) \quad D_w = \sqrt{\sum_{i=1}^w (p_i - q_i)^2}$$

For our case w is 16. Eq.5.6 shows the *Euclidean distance* in our case:

$$(5.6) \quad D_{16} = \sqrt{\sum_{i=1}^{16} (p_i - q_i)^2}.$$

A smaller distance means the similarity score is high and the images are similar. Similarity is measured through a *Fully Connected* layer with 1 node and sigmoid activation function as in Fig. 5.9.

5.5.4. Method and Training Protocol

Though AI has significantly advanced in recent years, application of AI in agriculture is in budding stage. Deep neural networks need large training datasets to predict accurately. But in reality, gathering a large dataset for training a deep learning network is not so easy or the required data is not always available. In such cases deep neural networks fail to function accurately. In agriculture, the unavailability of necessary datasets is one of the many reasons for the slow digital transformation of agriculture although datasets are available in

concentrated areas of research e.g., plant disease [18], soil health [21], groundwater nitrate contamination [12], and disaster analysis [10]. Due to limited availability of data, agriculture is yet to harvest the full benefits of AI.

Crop damage by natural causes is one of such issues - where large public dataset is not available. To overcome this issue, we applied the concept of *meta learning*. In *meta learning*, machine learning model learns the new task seeing only few data instead of being trained with a large dataset.

We apply a *few shot* learning approach. For classification problem, *few shot* learning is stated as N -way- K -shot classification where N denotes the number of classes with K images in each class. The network learns from a small dataset, called *support set*, with N classes and K samples in each class and is evaluated using a *query set*. *Support set* is usually a part of large dataset. An *episodic training* process is usually followed where in each episode different but small *support sets and query sets* from a large dataset are shown to the model. By this method, the model learns how to classify a new unseen class from the test query set when the test support set is available.

In our case, we have a small dataset and damages are known and specific. Known and specific crop damages remove the necessity of training the network in a true episodic manner of few shot learning as no unknown class is needed to be detected. We train the network with few images and detect the damage correctly.

5.5.5. Loss

Contrastive loss [88] has been used to train our Convolutional Siamese network. If there is an image pair of two input images x_1 and x_2 with pair label Y and D_w is the *Euclidean distance* between those two images in feature space, the Siamese network can find image similarity by measuring D_w . D_w is optimized by minimizing the *contrastive loss* L_{con} expressed in Eq. 5.7. The margin value m is set to 1 for *contrastive loss* L_{con} :

$$(5.7) \quad L_{con}(x_1, x_2) = (1 - Y) \frac{1}{2} (D_w)^2 + (Y) \frac{1}{2} \{ \max(0, m - D_w) \}^2.$$

5.5.6. Proposed Algorithm

To detect the type of damage, we propose Algo. 5. Whenever the UAV sends images to the IoAT edge server, the damage is detected using Algo. 5.

5.6. Evaluation of the Proposed Crop Damage Detection Method for Each Grid

In this section, we present the experimental validation of our crop damage detection method. To evaluate our method, we did a case study on corn. We used corn kernels to detect the damage type. With proper availability of ear level data, the procedure is the same.

5.6.1. Dataset

In practice, images taken by the UAV camera will be used for training and inferring. But to evaluate our system, we train our neural network with available images.

We chose four types of damages such as heat, frost, disease (fungal), and insect infestation for evaluating our method. Other types of natural events, e.g. flood, hail, and storms can also be included here. Crop diseases are crop specific. So, to evaluate our method, we chose only one crop, corn, and heat, frost, cob rot, and insect damages have been considered. Corn kernels images from [3] and [5] have been used for training and evaluating our method.

The resolution of the images in our dataset is low, as those images are collected from the pdf copy of the reports [5, 3]. We used the corn kernel images as is without doing much image enhancement. The dataset details for our work are mentioned in Table. 5.2. Fig. 5.10 shows some of the sample training images [5]

5.6.2. Implementation

The implementation has been done in Python. Keras API [51] with TensorFlow [26] backend has been used. The training of the Siamese network for crop damage detection has been performed in Jupyter Notebook of a Dell G5 Windows 10 laptop with NVIDIA® GeForce® RTX 2060, 6GB GDDR6 video card and 16 GB memory. Batch size has been varied from 4 to 32. Number of epochs has also been changed from 50 to 100. The model

Algorithm 5 How to Detect Crop Damage Caused by Natural Events?

- 1: **Input:** Image *testimage*
- 2: **Output:** Label *cropldamagetype*
- 3: Declare the *supportimages* path and Model \tilde{M} path
- 4: Declare and initialize variables *f*, *i*, *c*, *label similarity* to 0
- 5: Declare *average similarity* as a list
- 6: Read *testimage*
- 7: Resize *testimage* to 28×28
- 8: Normalize *testimage*
- 9: Load Support Images *supportimages*
- 10: Call MAKEPAIR()
- 11: Load Model \tilde{M}
- 12: Predict *similarity* for all *imagepair*
- 13: Get the similarity score between pairs with 30th percentile and 80th percentile to avoid any outlier
- 14: Get *average similarity* for each class from the range
- 15: Find maximum *average similarity* from the *average similarity*
- 16: Get corresponding *foldername* value for maximum *similarity*
- 17: Get the *label* name from *foldername* value
- 18: Set *cropldamagetype* to correct *label*
- 19: **function** MAKEPAIR
- 20: Declare *imagepair* and *f* as list and initialize to 0
- 21: Declare *foldername* as list and initialize to 0
- 22: **for** *images* \in *supportimages* **do**
- 23: **for** *i* \in *images* **do**
- 24: Read image *i*
- 25: Normalize image *i*
- 26: Make *imagepair* with *testimage* and image *i*

```

27:         Update foldername with f
28:     end for
29: end for
30: return imagepair, foldername
31: end function

```

TABLE 5.2. Dataset Details.

Cause of Damage	No. of Total Images
Disease (Cob Rot)	10
Frost	10
Heat	10
Insect	10

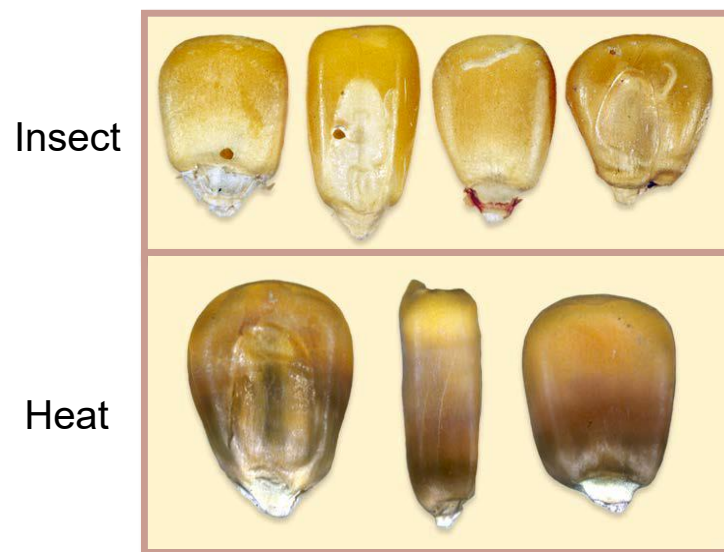


FIGURE 5.10. Sample damaged grain images used for training [5]. In practice images taken by UAV will be used.

has been evaluated in 4GB Raspberry Pi 4 using TensorFlow Lite Converter. Algorithm. 5 is used to detect the damage.

5.6.3. Validation

In this section, we present the validation details of the proposed crop damage detection method. The pipeline in Fig. 5.11 has been followed. Collected data are saved in different folders with the damage name. The folder names provide the class labels.

As we didn't have much data to train, the same data used for training has been used as support images during testing too. The RGB images are resized to 28×28 and normalized. Similar and dissimilar pairs are then formed with normalized images.

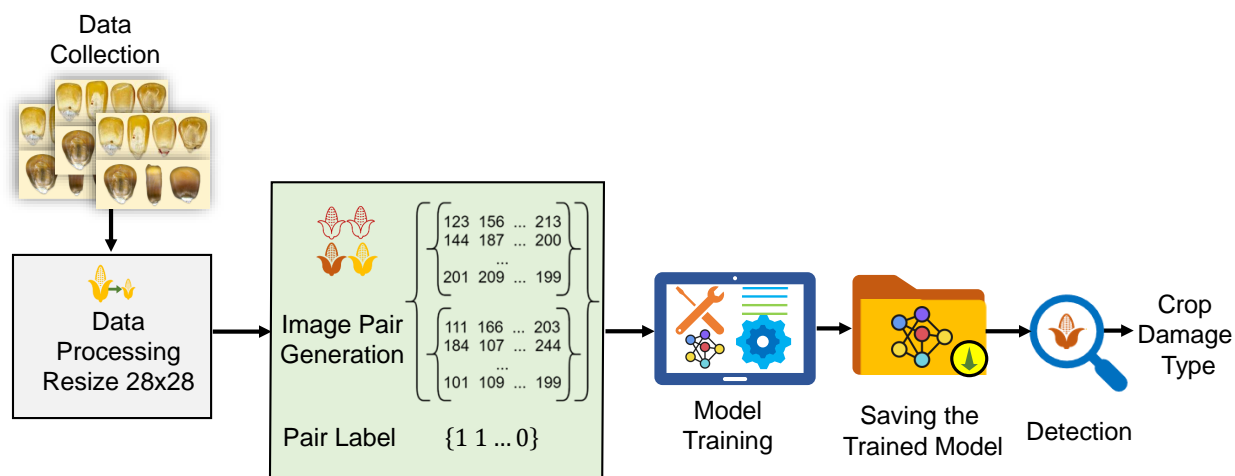


FIGURE 5.11. Crop damage detection pipeline.

Once the pairs are formed, they are used in training and evaluating the network mentioned in Fig. 5.9. The Euclidean distance between the pairs has been optimized using *contrastive loss*. The margin for the *contrastive loss* is kept at the default value 1. An *Adam* optimizer with learning rate 0.001 has been used. We trained the model with different values of N (number of ways/number of class types) and with different values of K (number of shots/number of images per class).

5.7. Results and Comparative Study

We evaluated our system by varying N and K . Fig. 5.12 and Fig. 5.13 show the accuracy vs number of shots (K) plots for two different values of N whereas Fig. 5.14 and

Fig. 5.15 show the training time vs number of shots (K) plots for those two values of N . N is set to 3 and 4.

N is 4 in Fig. 5.12. The four classes we chose here are *Cob Rot*, *Frost*, *Heat* and *Insect*. Each class has 10 images. We varied the number of training samples for each class in different evaluation scenario by changing K . We changed the value of K to 4, 5, and 6 keeping the number of validation images to 2 for all cases. Hence the number of test images varied from 4 to 2. N is 3 in Fig. 5.13. Here the classes are *Cob Rot*, *Frost*, and *Heat*. Table. 5.3 shows the number of images used for training, validation, and testing for both the cases.

During evaluation we compare the similarity of the test image with the support images, here the training images. Therefore, the number of testing combination is much higher than the number of test images. For example, let's assume the case when the number of classes N is 4 (all classes are considered) and $K = 4$ i.e., number of images per class is 4. Hence for a specific training scenario:

- number of training images = 4.
- number of validation images = 2.
- number of test images = $(10 - (4 + 2)) = 4$.

But evaluating a model with only 4 images does not perform the accurate validation. Here, the working of Siamese network plays a significant role.

- We have total 4 classes.
- Each class has 4 images.

Therefore, the total number of test combinations T is expressed as in Eq. 5.8:

$$\begin{aligned}
 T &= N \times K \times \text{testimage} \\
 (5.8) \quad &= 4 \times 4 \times 4 \\
 &= 64.
 \end{aligned}$$

In the above expression, N is the number of classes, K is the number of images in each class, and *testimage* is the number of test images. Hence the ratio of test combinations per class

and total training images is expressed in Eq. 5.9:

$$\begin{aligned}
 \frac{\text{Number of test combinations}}{\text{Number of total training images}} &= \binom{T}{NK} \\
 (5.9) \qquad \qquad \qquad &= \binom{64}{16} \\
 &= \binom{4}{1}.
 \end{aligned}$$

If we consider the ratio of all the test images with total training images, the ratio is even much higher. It will be equal to Eq. 5.10

$$\begin{aligned}
 \frac{\text{Number of test combinations}}{\text{Number of total training images}} &= \binom{T \times 4}{NK} \\
 (5.10) \qquad \qquad \qquad &= \binom{256}{16} \\
 &= \binom{16}{1}.
 \end{aligned}$$

However, when a single test image is given, the number of test combinations becomes as in Eq. 5.11:

$$\begin{aligned}
 \text{Number of test combinations} &= N \times K \times 1 \\
 (5.11) \qquad \qquad \qquad &= NK.
 \end{aligned}$$

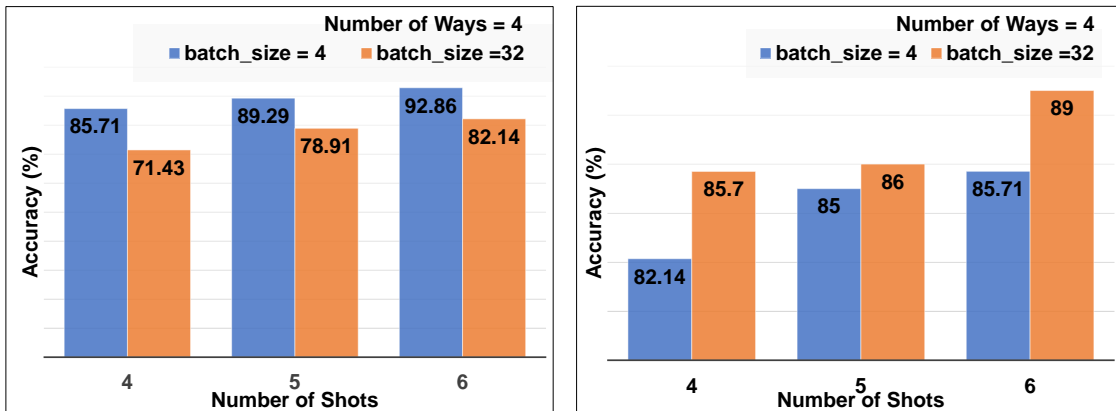
We noted accuracy with different epochs. The training was complete for epoch value of 100. After that, the model started to overfit as the *contrastive loss* became erratic. The best model was obtained for both N with learning rate 0.001. Fig. 5.16 shows the first 3 features of the support images obtained from the trained model for $N = 4$.

Fig. 5.12 and Fig. 5.13 show that the accuracy increases with N even if the number of support class types increases. In our case, as the support set is same as training set, the number of training images also increases with increase of N . As a result, the model learns better.

With an increase of K , the accuracy increases as the model has more data to learn. As in Table 5.4, we obtained the highest accuracy of 92.86% for $N = 4$ and $K = 6$. It

TABLE 5.3. No. of Images for Training, Validation, and Testing.

N	K	No. of Train Images	No. of Validation Images	No. of Test Images	No. of Testing Combinations
4	4	4	2	4	64
	5	5	2	3	60
	6	6	2	2	48
3	4	4	2	4	48
	5	5	2	3	45
	6	6	2	2	36



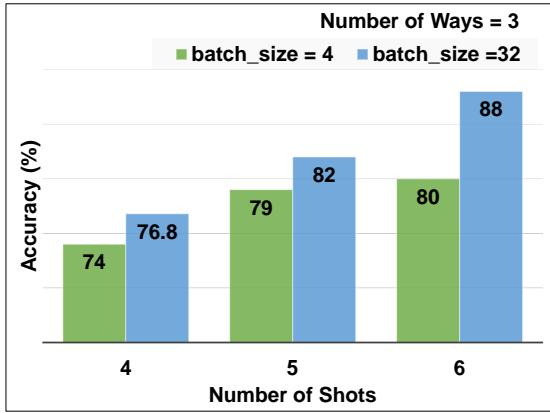
(a) For epoch=100.

(b) For epoch=50.

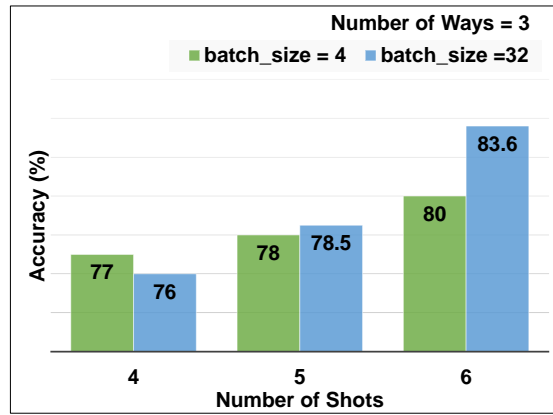
FIGURE 5.12. Accuracy vs number of shots (K) for number of ways (N)=4.

was achieved when the model was trained with epoch=100 and batch size=4. For $N = 3$ the highest accuracy 88% is obtained when the model is trained with epoch=100 but batch size=32 and $K = 6$.

We varied the batch size to see the effect of batch size on training time. The training time is low for higher batch size as expected. Fig. 5.14 and Fig. 5.15 confirm that. Fig. 5.14 and Fig. 5.15 show that the training time increases with N for both higher and lower batch sizes. It is expected as for higher number of N , the number of training images increases.

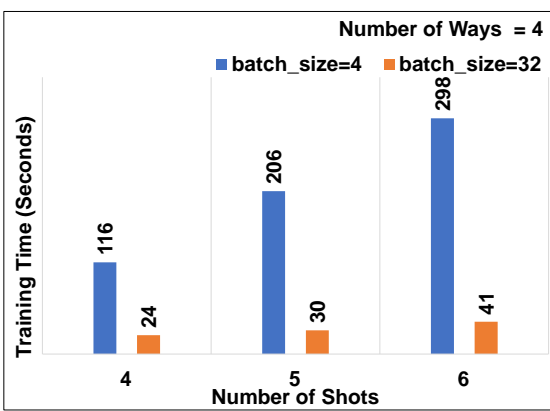


(a) For epoch=100.

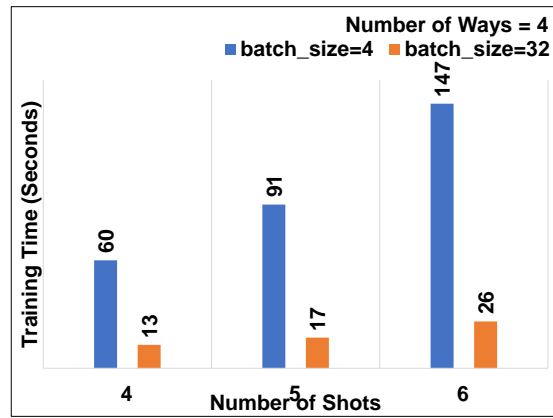


(b) For epoch=50.

FIGURE 5.13. Accuracy vs number of shots (K) for number of ways (N)=3.



(a) For epoch=100.

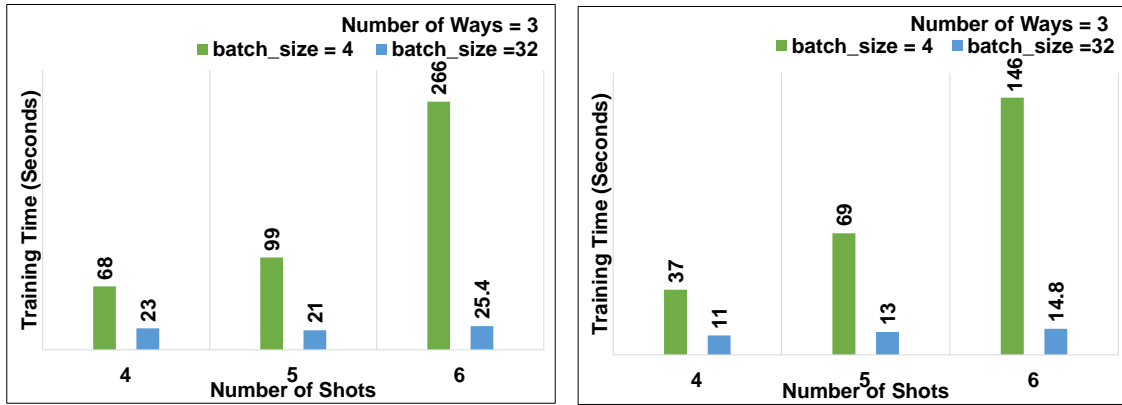


(b) For epoch=50.

FIGURE 5.14. Training time vs number of shots (K) for number of ways (N)=4.

TABLE 5.4. Accuracy for Different N and K.

N	K	Accuracy (%)
4	6	92.86
3	6	88



(a) For epoch=100.

(b) For epoch=50.

FIGURE 5.15. Training time vs number of shots (K) for number of ways (N)=3.

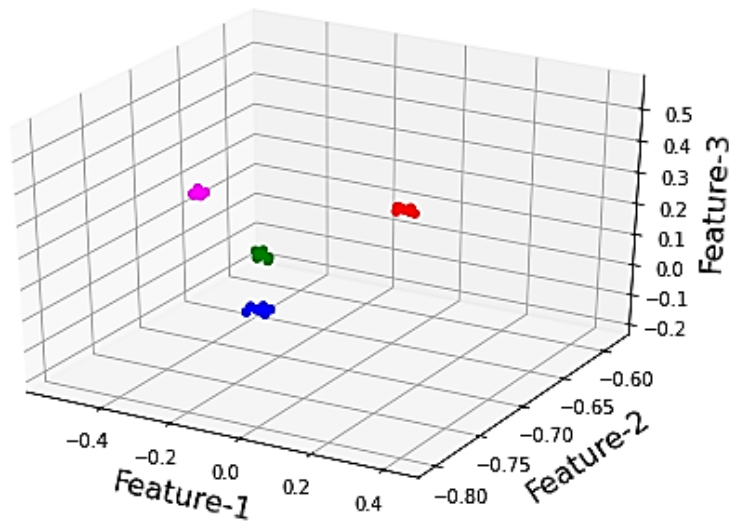


FIGURE 5.16. First 3 features of support images from the best trained model for N=4.

Table. 5.5 shows a comparative study between the existing works and our proposed method. Most of the papers present crop damage estimation for a single type of damage. They are not suitable for at-location real-time estimates. But our proposed damage detection method addresses those issues. It also includes more damage types compared to the existing works. Our *eCrop* system can estimate the crop damage over a large crop field.

TABLE 5.5. A Quantitative Comparison of the Current Paper with Existing Works.

Works	Year	Damage Type	Accuracy (%)	Real Time
Sosa et al. [202]	2021	Hail	87.01	No
Sawant et al. [195]	2019	Cyclone, earthquakes, hail storms, and flood	87.23, 92.22	No
Yang et al. [245]	2019	Cold	82.19	No
Pallagani et al. [174]	2019	Crop disease	99.24	Yes
Di et al. [67]	2018	Natural Disaster	95.00	No
Hsuan et al. [105]	2018	Heavy rain and typhoon	NA	No
Kwak et al. [133]	2015	Flood	80.00	No
eCrop	2022	Any damage type: heat, frost, diseases, and insect	92.86	Yes

5.8. Conclusions and Future Work

The agricultural industry may struggle to feed the world population which will reach to 9.7 billion by 2050. The condition will be aggravated due to the vulnerability of agriculture to climate change [2]. More and more researches are required to address various crop damage related issues for future sustainable agriculture.

In this paper, we performed the following tasks-

- We have addressed an agricultural problem where data scarcity poses a barrier to present solutions.
- We presented an agro cyber physical system and brought all agricultural research problems under one CPS. We addressed one such agricultural research problems: estimation of crop damage caused by natural disasters.
- We proposed a proof-of-concept of a grid based system. It can estimate the crop damage of a large crop field precisely and automatically.

- We built a crop damage detection method from very few training data.
- We also evaluated the damage detection method used at the grid level. Our method can detect the crop damage with a higher success rate and is scalable to any crop.

Higher accuracy can be obtained with higher quality and more number of training images which cover all types of damages for a specific crop. Integration of blockchain and PUF based methods will be explored in future for robust cyber-attack resilient smart agriculture Cyber Physical System (A-CPS) [114, 72].

Implementation of the end-to-end *eCrop* system will be an important and relevant future work for estimating crop damage due to natural disaster. As eCrop system reduces the work of the loss adjuster by making the process automated with high accuracy, we believe our work has the potential to be applied to assess the crop damage in practice.

CHAPTER 6

DATA INSUFFICIENCY: A NOVEL FRAMEWORK FOR PLANT DISEASE DETECTION AND LEAF DAMAGE ESTIMATION

In this chapter, how does data insufficiency, brings down the accuracy of a deep neural network-based solution has been studied through an agricultural domain problem [158]. Here with adequate data and additional data augmentation, the accuracy of a typical deep learning model has been boosted up.

6.1. Introduction

One of the most important sectors of the modern economy is the agricultural sector. It's complex, and it's vulnerable to factors like climate change, population growth, scarcity of natural resources, and plant diseases. Numerous agricultural problems are being solved as a direct result of recent developments in information and communication technology (ICT), including game-changing hardware advances and a shift in the computing paradigm from cloud computing to more edge-oriented computing. *Agriculture 4.0* [153] is here, thanks to the incorporation of automation in the form of Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) technologies, and *Agriculture 5.0* is on the doorstep. Initiatives for A-CPS-based agriculture solutions are more important than ever. Some of the agricultural issues that can be addressed with A-CPS ideas are illustrated in Fig. 6.1.

Similarly to other forms of life, plant life can contract diseases. When a plant is infected with a disease, it can't grow to its full potential [19]. Seasons and plant types affect the frequency. Diseases can be triggered by either natural environments or biological organisms. In contrast to biotic diseases, which are caused by plant pathogens such fungi, bacteria, viruses, and algae, abiotic diseases are those that are caused by environmental factors including nutrient deficiency, extreme temperatures, flooding, and freezing. Fig. 6.2 presents the "*Disease Triangle*" [204] that sheds light on the occurrence of a biotic disease. When a disease develops, it is due to the presence of all three of these elements: a favorable environment, a vulnerable host, and a harmful pathogen. The area of the Venn diagram

where the disease occurs is shown in red in Fig. 6.2. However, this may change depending on things like the genetic variety of the pathogen, the local micro climate, and the immunity of the host plant during a particular time in the life cycle [19]. The pathogen must finish its life cycle in the host in order for the plant to become infected.

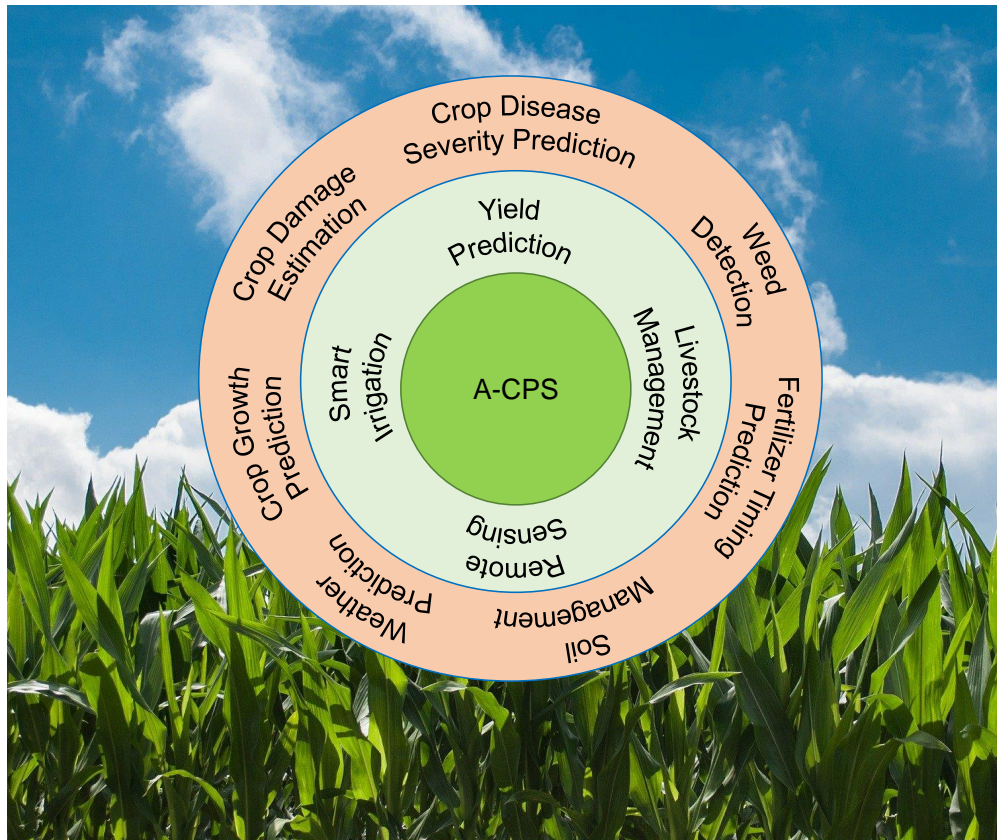


FIGURE 6.1. Agricultural problems solvable using agriculture cyber physical systems.

6.2. Addressed Research Problem

Plants can't grow when they're infected. The crop's quality suffers and its final yield is diminished because of plant diseases. These cause annual crop losses worth billions of dollars. The food distribution system is also severely affected [20]. That's why it's crucial for farmers to:

- Detect the disease early.
- Identify the disease.

- Know about the severity of the disease.
- Determine the extent of damage.

For a disease management plan to be effective, plant inspections must be performed on a regular basis. By 2030, early and accurate diagnosis of plant diseases and their prevention are expected to become the primary focuses of agricultural research, as predicted by [168]. In Chapter 5, we propose a solution for situations where a dearth of data prevents the use of AI for tasks like estimating agricultural damage after natural disasters. In this paper, we focus on the issue of insufficient training data from the opposite perspective. We have sufficient amounts of data for this second agricultural issue—the identification and quantification of damage caused by plant diseases. We studied the optimal amount of data required for acceptable accuracy. The previous three out of four points are covered in this section.

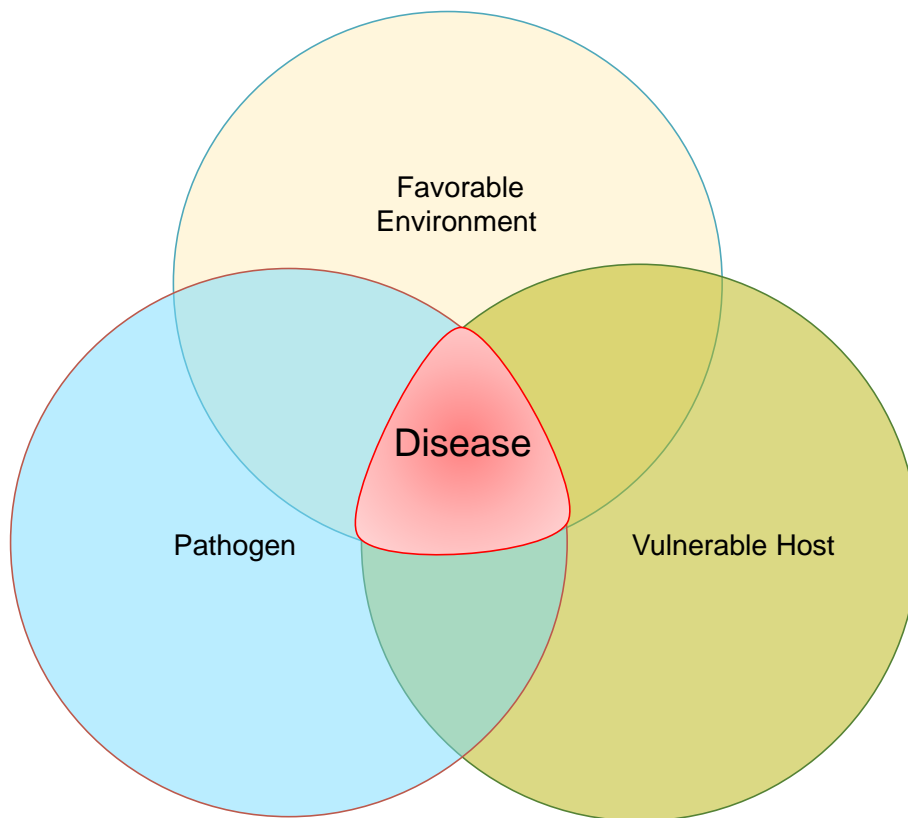


FIGURE 6.2. Disease triangle [204].

6.3. Proposed Solution

This paper proposes a novel automatic method, *aGROdet*, for identifying plant diseases and estimating the resulting leaf damage. Diseases can cause damage to plants at any stage of development and in any part of the plant. We propose a convolutional neural network-based method for disease detection and a novel pixel-based thresholding method for estimating the extent of leaf damage. Fields and plants can be monitored regularly using *aGROdet* to catch the disease early on.

How number of training data influences the accuracy has also been explored. In agriculture domain, public datasets are not always available or sufficient for several research problem areas. More publicly accessible datasets are required. This paper will guide the agricultural visual data collection process by showing the effect of training data on accuracy through a well known agricultural research problem.

6.4. aGROdet: Proposed Method for Detection of Plant Disease and Damage Estimation

6.4.1. Proposed A-CPS

The agriculture cyber physical system (A-CPS) [159] in Fig. 6.3 is used to detect plant diseases and estimate the severity of damage.

The A-CPS consists of physical systems and cyber systems. Things, stakeholders, and computing devices are the components of physical systems. In our case, UAV cameras and smart phone cameras are the things, single board computers and mobile phones computing devices, and stakeholders include microbiologists, plant pathologists, agriculture firms, farmers, and the Agriculture Research Service. Deep learning models, software, efficient data storage, and blockchain for data security are the components of cyber systems. It is distributed into two distinct platforms. Deep learning models and software are present both at the edge and in the cloud, with the remainder primarily in the cloud. Physical and cyber systems are linked via the network fabric. Depending on the location and range, the network fabric could be Sigfox, ZigBee, LoRa, Wi-Fi, 4G, or 5G.

We divide the work into two parts because *aGROdet* performs two tasks: plant

disease identification and damage severity estimation. The methods have been described in the following sections.

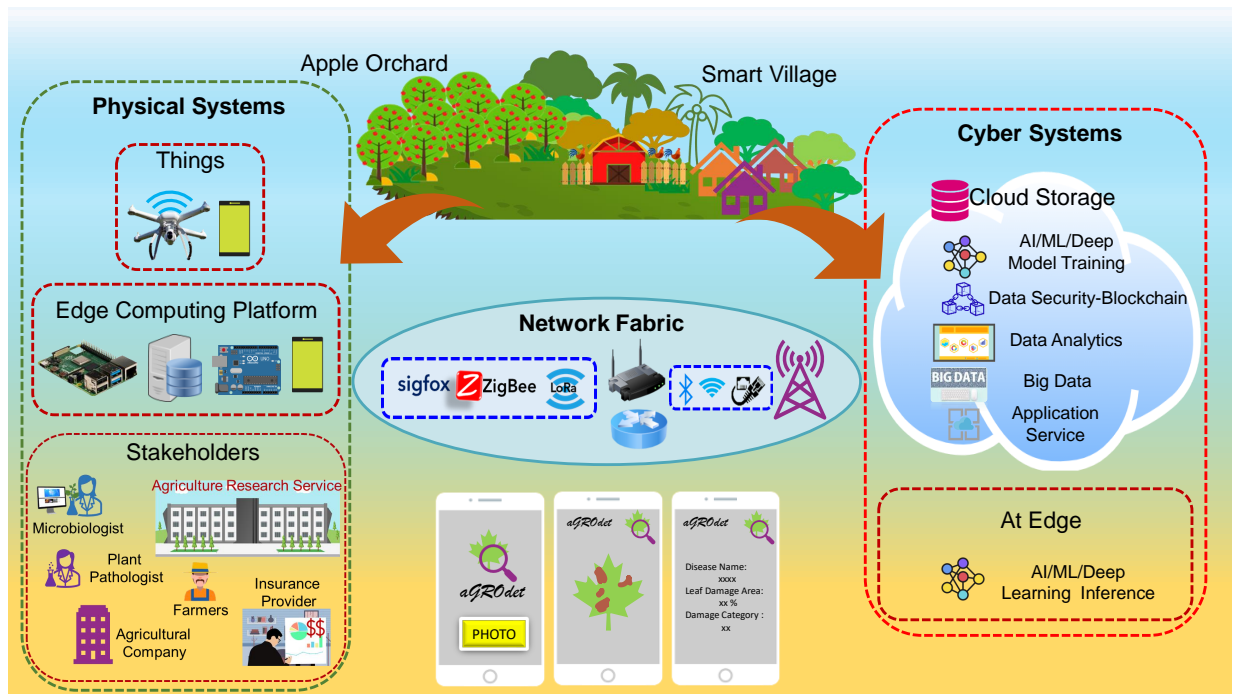


FIGURE 6.3. Agriculture cyber physical system.

6.4.2. Plant Disease Detection

6.4.2.1. Methodology

The proposed deep learning-based method for identifying plant diseases from leaf photos is described in this subsection. The model learns to label images using supervised learning techniques and predicts the label of an unknown image in this multi-class image classification task. The model learns the traits of the labeled images during training and categorizes unknown images with a confidence score. The model's capacity for classifying data, which in turn depends on how well the model has learned, determines the success of accurate prediction.

6.4.2.2. Network Architecture

For image classification, convolutional neural networks (CNN) are cutting-edge architectures. Here, a custom CNN, shown in Fig. 6.4, has been used for identifying plant

diseases. It comprises of 5 convolutional blocks. Each block has a *Convolutional* layer with *ReLU* activation, *BatchNormalization* layer, and *MaxPooling* layer. The number of filters in different convolutional layers varies: the first Conv2D layer has 32 filters, the next three blocks' Conv2D has 64 filters, and the last block Conv2D layer has 128 filters. The kernel sizes of the *Convolutional* layers are (3×3) with stride 1 and without zero padding. During inference, *BatchNormalization* layers only normalize the previous layer's output if they were trained on images of the same type as the testing data. Using a *MaxPooling* layer, the spatial dimensions have been diminished. With stride 2, the kernel size of the *MaxPooling* layer is 2×2 . The last block is followed by a *Flatten* layer and two *Dense* levels. The initial *Dense* layer employs *ReLU* activation and 1280 nodes, whereas the final layer employs *Softmax* activation and 39 nodes. The training of 6,117,287 of the 6,117,991 parameters. Table 6.1 describes in detail the output forms of the layers.

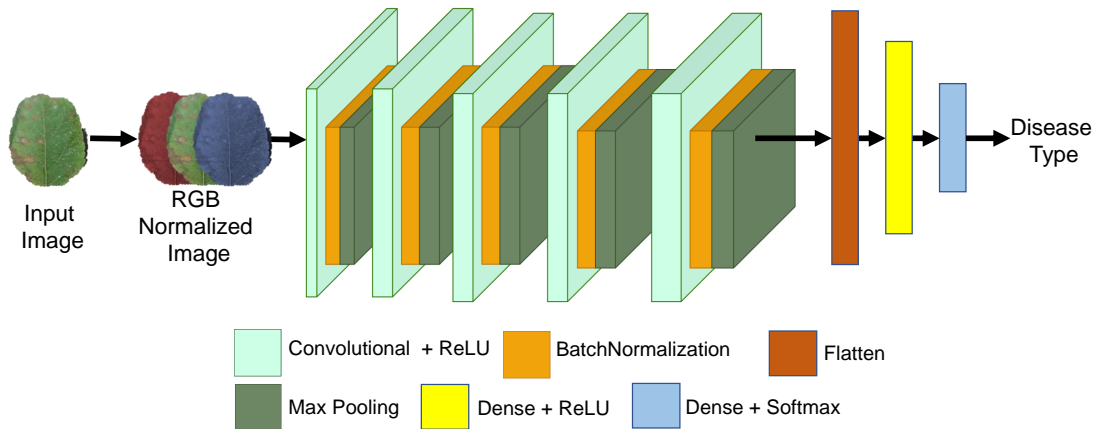


FIGURE 6.4. Plant disease detection network.

6.4.2.3. Experimental Validation

- (1) *Dataset Details:* For evaluation and training, the publicly accessible PlantVillage [108] dataset was used. The collection includes 55,448 images from 39 different categories. There is one class for images without leaves, as well as 38 classes for plant leaves. The method was evaluated with 5,562 images, while training and validation were done with 49,886 images. Some sample images from the dataset

are shown in Fig. 6.5. 80% - 20% distribution has been used for the training and validation.

TABLE 6.1. CNN Architecture for Plant Disease Identification.

Layers	Output Shape
Conv2D (f=32, k=3, s=1, p=0)	(254, 254, 32)
Activation: ReLU	
BatchNormalization	
Maxpooling2D (k=2, s=2)	(127, 127, 32)
Conv2D (f=64, k=3, s=1, p=0)	(125, 125, 64)
Activation: ReLU	
BatchNormalization	
Maxpooling2D (k=2, s=2)	(62, 62, 64)
Conv2D (f=64, k=3, s=1, p=0)	(60, 60, 64)
Activation: ReLU	
BatchNormalization	
Maxpooling2D (k=2, s=2)	(30, 30, 64)
Conv2D (f=64, k=3, s=1, p=0)	(28, 28, 64)
Activation: ReLU	
BatchNormalization	
Maxpooling2D (k=2, s=2)	(14, 14, 64)
Conv2D (f=128, k=3, s=1, p=0)	(12, 12, 128)
Activation: ReLU	
BatchNormalization	
Maxpooling2D (k=2, s=2)	(6, 6, 128)
Flatten	(4,608)
Dense (u=1280)	(1280,)
Dense (u=39)	(39,)

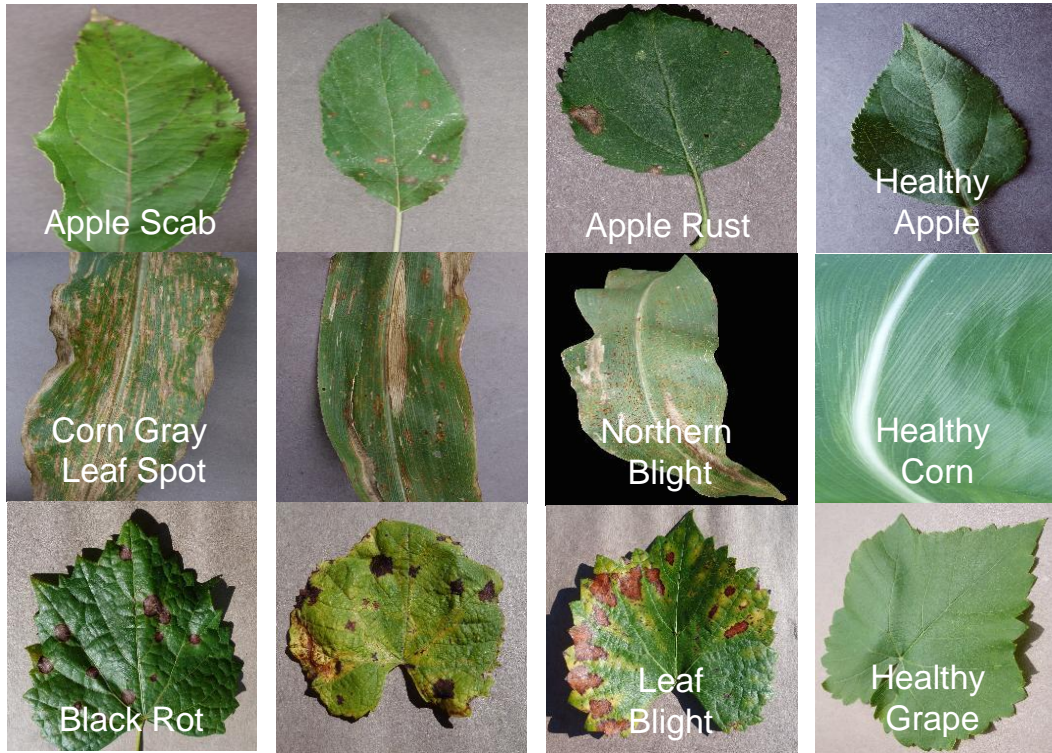


FIGURE 6.5. Sample images from PlantVillage dataset [108].

- (2) *Dataset Processing:* 256×256 RGB images have been used for training. Prior to sending the images to the network, the images have been normalized to prevent network slowdown by minimizing computation with big numbers during training. For improved and more accurate performance, training and validation data have undergone data augmentation. An example of augmented data is shown in Fig. 6.6. In order to create augmented data on the fly, image processing techniques like rotation, zoom, brightness, and horizontal and vertical ip have been applied.
- (3) *Experiment:* The process for identifying plant diseases is depicted in Fig. 6.7. The network is trained using the augmented and preprocessed data. With a starting learning rate of 0.001, the Adam optimizer [125] has been used. In order to train the model, 75 epochs were used, which means that the network was iterated 75 times across the entire dataset. The model has been trained with and without a reduced learning rate of factor 0.1. Once the model is trained, it is stored for further use

during inference. The model is analyzed by using the 5,562 additional images. The disease detection network in *aGROdet* has been implemented in Keras [51] with TensorFlow [26] back end.

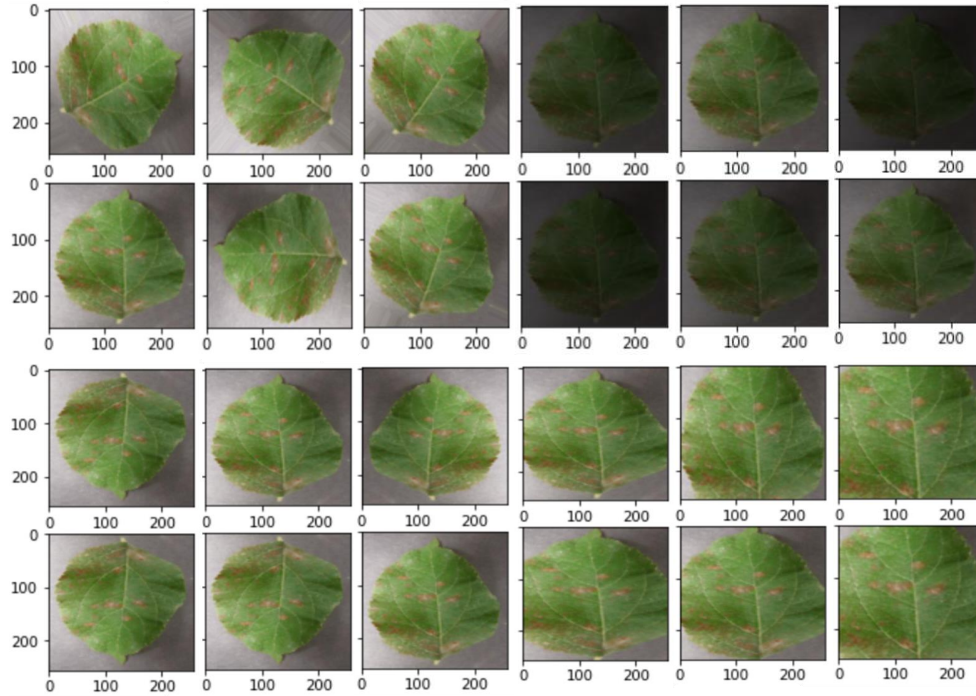


FIGURE 6.6. Sample augmented data. Data is augmented on the fly for different rotation, zoom, brightness, horizontal and vertical flip.

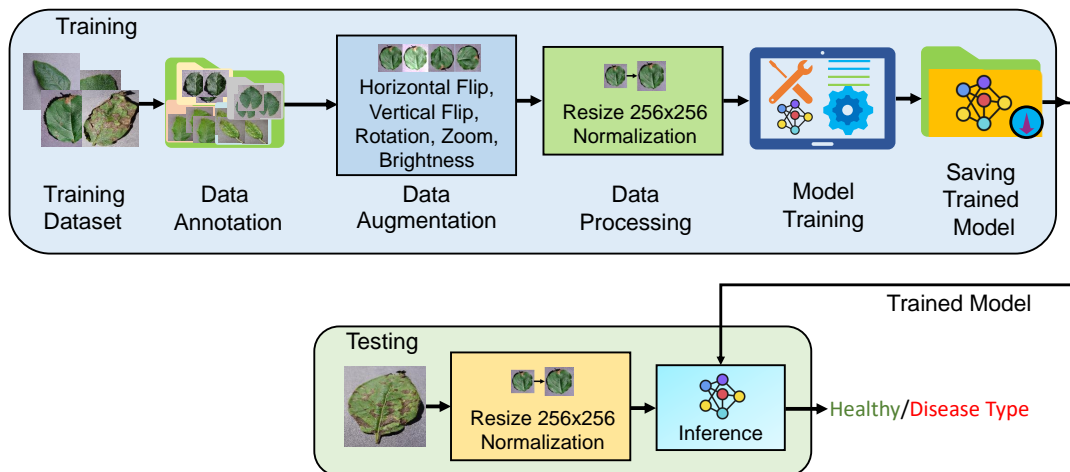


FIGURE 6.7. Plant disease detection workflow.

6.4.3. Estimation of Leaf Damage Severity

The method for estimating the severity of leaf damage is described in this subsection. Leaf area and damage area are estimated to determine the degree of the damage. The proportion of leaf damage is determined by the ratio between these two areas. The severity of the damage is finally predicted using a rule-based system. The method's pipeline is shown in Fig. 6.8.

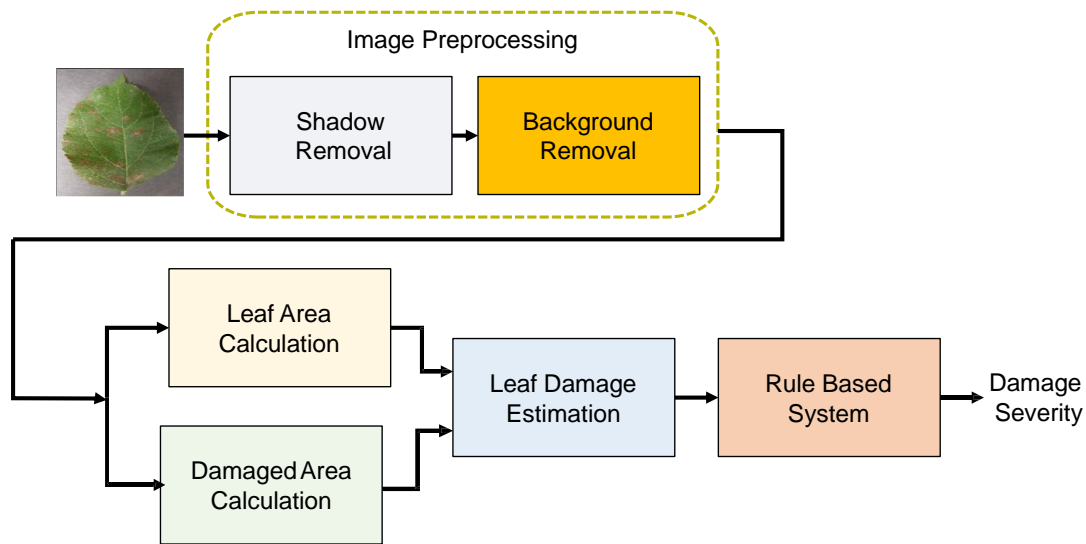


FIGURE 6.8. Leaf damage estimation workflow.

6.4.3.1. Leaf Area Detection

The first stage in determining the severity of leaf damage is leaf area detection. A leaf-specific mask is first constructed when the leaf area has been detected. The mask was created using thresholding and background segmentation. Finally, the mask's area is determined in order to get the leaf area.

- (1) *Background Segmentation:* The foreground object and the background make up the two sections of the leaf image. The foreground object, or leaf, is the subject of our attention. Using the GrabCut [187] technique, the background and leaf have been separated. When different foreground items are present, the algorithm's parameters and iteration count must be manually adjusted. However, because only a particular

type of object, in this example a leaf, is recognized, no manual correction is required. This technique starts by drawing an initial rectangle over the foreground object.

The defined background is the area outside of the rectangle. The foreground and some of the backdrop are contained inside the rectangle. In our work, we selected to create a sizable rectangle that was 226×226 in size while maintaining the image size of 256×256 as in Fig. 6.9(a). To guarantee that the entire foreground item or leaf remains inside the Region of Interest, a sizable rectangle is drawn (ROI).

The GrabCut [187] algorithm then applies a Gaussian Mixture Model (GMM) to the ROI after it has been defined. By comparing the colors of the pixels, they are grouped. Based on the distribution of pixels, a network is constructed with each pixel acting as a node. The reference nodes are two additional nodes. Foreground pixels are those that are connected to the *Source* node. However, the *Sink* node is connected to background pixels. The weights of the graph's edges are determined by the likelihoods of connecting to either *Source* or *Sink* nodes. Edges with higher weight values join similar pixel nodes. The total of the weights of the cut edges, which serves as the cost function, is minimized to separate the foreground pixels from the background pixels. To separate the leaf from its background, we repeated the operation 5 times. As seen in Fig. 6.9(b), the background pixels are colored black for the following stage of processing after segmentation.

- (2) *Thresholding and Leaf Area Detection:* On the leaves and all around them, there may be shadows. They affect how precisely leaves are detected. While the on-leaf shadows make it more difficult to create a flawless mask for the leaf, the outer shadow expands the area of the leaf. Smaller circles indicate on the leaf shadows, and the large red ovals in Fig. 6.9 indicate around the leaf shadows. We convert the leaf images from RGB color space to HSV color space because HSV color space distinguishes between image color (hue) and color intensity (value). The black color is then used for the thresholding, as seen in Fig. 6.9(c). The mask is reversed since the leaf in the front is what we are interested in.

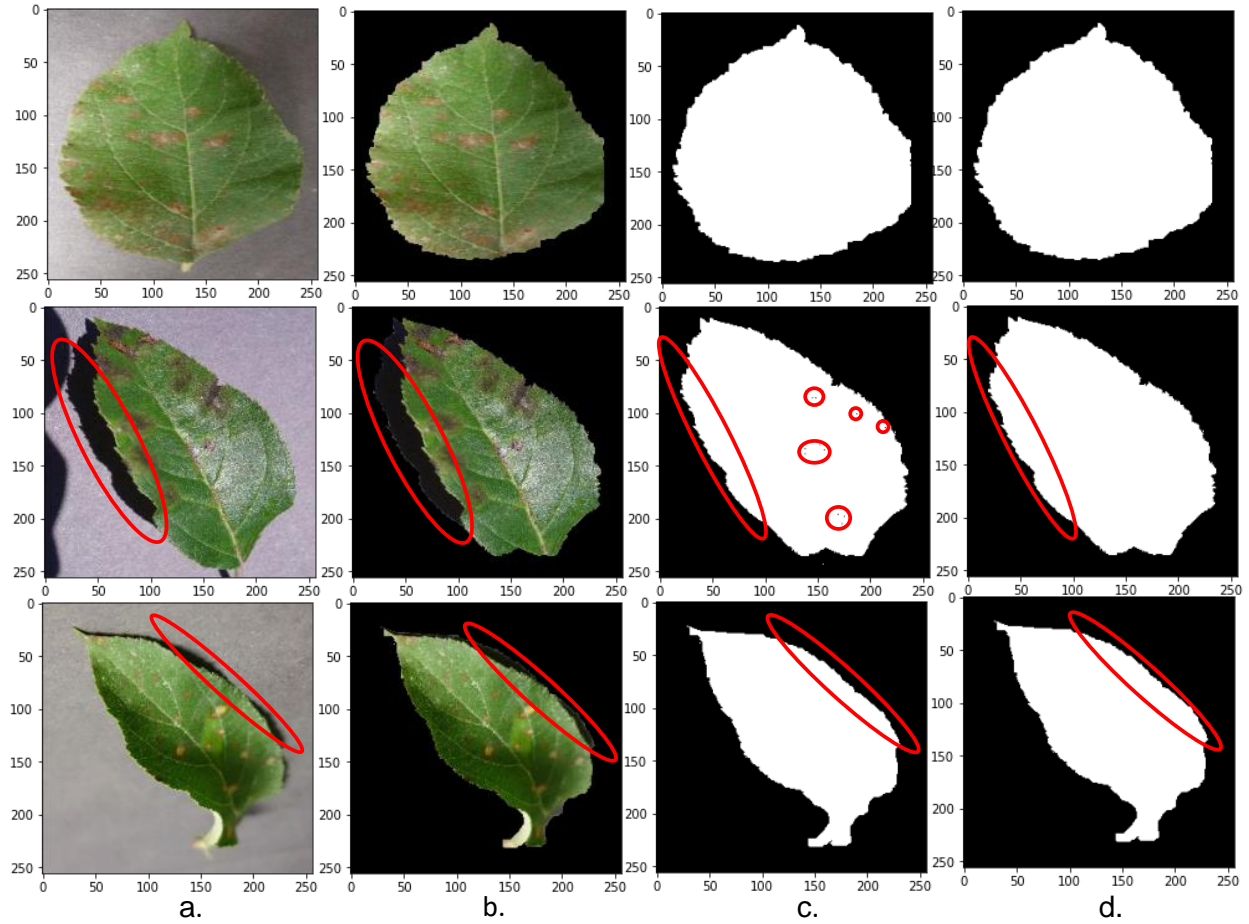


FIGURE 6.9. Leaf area detection by creating leaf mask. a. Input image b. background segmentation c. mask creation for the leaf d. noise reduction from the mask. Red large ovals show the shadow around the foreground object and small circles highlight the shadows on the foreground object.

However, several masks are depicted in small red circles in Fig. 6.9(c). We chose the largest contour of the foreground object to create a noise-free mask. A healthy leaf has a large contour, whereas a damaged leaf has a larger contour and several smaller contours depending on the severity of the damage. As a result, the largest contour from the foreground image is drawn over the mask, as shown in Fig. 6.9(d). It creates an ideal noise-free mask for the leaf.

As the foreground object, a leaf, is our object of interest, the mask is inverted. But several masks have noise due to specular reflection and shadows on the leaf.

This noise has been shown in small red circles in Fig. 6.9(c). To get a noise-free mask, we selected the largest contour of the foreground object. The healthy leaf consists of a large contour, whereas a damaged leaf has a larger contour and several smaller contours depending on the damage. Hence, the largest contour, selected from the foreground image, is drawn over the mask as in Fig. 6.9(d). It gives a perfect noise-free mask for the leaf.

(3) *Around the Leaf Shadow Removal:* Before background segmentation, shadows around the leaves were removed. To select the shadow, pixel-based thresholding is used, as shown in Fig. 6.10(b). As depicted in Fig. 6.10(c), the area around the leaf shadow part is then segmented from the foreground leaf during background segmentation. It is removed using contour selection during the final mask generation process, as shown in Fig. 6.10(d). Finally, in Fig. 6.10, the final mask is made noise free (e).

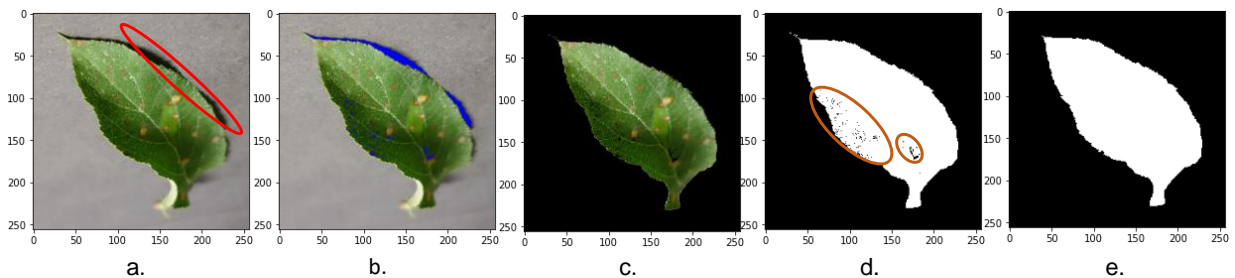


FIGURE 6.10. Removal of shadow around the leaf. a. Input image b. detection of shadow around the leaf c. shadow removal d. leaf mask creation e. noise reduction from the mask. Red large ovals show shadow around the leaf and brown ovals highlight the shadow on the leaf.

6.4.3.2. Damage Area Detection

Calculating the leaf damage area is also required to determine how severe the leaf damage is. The procedure is depicted in Figure 6.11. First, as shown in Fig. 6.11(b) and Fig. 6.11(c), the area around the leaf shadow is identified and eliminated. A mask is created for the leaf's green area as seen in Fig. 6.11(d), and it is bit-wise blended with the input

image as seen in Fig. 6.11(e). As shown in Fig. 6.11(f), the image's black background is then separated from the merged image and recolored with any other color to differentiate it from the damage. The damage mask is then created by performing pixel-based thresholding on the black color, as shown in Fig. 6.11(g).

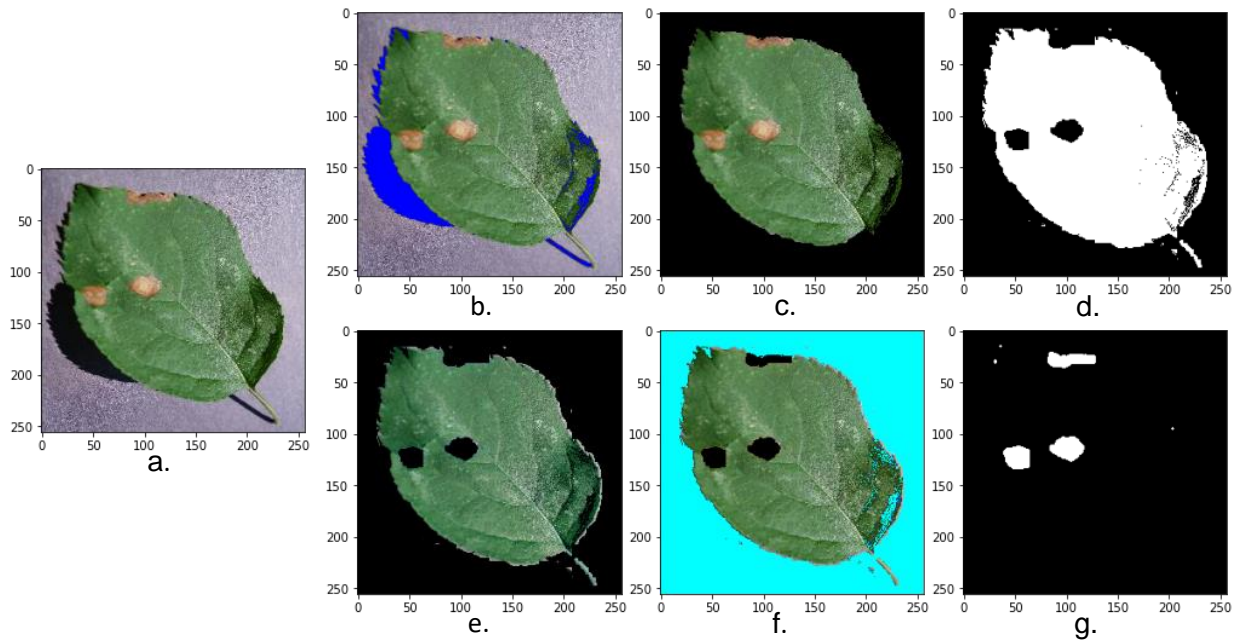


FIGURE 6.11. Leaf damage area detection a. Input image b. detection of shadow around the leaf c. shadow removal d. leaf mask creation e. merging of mask and input image f. recoloration of the black background to differentiate them from the damage g. damage mask creation.

6.4.3.3. Leaf Damage Estimation

The areas of the damage mask and leaf mask are determined in order to estimate leaf damage. To determine the area, the pixels that are present in the masks are measured. A sample area calculation and the projected percentage of damage to a leaf are shown in Fig. 6.12.

The severity of the damage to the leaf is then determined using a rule-based approach. Table 6.2 offers a suggested scale for grading the degree of damage. The technique predicts that the leaf is healthy if no damage is found. However, if the percentage of damage is higher

than 0, it categorizes the severity of the damage into several stages based on the numbers. The injured leaf in Fig. 6.12 has a damage severity grade of $Gr-1$ (between 0% and 5%) according to Table 6.2.

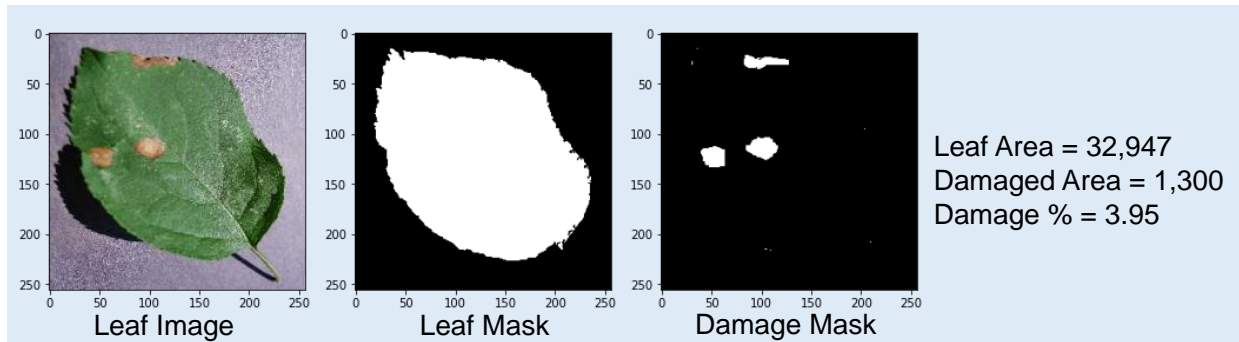


FIGURE 6.12. Leaf damage estimation.

TABLE 6.2. Grade Scale for Calculating Damage Severity.

Estimated Damage (%)	Damage Severity Grade
0	Healthy
>0 and ≤5	1
>5 and ≤10	2
>10 and ≤25	3
>25 and ≤50	4
> 50	5

6.4.4. Performance Evaluation of aGROdet

This section has described how well *aGROdet* performs in terms of identifying diseases and estimating their severity. For evaluation, unseen pictures from the PlantVillage Dataset [108] were used.

6.4.4.1. Disease Detection:

A number of indicators have been used to assess the model's performance. 5,562 un-viewed images of [108] were utilized to validate the model. The *confusion matrix* for this

multi-class problem is depicted in Fig. 6.13. As shown in Eqns. 6.1, 6.2, 6.3, and 6.4, several performance metrics [156] have been derived.

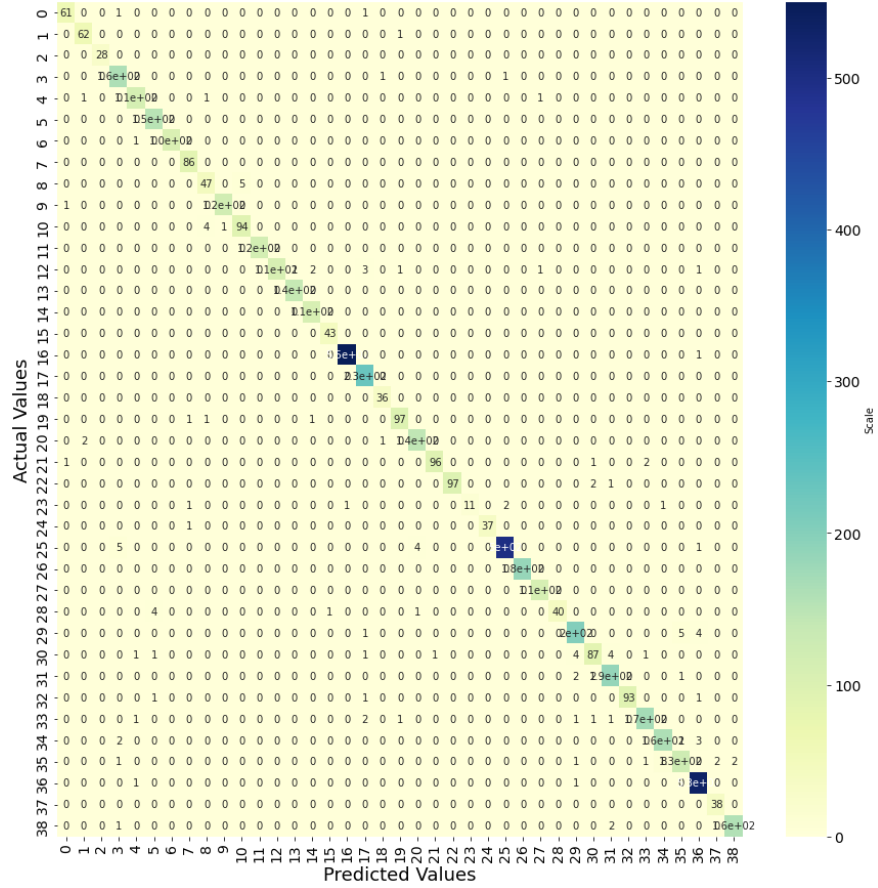


FIGURE 6.13. Confusion matrix for disease detection network (trained without reduced learning rate). Classes are denoted by numbers instead of the class names to fit into the figure space.

$$(6.1) \quad Accuracy = \left(\frac{TP + TN}{TP + TN + FP + FN} \right)$$

$$(6.2) \quad Precision = \left(\frac{TP}{TP + FP} \right)$$

$$(6.3) \quad Recall = \left(\frac{TP}{TP + FN} \right)$$

$$(6.4) \quad F1 - score = \left(\frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \right).$$

TP is *True Positive*, *TN* is *True Negative*, *FP* is *False Positive*, and *FN* is *False Negative*. *ROC* curves and *Precision-Recall* curves, two diagnostic curves, are also drawn. Such curves are shown in Figures 6.14(a) and 6.14(b) for only 8 classes. These evaluation tools were developed first for binary class issues. The *one vs. all* method has been used to derive these metrics and curves for multi-class situations, nevertheless. A weighted average precision of 98% has been achieved.

Table 6.3 displays the model’s accuracy for two distinct training circumstances. Better accuracy is attained when the model is trained with a reduced learning rate of factor 0.1.

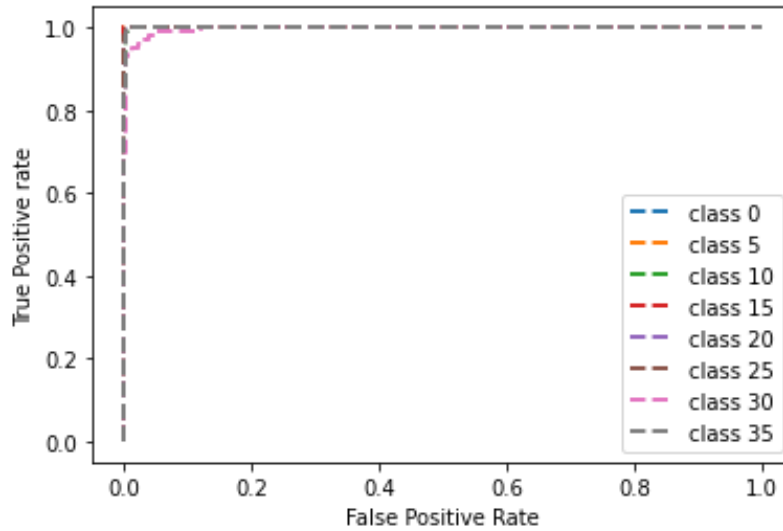
TABLE 6.3. Accuracy for Disease Detection Network.

Training Type	Data Augmentation	Accuracy (%)		
		Training	Validation	Testing
Without reduced learning rate	Yes	96.34	96.40	96.10
With reduced learning rate	Yes	98.89	98.41	98.58

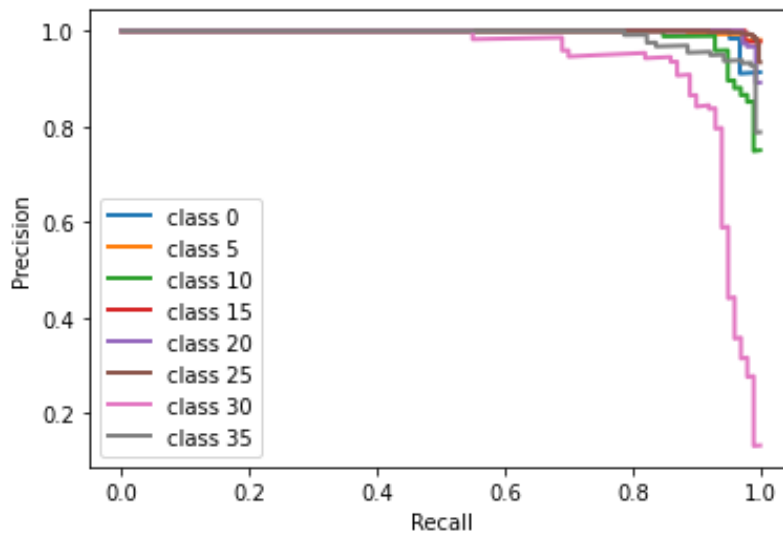
6.4.4.2. Effect of Data Insufficiency on Accuracy:

By varying the number of training images, the effect of data insufficiency has been observed. Table 6.4 shows how the number of training data affects the accuracy of the model. In the first case, the number of training and validation images were close to 50,000, though it was an imbalanced dataset. Data augmentation has been used to increase the total number of training data on the go. It significantly improved the validation accuracy. Test accuracy has also improved.

In the second row, the number of training and validation data has been kept constant, but no data has been augmented. In the third row, the total number of training data was halved. In both cases, accuracy decreased. The last row shows the result with Synthetic Minority Over-Sampling Technique (SMOTE) data augmentation. It took care of the imbalanced data, and the accuracy was higher even without a reduced learning rate.



(a) ROC curve.



(b) Precision vs recall curve

FIGURE 6.14. Performance evaluation curves for disease detection (trained without reduced learning rate).

Fig. 6.15 shows the accuracy and loss plots during training the model at various conditions. Fig. 6.15(a). shows the accuracy and loss plot for close to 50,000 samples. Data augmentation has been used to increase the number of training data. However, some more training epochs should have been added. Fig. 6.15(b). shows that without data augmentation, there is a gap between training and validation accuracy, though the model

was stabilized. The same is true for Fig. 6.15(c). The number of training and validation data, however, is half that shown in Fig. 6.15(b).

TABLE 6.4. Effect of Data Insufficiency on Accuracy (Trained without reduced learning rate).

No. of Training + Validation Data	Data Augmentation	Accuracy (%)		
		Training	Validation	Testing
49,886	Yes	96.34	96.40	96.10
49,886	No	98.06	96.17	94.23
25,685	No	97.84	92.33	89.25
49,886	SMOTE	97.62	97.42	97.68

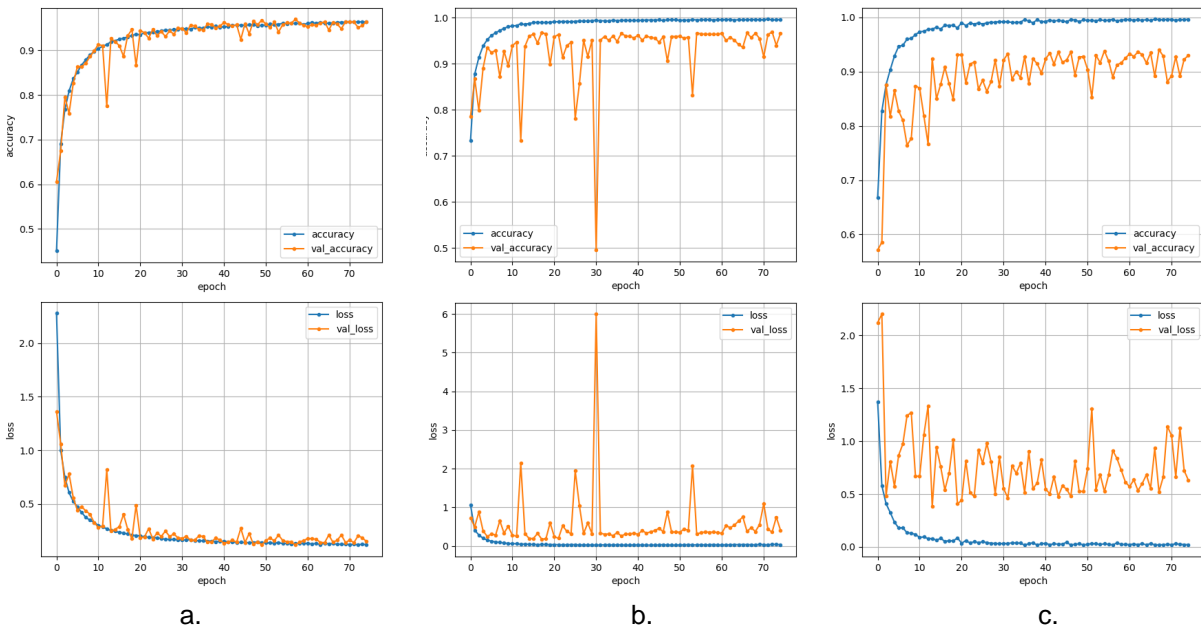


FIGURE 6.15. Accuracy and loss plots at various scenarios. a. data augmentation has been used. b. and c. no data augmentation has been used.

6.4.4.3. Leaf Damage Severity Estimation:

A part of the PlantVillage [108] dataset has been used to validate the approach. With regard to the dataset’s maize leaf images, no experiment has been conducted. In those situations, damage estimation will be inaccurate because the entire leaf is not visible in the image. Sample results are displayed in Table 6.5. The first column displays the tested images, while the second and third columns, respectively, display leaf and damage masks. In columns 4 and 5, the findings are listed. The estimated leaf damage presented in the fourth column of the table matches with the leaf and mask damage images of columns two and three.

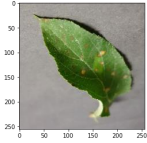
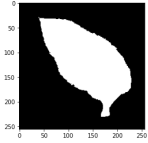
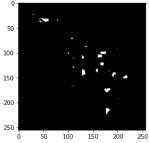
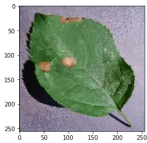
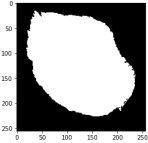
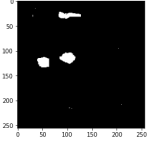
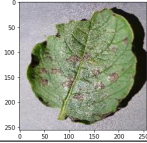
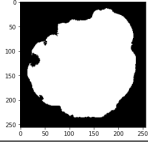
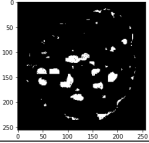
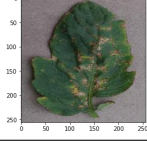
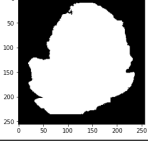
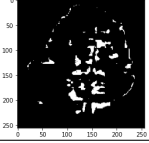
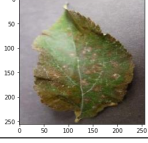
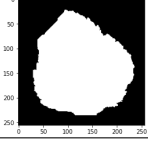
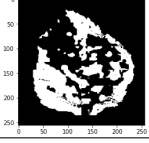
In general, shadows on and around the leaves negatively affect damage estimation. Damage masks in column three of Table 6.5 are accurately formed even in the presence of shadows, therefore damage estimation by *aGROdet* is unaffected. *aGROdet* can accurately evaluate the damage of leaves even if there is some specular reflection in the image.

In certain situations, like as with variegated plants, *aGROdet* may not accurately measure leaf damage. The healthy leaves of those plants also have different colors, such as yellow or white, in addition to green. Such variegated plants include the Abelia, Azalia, Boxwood, Cape Jasmine, Hydrangea, and Lilac. Our focus, however, is mostly on green-leaved plants and trees that are used to produce crops, fruits, and vegetables. If they are subjected to abiotic stress—a lack of nutrients in the soil, excessive or insufficient watering, excessive fertilizer use, extremely low temperatures, and insufficient light—they may become yellow. However, *aGROdet* can identify damage to those yellow components.

6.4.4.4. Comparative Analysis:

A comparison between *aGROdet* and other works is shown in Table 6.6. The question of disease severity was not covered in the majority of the studies. The disease severity issue has been addressed by [112], albeit with less success. However, an accurate leaf damage percentage has been achieved in our work along with the disease type. *aGROdet* gives a better perspective of leaf damage.

TABLE 6.5. Damage Severity Prediction through aGROdet.

Image	Leaf Mask	Damage Mask	Estimated Damage (%)	Damage Severity Grade
			2.97	1
			3.95	1
			9.49	2
			10.69	3
			53.49	5

6.5. Conclusions and Future Work

One of the main reasons for crop damage is plant disease. It slows down a plant's growth and keeps it from developing to its maximum potential. Therefore, it is crucial to detect plant diseases. Farmers need to understand the seriousness of the disease, though, in order to prevent it. In order to determine the severity of the disease, damage estimate is another crucial area of research. Our suggested *aGROdet* could be a helpful addition to programs for smart villages. In this paper:

For the purpose of detecting plant diseases and estimating leaf damage, we presented

the *aGROdet* system. Through the use of numerous performance measurements, we assessed our system. When it comes to identifying disease and estimating leaf damage, *aGROdet* has a very high success rate. *aGROdet* calculates the damage precisely even when there are shadows in the image. Even with some specular reflection, *aGROdet* provides accurate damage estimates. Effect of number of training data has also been observed.

TABLE 6.6. A Quantitative Analysis of the Current Paper with Existing Works.

Works	Disease Type	Accuracy (%)	Damage Estimation
Ji et al. [112]	Multi Disease	86.70	Yes
Mohanty et al. [163]	Multi Disease	99.35	No
Ji et al. [113]	Single	98.57	No
Wang [231]	Single	96.26	No
Ozguven et al. [173]	Single	95.48	No
Pallagani et al. [174]	Multi Disease	99.24	No
Current paper	Multi Disease	98.58	Yes

However, there are limitations to *aGROdet* which need further experimentation. In future work, these limitations are required to be addressed.

- [108] contains images of individual leaves. In reality, several leaves will appear in the same image when taken with a mobile 167 phone camera or UAV. As a result, before using *aGROdet*, a single leaf image must be detected from the shot image.
- *aGROdet*, as previously stated, does not estimate damage in variegated leaves. The inclusion of damage estimates for these plants would be a welcome addition.
- Another issue that requires attention is the extent of the damage.
- Disease can manifest itself in any part of the plant. Only the tops of the leaves are considered in this case. Other plant parts affected by disease must be considered in

the future.

- More work on removing shadows and specular reflections is required. This will improve damage estimation accuracy.
- The presence of pests on the leaf has not been taken into account. Incorporating damage estimation in the presence of the pest would also be an interesting task.
- Finally, more publicly available datasets will be beneficial to this research. Cleaner and more detailed datasets will guide the progress of data-centric AI initiatives.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This dissertation presented our contributions that provide ML/DL-based solutions overcoming various data quality aspects in different application fields for edge computing platforms. The contributions are presented in Chapter 3 to Chapter 6. In this chapter, we discuss and wind up each contribution along with potential limitations and future research directions.

7.1. Contribution I: Summary, Limitations, and Future Work

In Chapter 3, we presented efficient methods for social media deepfake video and image detection. Those solutions are aimed to be deployed at edge devices.

7.1.1. Detection of Deepfake Videos

In Section 3.5, we proposed a novel method of detecting deepfake videos in social media at any compression level applying key video frame approach and deep learning network. Our method involves fewer computations and is fit for deploying at edge platforms. A model, consisting of a convolutional neural network (CNN) and a classifier network, is proposed along with the algorithm. XceptionNet has been chosen over two other structures - InceptionV3 and Resnet50 - for pairing with the classifier. Our method is a visual artifact-based detection technique. The feature vectors from the CNN module are used as the input of the subsequent classifier network for classifying the video. We used the FaceForensics++ DF and DFDC datasets to reach the best model. Our model detects highly compressed deepfake videos in social media with a very high accuracy and low computational requirements. We achieved 98.5% accuracy with the FaceForensics++ DF dataset and 92.33% accuracy with a combined dataset of FaceForensics++ DF and Deepfake Detection Challenge. The simplicity of the method will help people to check the authenticity of a video. Our work is focused, but not limited, to addressing the social and economical issues due to fake videos in social media. Key video frame extraction method reduces the computations significantly,

as compared to existing works.

If there is only one key video frame in a test video, success rate of detecting the video is lower than the video with two or more key video frames.

Embedded deep learning is a growing field. Serious demand for various application domains is pushing today's cloud dependent deep learning area. As our algorithm detects fake videos by detecting key video frames, it substantially reduces the computation. So, it is one step forward to deploy a video detecting model at an edge device. But due to the memory limitation deep neural network structure is large to fit at the edge devices. So, reducing the run-time memory and the model size would be a great effort as the future work. It can be deployed at edge device by applying post-training float16 quantization and pruning.

7.1.2. Detection of Deepfake Images at Edge

In Section 3.6, a novel, memory-efficient lightweight ML-based method has been proposed to detect deepfake image at an edge device. As the method is pixel-based, it is more generic and can be applicable to any type of GAN. The model was successfully deployed in a Raspberry pi board. The method is fully automatic and accessible through the proposed detection API. The novelty of the work is achieving a considerable amount of accuracy with a short training time. No GPU was used. 48,000 images (24,000 deepfake images generated by StarGAN [50] and 24,000 real images from CelebA [148]) have been utilized for training and 12,000 images have been used for validation of the model. Total time for training and validation of the model was 27 minutes. AUC score of the ROC curve is 96%. The total time for sending the image to the edge, detecting, and displaying the result through the API is promising.

It is challenging to implement a computation-intensive computer vision problem like deepfake detection in an edge device. We tried to keep the computation as light as possible. We chose only 30 features for each image so that we could infer at a limited resource IoT device with considerable accuracy. Accuracy can be improved by increasing the number of trees and also by changing the feature set. With more features, the accuracy will be

higher and the generalization of the model will be achieved. To improve the inference time instead of sending the image in base64 format, binary image can be sent. As a future work, generalization of the model and higher accuracy can be achieved along with improved inference time.

In future, these two methods can be combined to eventually propose a deepfake video/image detecting tool for edge devices which can be accessed through an UI in mobile app.

7.2. Contribution II: Summary, Limitations, and Future Work

In Chapter 4, a deepfake resilient digital id system has been discussed. “Smart Cities” are a viable option to various issues caused by accelerated urban growth. To make smart cities a reality, smart citizens need to be connected to the “Smart City” through a digital ID. Biometric-based digital ID enables citizens to utilize smart city facilities healthcare, transportation, finance, and energy with ease and efficiency. However, deepfakes along with various presentation attacks, pose serious threat to the digital ID system. In the first part of this chapter, we proposed a deep learning-based method, iFace, for deepfake resilient digital id system of smart cities and in the latter half an improved version, iFace 1.1, with more robustness towards presentation attack.

7.2.1. iFace

iFace works in a cloud-edge setting, where encoded facial biometric data is sent from the cloud server to the edge device during authentication. The registration is done remotely and securely at the edge. The encoded bio key is kept on a cloud server. The digital ID is authenticated at the edge as well. Because it is done close to the user, it is immune to various indirect attacks. During authentication, a photo is taken at the edge of the device, and bio keys are generated using the stored registration bio key. At this point, the system will not allow any tampering by checking for deepfake attacks. It is impossible for people to maintain the same expression all the time. Our method can accommodate such modification of biometric data. For authentication, only a neutral frontal face (NFF) is required. Our

model is suitable for faces with minimal makeup. As the biometric of our proposed digital ID, we use facial features such as facial landmark points and specific distances in the eye and nose regions.

iFace generates an FRR of 2.77% and FAR of 0%. The accuracy of the deepfake attack system is 91%. It does not allow impostors to access users data. However, the accuracy of the deepfake detection method is not so high. Presentation attacks have also not been addressed. Authentication of an user involves computation spread in an edge-cloud environment. These limitations demand another improved version of the work i.e. iFace 1.1.

7.2.2. iFace 1.1

iFace 1.1 has been proposed to prevent biometric data movement over the network. The authentication process is performed at an edge device. User registration process is done remotely and in a secure way. In an offline application, the user's device is used to pull out facial embeddings. Since these embeddings are just a string of numbers made by a neural network, they can't be used to find out who they belong to. But to protect them even more, they are encrypted and sent to a remote cloud server run by the smart city. The authentication process is also done at edge devices. During authentication, no facial data is sent to the cloud. It makes the process resistant to indirect attacks. FSGAN deepfakes are a serious threat to people because they are easy to make and can be made quickly. Our system can find deepfakes that were created by FSGAN. Other presentation attack can also be found by the proposed system. When a user tries to use any facility in the smart city, a check is done by taking a picture of them at the time of use. This makes sure the right user is there and lowers the chance of a presentation attack. During authentication, no data is stored anywhere in the system. It's done as people come and go. This feature also reduces the chance of presentation attack. During authentication phase, none of the biometric information about the face leaves the edge platform or saved. Only the reference facial embeddings that are taken during enrollment are saved in the cloud, as encrypted data for future retraining. They never leave the cloud storage.

In this paper, several things have been achieved. Our system can detect intruder

attacks like presentation attacks and deepfake attack which is also a special type of presentation attack. The system is resistant against deepmorph deepfake attacks. It shows high accuracy with high quality face swapped GAN generated images. It can detect presentation threats with 93% accuracy. Face authentication is 97% accurate with FRR of 2% and FAR of 3%. Face authentication at the edge reduces security risk. No photos are stored anywhere in the system. Face features are stored in the cloud in terms of a numerical value. Hence, it is not possible to reverse engineer the photos from these numbers, which makes the process secure. Biometric data is stored without any identifiable information about the user, eliminating the data privacy regulation issues.

The deepfake detection module can currently identify face swapped images. Nonetheless, we plan to develop a full deepfake detection module to cover all potential deepfakes. More testing will also be done on the PAD module upgrade. The results of our proposed system are encouraging. Nonetheless, smart city rollout requires additional research and experimentation.

As for future work, a more efficient digital ID system can be achieved by addressing several aspects of the system. If the person ages noticeably, he or she will need to re-enroll face features to reflect these changes. Following a user's death, the system must initiate data deletion procedures. If a person's appearance has changed as a result of an injury or elective surgery, an option to update their data should also be made available. In the event of identity theft, the ability to have one's data wiped clean and to create a new user ID should be a standard feature.

Modern authentication systems need to be used, and the modules need update accordingly. A large dataset of human faces is required with images generated by various GANs to obtain a generalized deepfake detection module. The systems need to accommodate users of varying ages, races, colors, and genders. It's important to incorporate data from people who use glasses, piercings, head coverings, hearing aids, or braces in the training set.

This extensive training requires a lot of time and energy, and the hardware should be able to keep up. The current PAD module can identify faked photos of a person's face. The

PAD component must be evaluated in a variety of indoor, outdoor, daytime, and nighttime environments, as well as with a variety of head positions and lighting situations.

The use of challenge response techniques with arbitrary motion instructions (such as moving head or eyes, opening and closing mouth, or reading aloud any nonsense words) can help. As another layer of security, voice verification can be added. All facial piercings and other forms of facial occlusion, such as sunglasses, or mask should not affect the system's ability to authenticate a user. It's intriguing to think that the issue of identical twins might be solved.

7.3. Contribution III: Summary, Limitations, and Future Work

In Chapter 5, a ML-based especially deep neural network-based solution for a data scarcity problem in agriculture domain has been proposed. Agriculture is affected by natural calamities. Damage to crops causes significant financial losses for farmers. Many billions of dollars are lost every year because of climate and weather-related natural disasters. To make up for financial losses, crop insurance gives the agricultural sector a more secure financial footing. It takes long time to process a claim.

Machine learning and deep neural network-based models have been used to a wide variety of research problems in recent years thanks to advancements in AI. In order to make reliable predictions, deep neural networks require big training datasets. However, there is currently no publicly available dataset that can be used for estimating crop damage. Here we introduced a new approach to detecting crop damage, developed using a minimal dataset. It's the backbone of the *eCrop* approach for estimating crop damage on a grid. Our suggested agriculture cyber physical system includes a proof-of-concept implementation of the approach *eCrop*. The model used for detecting crop damage is a Convolutional Siamese Neural Network (CSNN). The model was trained using a meta-learning strategy. This has been done with a accuracy of 92.86%.

The method requires very few good quality training data samples. Our proposed *eCrop* system estimates the crop damage from the images taken by an Unmanned Aerial Vehicle (UAV). Hence, there is no need to set foot in the field for taking pictures. It

essentially reduces the risk of more damage to the crop land. The method is applicable to any stage of crop growth and to any crop type. We achieved high accuracy in detecting the damages caused by natural events. Our proposed proof-of-concept estimates the overall crop damage precisely. The data processing and computation is done at the edge server. Real time processing is also possible. In general deep learning based methods need a large number of data sets for training but application of artificial intelligence (AI) in agriculture is not in a mature state yet. As a result, the required data is not always available which in turn poses a bottleneck to transform agriculture to smart agriculture [153, 223]. However, our method does not suffer from the unavailability of data issue. The model for crop damage detection has been trained with very few data. Our work can be a promising method for the researches in agriculture domain when large datasets are not available.

As future work, integration of blockchain and PUF-based methods will be explored for robust cyber-attack resilient smart agriculture Cyber-Physical System (A-CPS). Implementation of the end-to-end *eCrop* system will be an important and relevant future work for estimating crop damage due to natural disaster. As eCrop system reduces the work of the loss adjuster by making the process automated with high accuracy, we believe our work has the potential to be applied to assess the crop damage in practice.

7.4. Contribution IV: Summary, Limitations, and Future Work

Contribution IV is the extension of Contribution III. This paper investigates the amount of data required to obtain acceptable accuracy in a machine learning model. As the problem of data scarcity is studied in the agriculture domain, a similar situation was used for this verification. In this instance, how inadequate data lowers the accuracy of a typical deep learning model, has been studied. Finally, A successful high accuracy model has been proposed.

One of the leading causes of crop damage is plant disease. It inhibits the growth of plants and prevents them from achieving their full potential. Hence, plant disease detection is vital. However, in order to prevent the disease, farmers must be aware of its severity. Determining the severity of the condition necessitates further investigation of the evaluation

of the harm.

We proposed a novel method, *aGROdet*, to detect plant disease and to estimate the leaf damage severity. The optimum data needed for an acceptable accuracy has also been studied. *aGROdet* is aimed at being implemented at the edge platform of IoT systems in the proposed Agriculture Cyber Physical System. A convolutional neural network-based model has been proposed to detect different plant diseases. The model has been trained with a publicly available datasets. The number of training images has been varied to see the effects of the number of training data. More than 97% accuracy has been achieved in the initial phase of the experiment. A pixel-based thresholding method has been used for estimating the severity of the damage. Damage estimation limiting factors, such as on the leaf and around the leaf shadows, have also been addressed. *aGROdet* accurately estimates damage, even in the presence of some specular reflection. This is a very useful tool for farmers who can detect plant diseases with an estimation of plant damage on their own. No expert knowledge is required. We hope that *aGROdet* will help farmers take proper control measures and save time, money, and secondary plant losses.

However, *aGROdet* has some shortcomings that call for additional research and testing. In subsequent work, consideration needs to be given to how best to overcome these limitations. The dataset has images of single leaves. In fact, when photographs are captured with a mobile phone camera or UAV, multiple leaves will appear in a single image. Before using *aGROdet*, a single image of a leaf must be extracted from the captured image. As indicated previously, *aGROdet* is unable of estimating damage to variegated leaves. Damage estimates for these plants would be a welcome addition. The extent of the damage is yet another area requiring attention. This topic has not been addressed. Disease can affect any portion of a plant. Here, only the leaf tops are taken into account. In the future, other plant sections affected by disease must be considered. More effort is required to eliminate shadows and specular reflections. This will improve the precision of damage estimates. On the leaf, the existence of pests has not been considered. Incorporating the calculation of damage in the presence of the pest would also be an intriguing undertaking. Lastly, more

publicly accessible datasets will be a crucial complement to this study. Cleaner and more informative datasets will facilitate the advancement of data-centric AI projects.

7.5. Discussions

In this Section, we discuss how different chapters work together and formulate the bigger problem of data quality aspects. There are many ways to define data quality. However, we define “data quality” as the condition of qualitative and quantitative information in a dataset. Among various elements of data quality, three aspects - misleading fake data especially deepfake images and videos, data scarcity, and data insufficiency - have been explored. This research aims to provide efficient high accuracy ML or DL-based methods to solve the aforementioned data quality related issues. Different application domains where the selected aspects pose issues have been chosen. To address the issues of data privacy, security, and regulation, these solutions have been deployed at edge devices. We hope that this dissertation will contribute to research on the application of ML methods in areas where data quality is a barrier to success.

REFERENCES

- [1] *Climate Change: How Do We Know?*, <https://climate.nasa.gov/evidence/>, Accessed on 29 December, 2021.
- [2] *Climate Smart Agriculture*, Accessed on 18 January, 2022.
- [3] *Corn Kernel Damage*., Accessed on 10 December, 2021.
- [4] *Cyber-Physical Systems Executive Summary*, Accessed on 27 February 2022.
- [5] *Dataset: USDA*, Accessed on 10 December, 2021.
- [6] *Deep Learning for Computer Vision*, https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture01.pdf, Accessed on 25 August, 2022.
- [7] *DeepFaceLab*, Retrieved from <https://github.com/iperov/DeepFaceLab>, Accessed on 19 January 2021.
- [8] *Deepfake Video*, Retrieved from <https://edition.cnn.com/2019/06/11/tech/zuckerberg-deepfake/index.html>, Accessed on 19 January 2021.
- [9] *DFaker*, <https://github.com/dfaker/df>, Accessed on 19 January 2021.
- [10] *Disaster Analysis*, https://www.nass.usda.gov/Research_and_Science/Disaster-Analysis/, Accessed on 23 December, 2021.
- [11] *Faceswap*, <https://github.com/deepfakes/faceswap>, Accessed on 19 January 2021.
- [12] *Groundwater Nitrate Contamination*, <https://prd-wret.s3.us-west-2.amazonaws.com/assets/palladium/production/s3fs-public/thumbnails/image/wss-nitrogen-map-us-risk-areas.jpg>, Accessed on 23 December, 2021.
- [13] *IEEE Smart Village Map*, <https://smartvillage.ieee.org/our-projects/>, Accessed April 6, 2022.
- [14] *image "hacker"*, image: freepik.com, Accessed on 07 June 2021.
- [15] *IPCC Sixth Assessment Report, Summary for Policymakers*, Accessed on 29 December, 2021.
- [16] *Mathworks*, <https://www.mathworks.com/help/images/>

- [texture-analysis-using-the-gray-level-co-occurrence-matrix-gldm.html](#),
Accessed 28 January 2021.
- [17] *Natural Disasters and Crop Insurance*, Accessed on 30 December, 2021.
- [18] *Plant Disease*, Accessed on 23 December, 2021.
- [19] *Plant Disease: Pathogens and Cycles*, <https://cropwatch.unl.edu/soybean-management/plant-disease>, Accessed March 31, 2022.
- [20] *Plant Diseases*, <https://www.ars.usda.gov/crop-production-and-protection/plant-diseases/docs/action-plan-2022-2026/>, Accessed April 4, 2022.
- [21] *Soil Health*, <https://new.cloudvault.usda.gov/index.php/s/7iknp275KdTKwCA>,
Accessed on 23 December, 2021.
- [22] *Data Quality vs Data Quantity: What's More Important for AI? - TechNative*, nov 1 2018, Accessed on 29 June 2022.
- [23] *DARPA News*, <https://www.darpa.mil/news-events/2021-03-02>, March 2021, Accessed 07 May 2021.
- [24] *Computer Vision vs. Machine Learning: What are the Differences?*, <https://kili-technology.com/blog/computer-vision-and-machine-learning-differences>, February 2 2022, Accessed on 21 September 2022.
- [25] *Report: Cisco and IBM leaders in the smart cities technology market*, November 21, 2014, Accessed: February 24, 2022.
- [26] 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, Software available from tensorflow.org.
- [27] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen, *Mesonet: a compact facial video forgery detection network*, Proceedings of the IEEE International Workshop on Information Forensics and Security (WIFS), 2018, doi: 10.1109/WIFS.2018.8630761, pp. 1–7.
- [28] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen, *Face description with local binary patterns: Application to face recognition*, IEEE transactions on pattern analysis and machine intelligence 28 (2006), no. 12, 2037–2041, doi: 10.1109/TPAMI.2006.244.
- [29] Hunt Allcott and Matthew Gentzkow, *Social media and fake news in the 2016 election*, Journal of Economic Perspectives 31 (Spring 2017), no. 2, pp. 211–36.
- [30] Rockwell Anyoha, *The History of Artificial Intelligence - Science in the News*, <https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/>, Aug 28 2017, Accessed on 29 June 2022.
- [31] Martin Arjovsky, Soumith Chintala, and Léon Bottou, *Wasserstein GAN*, arXiv: 1701.07875 (2017).
- [32] Christian Baccarella, Timm Wagner, Jan Kietzmann, and Ian McCarthy, *Social media? it's serious! understanding the dark side of social media*, European Management Journal 36 (2018), 431–438, doi: 10.1016/j.emj.2018.07.002.
- [33] P. Baldi and Peter Sadowski, *Understanding dropout*, Advances in Neural Information Processing Systems, 2013.
- [34] Shubhajit Basak, Peter Corcoran, Faisal Khan, Rachel McDonnell, and Michael Schukat, *Learning 3D Head Pose From Synthetic Data: A Semi-Supervised Approach*, IEEE Access 9 (2021), 37557–37573, doi: 10.1109/ACCESS.2021.3063884.
- [35] Peter N. Belhumeur, Joao P Hespanha, and David J. Kriegman, *Eigenfaces vs. fisherfaces: Recognition using class specific linear projection*, IEEE Transactions on pattern analysis and machine intelligence 19 (1997), no. 7, 711–720, doi: 10.1109/34.598228.
- [36] Jordan Bell, Esayas Gebremichael, Andrew Molthan, Lori Schultz, Franz Meyer, and

- Suravi Shrestha, *Synthetic Aperture Radar and Optical Remote Sensing of Crop Damage Attributed to Severe Weather in the Central United States*, Proceedings of the IEEE International Geoscience and Remote Sensing Symposium(IGARSS 2019), 2019, doi: 10.1109/IGARSS.2019.8899775, pp. 9938–9941.
- [37] F. Bellard and M. Niedermayer, *Ffmpeg*, <http://ffmpeg.org>, 2012.
- [38] Prakruti V Bhatt, Sanat Sarangi, and Srinivasu Pappula, *Detection of diseases and pests on images captured in uncontrolled conditions from tea plantations*, Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping IV, vol. 11008, International Society for Optics and Photonics, 2019, p. 1100808.
- [39] Namrata R Bhimte and VR Thool, *Diseases detection of cotton leaf spot using image processing and SVM classifier*, Proceedings of the Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018, pp. 340–344.
- [40] Mohammed Brahim, Saïd Mahmoudi, Kamel Boukhalfa, and Abdelouhab Moussaoui, *Deep interpretable architecture for plant diseases classification*, Proceedings of the Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2019, pp. 111–116.
- [41] C. Bregler, M. Covell, and M. Slaney, *Video rewrite: Driving visual speech with audio*, Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, 1997, pp. 353–360.
- [42] Andrew Brock, Jeff Donahue, and Karen Simonyan, *Large scale GAN training for high fidelity natural image synthesis*, CoRR abs/1809.11096 (2018).
- [43] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah, *Signature verification using a “siamese” time delay neural network*, International Journal of Pattern Recognition and Artificial Intelligence 7 (1993), no. 04, 669–688.
- [44] Francisco Ceballos, Berber Kramer, and Miguel Robles, *The feasibility of picture-based insurance (PBI): Smartphone pictures for affordable crop insurance*, Development Engineering 4 (2019), 100042, doi: <https://doi.org/10.1016/j.deveng.2019.100042>.

- [45] Polychronis Charitidis, Giorgos Kordopatis-Zilos, Symeon Papadopoulos, and Ioannis Kompatsiaris, *Investigating the impact of pre-processing and prediction aggregation on the deepfake detection task*, 2020.
- [46] B. Chen and V. Chandran, *Biometric based cryptographic key generation from faces*, Proceedings of the 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007), 2007, doi: 10.1109/DICTA.2007.4426824, pp. 394–401.
- [47] Huangxun Chen, Wei Wang, Jin Zhang, and Qian Zhang, *EchoFace: Acoustic Sensor-Based Media Attack Detection for Face Authentication*, IEEE Internet of Things Journal 7 (2020), no. 3, 2152–2159, doi: 10.1109/JIOT.2019.2959203.
- [48] Yimin Chen, Jingchao Sun, Xiaocong Jin, Tao Li, Rui Zhang, and Yanchao Zhang, *Your face your heart: Secure mobile face authentication with photoplethysmograms*, Proceedings of the IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, doi: 10.1109/INFOCOM.2017.8057220, pp. 1–9.
- [49] Ivana Chingovska, André Anjos, and Sébastien Marcel, *On the effectiveness of local binary patterns in face anti-spoofing*, Proceedings of the International Conference of Biometrics Special Interest Group (BIOSIG), 2012, pp. 1–7.
- [50] Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo, *Stargan: Unified generative adversarial networks for multi-domain image-to-image translation*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8789–8797.
- [51] François Chollet, *Keras*, <https://keras.io>, 2015, Accessed on 15 July 2021.
- [52] François Chollet, *Xception: Deep learning with depthwise separable convolutions*, CoRR abs/1610.02357 (2016).
- [53] Jayraj Chopda, Hiral Raveshiya, Sagar Nakum, and Vivek Nakrani, *Cotton crop disease detection using decision tree classifier*, Proceedings of the International Conference on Smart City and Emerging Technology (ICSCET), 2018, pp. 1–5.
- [54] H. T. Grace Chou and Nicholas Edge, *They are happier and having better lives than*

- i am*”: *The impact of using facebook on perceptions of others’ lives*, *Cyberpsychology, Behavior, and Social Networking* 15 (February 2012), no. 2.
- [55] Sarit Chung, Junichiro Takeuchi, Masayuki Fujihara, and Chantha Oeurng, *Flood damage assessment on rice crop in the Stung Sen River Basin of Cambodia*, *Paddy and Water Environment* 17 (2019), 255–263, doi: 10.1007/s10333-019-00718-1.
- [56] Danielle K Citron and Robert Chesney, *Deepfakes: A looming crisis for national security, democracy and privacy?*, *Lawfare* (2018).
- [57] Davide Alessandro Coccomini, Nicola Messina, Claudio Gennaro, and Fabrizio Falchi, *Combining efficientnet and vision transformers for video deepfake detection*, *Proceedings of the International Conference on Image Analysis and Processing*, Springer, 2022, pp. 219–229.
- [58] Michael Copeland, *What’s the Difference Between Artificial Intelligence, Machine Learning and Deep Learning?*, <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>, July 29 2016, Accessed on 21 September 2022.
- [59] Navneet Dalal and Bill Triggs, *Histograms of oriented gradients for human detection*, *Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, vol. 1, IEEE, 2005, doi: 10.1109/CVPR.2005.177, pp. 886–893.
- [60] L. D’Amiano, D. Cozzolino, G. Poggi, and L. Verdoliva, *A PatchMatch-based dense-field algorithm for video copy-move detection and localization*, *IEEE Transactions on Circuits and Systems for Video Technology* 29 (2019), no. 3, 669–682.
- [61] Amit Degada, Himanshu Thapliyal, and Saraju P Mohanty, *Smart Village: An IoT Based Digital Transformation*, *Proceedings of the IEEE 7th World Forum on Internet of Things (WF-IoT)*, 2021, pp. 459–463.
- [62] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, *Imagenet: A large-scale hierarchical image database*, *Proceedings of the IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

- [63] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou, *Retinaface: Single-Shot Multi-Level Face Localisation in the Wild*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, doi: 10.1109/CVPR42600.2020.00525, pp. 5203–5212.
- [64] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, *Arcface: Additive Angular Margin Loss for Deep Face Recognition*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, doi: 10.1109/CVPR.2019.00482, pp. 4690–4699.
- [65] Jiankang Deng, Yuxiang Zhou, and Stefanos Zafeiriou, *Marginal Loss for Deep Face Recognition*, Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2017, doi: 10.1109/CVPRW.2017.251, pp. 60–68.
- [66] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805 (2018).
- [67] Liping Di, Eugene Yu, Ranjay Shrestha, and Li Lin, *DVDI: A New Remotely Sensed Index for Measuring Vegetation Damage Caused by Natural Disasters*, Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2018), 2018, doi: 10.1109/IGARSS.2018.8518022, pp. 9067–9069.
- [68] X. Ding, G. Yang, R. Li, L. Zhang, Y. Li, and X. Sun, *Identification of motion-compensated frame rate up-conversion based on residual signals*, IEEE Transactions on Circuits and Systems for Video Technology 28 (2018), no. 7, 1497–1512.
- [69] Trong-Le Do, Mai-Khiem Tran, Huy H. Nguyen, and Minh-Triet Tran, *Potential Threat of Face Swapping to eKYC with Face Registration and Augmented Solution with Deepfake Detection*, Future Data and Security Engineering (Cham) (Tran Khanh Dang, Josef Küng, Tai M. Chung, and Makoto Takizawa, eds.), Springer International Publishing, 2021, doi: https://doi.org/10.1007/978-3-030-91387-8_19, pp. 293–307.
- [70] Yahya Dogan and Hacer Yalim Keles, *Semi-supervised image attribute editing using generative adversarial networks*, Neurocomputing 401 (2020), 338–352.

- [71] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer, *The deepfake detection challenge dataset*, 2020.
- [72] Bhaskara S. Egala, Ashok K. Pradhan, Venkataramana Badarla, and Saraju P. Mohanty, *Fortified-Chain: A Blockchain-Based Framework for Security and Privacy-Assured Internet of Medical Things With Effective Access Control*, IEEE Internet of Things Journal 8 (2021), no. 14, 11717–11731, doi: 10.1109/JIOT.2021.3058946.
- [73] Ellen Gray, *Global Climate Change Impact on Crops Expected Within 10 Years, NASA Study Finds*, 2021, Accessed on 01 January, 2022.
- [74] Emily Sohn, *Climate change and the rise and fall of civilizations*, 2014, Accessed on 01 January, 2022.
- [75] Mohammed E Fathy, Vishal M Patel, and Rama Chellappa, *Face-based active authentication on mobile devices*, Proceedings of the IEEE international conference on acoustics, speech and signal processing (ICASSP), 2015, doi: 10.1109/ICASSP.2015.7178258, pp. 1687–1691.
- [76] Konstantinos P. Ferentinos, *Deep learning models for plant disease detection and diagnosis*, Computers and Electronics in Agriculture 145 (2018), 311–318, doi: 10.1016/j.compag.2018.01.009.
- [77] Mohamed Amine Ferrag, Leandros Maglaras, and Abdelouahid Derhab, *Authentication and authorization for mobile IoT devices using biofeatures: Recent advances and future trends*, Security and Communication Networks 2019 (2019), 20 pages, doi: 10.1155/2019/5452870.
- [78] Ipek Ganiyusufoglu, L. Minh Ngô, Nedko Savov, Sezer Karaoglu, and Theo Gevers, *Spatio-temporal features for generalized detection of deepfake videos*, 2020.
- [79] P. Garrido, L. Valgaerts, O. Rehmsen, T. Thormaehlen, P. Perez, and C. Theobalt, *Automatic face reenactment*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 4217–4224.
- [80] Erik Gerstner, *Face/off: “DeepFake” face swaps and privacy laws*, Defense Counsel Journal 87 (2020), no. 1.

- [81] A. Gironi, M. Fontani, T. Bianchi, A. Piva, and M. Barni, *A video forensic technique for detecting frame deletion and insertion*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 6226–6230.
- [82] Ross Girshick, *Fast R-CNN*, 2015, doi: 10.48550/ARXIV.1504.08083.
- [83] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
- [84] David Güera and Edward J. Delp, *Deepfake video detection using recurrent neural networks*, Proceedings of the 15th IEEE International Conference on Advanced Video and Signal Based Surveillance, 2018, pp. 1–6.
- [85] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville, *Improved training of Wasserstein GANs*, Proceedings of Advances in Neural Information Processing Systems, 2017, p. 5767–5777.
- [86] Suyog Gupta and Mingxing Tan, *EfficientNet-EdgeTPU: Creating Accelerator-Optimized Neural Networks with AutoML*, <https://ai.googleblog.com/2019/08/efficientnet-edgetpu-creating.html>, Accessed: January 30, 2022.
- [87] A. Hadid, J. Y. Heikkilä, O. Silven, and M. Pietikainen, *Face and Eye Detection for Person Authentication in Mobile Phones*, Proceedings of the First ACM/IEEE International Conference on Distributed Smart Cameras, 2007, doi: 10.1109/ICDSC.2007.4357512, pp. 101–108.
- [88] Raia Hadsell, Sumit Chopra, and Yann LeCun, *Dimensionality reduction by learning an invariant mapping*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), vol. 2, 2006, pp. 1735–1742.
- [89] R. M. Haralick, K. Shanmugam, and I. Dinstein, *Textural features for image classification*, IEEE Transactions on Systems, Man, and Cybernetics SMC-3 (1973), no. 6, 610–621.

- [90] H. R. Hasan and K. Salah, *Combating deepfake videos using blockchain and smart contracts*, IEEE Access 7 (2019), 41596–41606.
- [91] Md Jahid Hasan, Shamim Mahbub, Md Shahin Alom, and Md Abu Nasim, *Rice disease identification and classification by integrating support vector machine with deep convolutional neural network*, Proceedings of the 1st international conference on advances in science, engineering and robotics technology (ICASERT), 2019, pp. 1–6.
- [92] M. F. Hashmi, B. K. K. Ashish, A. G. Keskar, N. D. Bokde, J. H. Yoon, and Z. W. Geem, *An exploratory analysis on visual counterfeits using conv-lstm hybrid architecture*, IEEE Access 8 (2020), 101293–101308.
- [93] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
- [94] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, *Mask R-CNN*, 2017, doi: 10.48550/ARXIV.1703.06870.
- [95] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep residual learning for image recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, doi: 10.1109/CVPR.2016.90, pp. 770–778.
- [96] P. He, H. Li, and H. Wang, *Detection of fake images via the ensemble of deep representations from multi color spaces*, Proceedings of the IEEE International Conference on Image Processing, 2019, pp. 2299–2303.
- [97] Xiaofei He, Shuicheng Yan, Yuxiao Hu, Partha Niyogi, and Hong-Jiang Zhang, *Face recognition using Laplacianfaces*, IEEE transactions on pattern analysis and machine intelligence 27 (2005), no. 3, 328–340, doi: 10.1109/TPAMI.2005.55.
- [98] Javier Hernandez-Ortega, Julian Fierrez, Aythami Morales, and Javier Galbally, *Introduction to Presentation Attack Detection in Face Biometrics and Recent Advances*, 2021, doi: 10.48550/ARXIV.2111.11794.
- [99] University of Washington History of Computing, *The History of Artificial Intel-*

- ligence*, <https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf>, Dec 2006, Accessed on 29 June 2022.
- [100] M. Shamim Hossain, Ghulam Muhammad, Sk Md Mizanur Rahman, Wadood Abdul, Abdulhameed Alelaiwi, and Atif Alamri, *Toward end-to-end biometrics-based security for IoT infrastructure*, IEEE Wireless Communications 23 (2016), no. 5, 44–51, doi: 10.1109/MWC.2016.7721741.
- [101] Yi Sun, Xiaogang Wang, and Xiaoou Tang, *Deep learning face representation by joint identification-verification*, 2014, doi: 10.48550/ARXIV.1406.4773.
- [102] Alakananda Mitra, Saraju P. Mohanty, Peter Corcoran, and Elias Kougiianos, *Detection of Deep-Morphed Deepfake Images to Make Robust Automatic Facial Recognition Systems*, Proceedings of the 19th OITS International Conference on Information Technology (OCIT), 2021, doi: 10.1109/OCIT53463.2021.00039, pp. 149–154.
- [103] Yi Sun, Xiaogang Wang, and Xiaoou Tang, *Deeply learned face representations are sparse, selective, and robust*, 2014, doi: 10.48550/ARXIV.1412.1265.
- [104] Alakananda Mitra, Saraju P. Mohanty, Peter Corcoran, and Elias Kougiianos, *Easy-Deep: An IoT Friendly Robust Detection Method for GAN Generated Deepfake Images in SocialMedia*, Proceedings of the 4th IFIP International Internet of Things (IoT) Conference (IFIP-IoT), 2021, doi: 10.1007/978-3-030-96466-5_14, pp. 217–236.
- [105] Wu Ching Hsuan, Liang Sheng Hao, and Yeh Ching Kuo, *Recognition of rice damage area on UAV ortho-images*, Proceedings of the IEEE International Conference on Applied System Invention (ICASI), 2018, doi: 10.1109/ICASI.2018.8394470, pp. 1092–1094.
- [106] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, *Densely connected convolutional networks*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.
- [107] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller, *Labeled faces in the wild: A database for studying face recognition in unconstrained environments*, Tech. Report 07-49, University of Massachusetts, Amherst, October 2007.

- [108] David P. Hughes and Marcel Salathé, *An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing*, CoRR abs/1511.08060 (2015).
- [109] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa, *Globally and locally consistent image completion*, ACM Transactions on Graph. 36 (2017), no. 4.
- [110] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros, *Image-to-image translation with conditional adversarial networks*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, p. 1125–1134.
- [111] Ashish Jaiman, <https://www.orfonline.org/expert-speak/debating-the-ethics-of-deepfakes/>, <https://www.orfonline.org/expert-speak/debating-the-ethics-of-deepfakes/>, Accessed on 05 July 2022.
- [112] Miaomiao Ji, Keke Zhang, Qiufeng Wu, and Zhao Deng, *Multi-label learning for crop leaf diseases recognition and severity estimation based on convolutional neural networks*, Soft Computing 24 (2020), no. 20, 15327–15340.
- [113] Miaomiao Ji, Lei Zhang, and Qiufeng Wu, *Automatic grape leaf diseases identification via unitedmodel based on multiple convolutional neural networks*, Information Processing in Agriculture 7 (2020), no. 3, 418–426, doi: 10.1016/j.inpa.2019.10.003.
- [114] Shital Joshi, Saraju P. Mohanty, and Elias Kougiannos, *Everything You Wanted to Know About PUFs*, IEEE Potentials 36 (2017), no. 6, 38–46, doi: 10.1109/MPOT.2015.2490261.
- [115] TackHyun Jung, Sangwon Kim, and Keecheon Kim, *Deepvision: Deepfakes detection using human eye blinking pattern*, IEEE Access PP (2020), 1–1, doi: 10.1109/ACCESS.2020.2988660.
- [116] T. Karras, S. Laine, and T. Aila, *A style-based generator architecture for generative adversarial networks*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4396–4405.
- [117] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, *Analyzing and improving the image quality of stylegan*, Proceedings of the IEEE/CVF Conference on

- Computer Vision and Pattern Recognition (CVPR), 2020, doi: 10.1109/CVPR42600.2020.00813, pp. 8107–8116.
- [118] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen, *Progressive growing of gans for improved quality, stability, and variation*, arXiv: abs/1710.10196 (2017).
- [119] Guolin Ke, Q. Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Q. Ye, and T. Liu, *Lightgbm: A highly efficient gradient boosting decision tree*, Proceedings of Advances in Neural Information Processing Systems, 2017.
- [120] Mohamed Kerkech, Adel Hafiane, and Raphael Canals, *Vine disease detection in UAV multispectral images using optimized image registration and deep learning segmentation approach*, Computers and Electronics in Agriculture 174 (2020), 105446.
- [121] Jan Kietzmann, Linda W. Lee, Ian P. McCarthy, and Tim C. Kietzmann, *Deepfakes: Trick or treat?*, Business Horizons 63 (2020), no. 2, pp. 135–146.
- [122] Minha Kim, Shahroz Tariq, and Simon S Woo, *Fretal: Generalizing deepfake detection using knowledge distillation and representation learning*, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 1001–1012.
- [123] Taeksoo Kim, Moonsoo Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim, *Learning to discover cross-domain relations with generative adversarial networks*, arXiv: abs/1703.05192 (2017).
- [124] Davis E. King, *Dlib-ml: A Machine Learning Toolkit*, Journal of Machine Learning Research 10 (2009), 1755–1758.
- [125] Diederik P. Kingma and Jimmy Ba, *Adam: A Method for Stochastic Optimization*, 2014, doi: 10.48550/ARXIV.1412.6980.
- [126] Simon Kloos, Ye Yuan, Mariapina Castelli, and Annette Menzel, *Agricultural Drought Detection with MODIS Based Vegetation Health Indices in Southeast Germany*, Remote Sensing 13 (2021), no. 19.
- [127] Pavel Korshunov and Sebastien Marcel, *Deepfakes: a new threat to face recognition? assessment and detection*, arXiv:abs/1812.08685 (2018).
- [128] Pavel Korshunov and Sébastien Marcel, *DeepFakes: a New Threat to Face*

- Recognition? Assessment and Detection*, Idiap-RR Idiap-RR-18-2018, Idiap, 12 2018, http://publications.idiap.ch/attachments/reports/2018/Korshunov_Idiap-RR-18-2018.pdf.
- [129] Elias Kougianos, Saraju P Mohanty, Gavin Coelho, Umar Albalawi, and Prabha Sundaravadivel, *Design of a high-performance system for secure image communication in the Internet of Things*, IEEE Access 4 (2016), 1222–1242, doi: 10.1109/ACCESS.2016.2542800.
- [130] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, *Cifar-10 (canadian institute for advanced research)*.
- [131] Karel Kuželka and Peter Surový, *Automatic detection and quantification of wild game crop damage using an unmanned aerial vehicle (UAV) equipped with an optical sensor payload: a case study in wheat*, European Journal of Remote Sensing 51 (2018), no. 1, 241–250, doi: 10.1080/22797254.2017.1419442.
- [132] A. Kumar, A. Bhavsar, and R. Verma, *Detecting deepfakes with metric learning*, Proceedings of the 8th International Workshop on Biometrics and Forensics (IWBF), 2020, pp. 1–6.
- [133] Youngjoo Kwak, Badri Bhakta Shrestha, Atsuhiko Yorozyua, and Hisaya Sawano, *Rapid Damage Assessment of Rice Crop After Large-Scale Flood in the Cambodian Floodplain Using Temporal Spatial Data*, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 8 (2015), no. 7, 3700–3709, doi: 10.1109/JSTARS.2015.2440439.
- [134] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi, *Photo-realistic single image super-resolution using a generative adversarial network*, arXiv: abs/1609.04802 (2016).
- [135] Xiaodan Li, Yining Lang, Yuefeng Chen, Xiaofeng Mao, Yuan He, Shuhui Wang, Hui Xue, and Quan Lu, *Sharp multiple instance learning for deepfake video detection*, p. 1864–1872, Association for Computing Machinery, New York, NY, USA, 2020.

- [136] Y. Li, M. Chang, and S. Lyu, *In Ictu Oculi: Exposing AI created fake videos by detecting eye blinking*, Proceedings of the IEEE International Workshop on Information Forensics and Security, 2018, pp. 1–7.
- [137] Yijun Li, Sifei Liu, Jimei Yang, and Ming-Hsuan Yang, *Generative face completion*, arXiv: abs/1704.05838 (2017).
- [138] Yuezun Li and Siwei Lyu, *Exposing deepfake videos by detecting face warping artifacts*, CoRR abs/1811.00656 (2018).
- [139] Qiaokang Liang, Shao Xiang, Yucheng Hu, Gianmarc Coppola, Dan Zhang, and Wei Sun, *PD2SE-Net: Computer-assisted plant disease diagnosis and severity estimation network*, Computers and Electronics in Agriculture 157 (2019), 518–529, doi: 10.1016/j.compag.2019.01.034.
- [140] Ke Lin, Liang Gong, Yixiang Huang, Chengliang Liu, and Junsong Pan, *Deep Learning-Based Segmentation and Quantification of Cucumber Powdery Mildew Using Convolutional Neural Network*, Frontiers in Plant Science 10 (2019).
- [141] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, *Microsoft coco: Common objects in context*, Proceedings of the European conference on computer vision, Springer, 2014, pp. 740–755.
- [142] Chengjun Liu and Harry Wechsler, *Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition*, IEEE Transactions on Image processing 11 (2002), no. 4, 467–476, doi: 10.1109/TIP.2002.999679.
- [143] J. Liu and X. Wang, *Plant diseases and pests detection based on deep learning: a review.*, Plant Methods 17 (2021), no. 22, doi: 10.1186/s13007-021-00722-9.
- [144] Ming-Yu Liu, Thomas Breuel, and Jan Kautz, *Unsupervised image-to-image translation networks*, arXiv:abs/1703.00848 (2017).
- [145] Ming-Yu Liu and Oncel Tuzel, *Coupled generative adversarial networks*, arXiv: abs/1606.07536 (2016).
- [146] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-

- Yang Fu, and Alexander C Berg, *SSD: Single Shot Multibox Detector*, European conference on computer vision, Springer, 2016, doi: https://doi.org/10.1007/978-3-319-46448-0_2, pp. 21–37.
- [147] Z. Liu, X. Qi, and P. H. S. Torr, *Global texture enhancement for fake face detection in the wild*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 8057–8066.
- [148] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang, *Deep Learning Face Attributes in the Wild*, Proceedings of the International Conference on Computer Vision, December 2015, doi: [10.1109/ICCV.2015.425](https://doi.org/10.1109/ICCV.2015.425).
- [149] Mehedi Masud, Ghulam Muhammad, Hesham Alhumyani, Sultan S Alshamrani, Omar Cheikhrouhou, Saleh Ibrahim, and M Shamim Hossain, *Deep learning-based intelligent face recognition in IoT-cloud environment*, Computer Communications 152 (2020), 215–222, doi: [10.1016/j.comcom.2020.01.050](https://doi.org/10.1016/j.comcom.2020.01.050).
- [150] F. Matern, C. Riess, and M. Stamminger, *Exploiting visual artifacts to expose deepfakes and face manipulations*, Proceedings of the IEEE Winter Applications of Computer Vision Workshops (WACVW), 2019, pp. 83–92.
- [151] Scott McCloskey and Michael Albright, *Detecting gan-generated imagery using color cues*, arXiv:abs/1812.08247 (2018).
- [152] Mehdi Mirza and Simon Osindero, *Conditional Generative Adversarial Nets*, arXiv: 1411.1784 (2014).
- [153] A. Mitra, S. L. T. Vangipuram, A. K. Bapatla, V. K. V. V. Bathalapalli, S. P. Mohanty, E. Kougianos, and C. Ray, *Everything You wanted to Know about Smart Agriculture*, arXiv Computer Science arXiv:2201.04754 (Jan 2022), 45 pages, doi: [10.48550/arXiv.2201.04754](https://doi.org/10.48550/arXiv.2201.04754).
- [154] Alakananda Mitra, Dan Bigioi, Saraju P. Mohanty, Peter Corcoran, and Elias Kougianos, *iFace 1.1: A Proof-of-Concept of a Facial Authentication Based Digital ID for Smart Cities*, IEEE Access 10 (2022), 71791–71804, doi: [10.1109/ACCESS.2022.3187686](https://doi.org/10.1109/ACCESS.2022.3187686).

- [155] Alakananda Mitra, Saraju P. Mohanty, Peter Corcoran, and Elias Kougianos, *A Novel Machine Learning based Method for Deepfake Video Detection in Social Media*, Proceedings of the IEEE International Symposium on Smart Electronic Systems (iSES), 2020, doi: 10.1109/iSES50453.2020.00031, pp. 91–96.
- [156] Alakananda Mitra, Saraju P. Mohanty, Peter Corcoran, and Elias Kougianos, *A Machine Learning Based Approach for Deepfake Detection in Social Media Through Key Video Frame Extraction*, SN Computer Science 2 (2021), no. 2, 98, doi: 10.1007/s42979-021-00495-x.
- [157] Alakananda Mitra, Saraju P. Mohanty, Peter Corcoran, and Elias Kougianos, *iFace: A Deepfake Resilient Digital Identification Framework for Smart Cities*, Proceedings of the IEEE International Symposium on Smart Electronic Systems (iSES), 2021, doi: 10.1109/iSES52644.2021.00090, pp. 361–366.
- [158] Alakananda Mitra, Saraju P. Mohanty, and Elias Kougianos, *aGROdet: A Novel Framework for Plant Disease Detection and Leaf Damage Estimation*, Proceedings of the 5th IFIP International Internet of Things (IoT) Conference (IFIP-IoT), 2022, doi: https://doi.org/10.1007/978-3-031-18872-5_1, pp. 3–22.
- [159] Alakananda Mitra, Anshuman Singhal, Saraju P Mohanty, Elias Kougianos, and Chittaranjan Ray, *eCrop: A Novel Framework for Automatic Crop Damage Estimation in Smart Agriculture*, SN Computer Science 3 (2022), no. 4, 1–16, doi: 10.1007/s42979-022-01216-8.
- [160] Trisha Mittal, Uttaran Bhattacharya, Rohan Chandra, Aniket Bera, and Dinesh Manocha, *Emotions don't lie: An audio-visual deepfake detection method using affective cues*, p. 2823–2832, Association for Computing Machinery, New York, NY, USA, 2020.
- [161] Saraju P. Mohanty, *Security and Privacy by Design is Key in the Internet of Everything (IoE) Era*, IEEE Consumer Electronics Magazine 9 (2020), no. 2, 4–5, doi: 10.1109/MCE.2019.2954959.
- [162] Saraju P. Mohanty, Uma Choppali, and Elias Kougianos, *Everything you wanted to*

- know about smart cities: The internet of things is the backbone*, IEEE Consumer Electronics Magazine 5 (2016), no. 3, 60–70, doi: 10.1109/MCE.2016.2556879.
- [163] Sharada P. Mohanty, David P. Hughes, and Marcel Salathé, *Using Deep Learning for Image-Based Plant Disease Detection*, Frontiers in Plant Science 7 (2016), doi: 10.3389/fpls.2016.01419.
- [164] Anton Morzhakov, *Frontal Faces Neutral Expression 95 Landmarks*, <https://www.kaggle.com/antonmorzhakov/frontal-faces-neutral-expression-95-landmarks/code>, Accessed on 01 July 2021.
- [165] K. Nagasubramanian, S. Jones, and A. K. et al. Singh, *Plant disease identification using explainable 3D deep learning on hyperspectral images*, Plant Methods 15 (2019), 98, doi: 10.1186/s13007-019-0479-8.
- [166] RP Narmadha and G Arulvadivu, *Detection and measurement of paddy leaf disease symptoms using image processing*, Proceedings of the International Conference on Computer Communication and Informatics (ICCCI), 2017, pp. 1–4.
- [167] L. Nataraj, Tajuddin Manhar Mohammed, B. S. Manjunath, S. Chandrasekaran, A. Flenner, Jawadul H. Bappy, and A. Roy-Chowdhury, *Detecting GAN generated fake images using co-occurrence matrices*, arxiv:abs/1903.06836 (2019).
- [168] National Academies of Sciences, Engineering and Medicine and others, *Science breakthroughs to advance food and agricultural research by 2030*, National Academies Press, 2019.
- [169] H. H. Nguyen, J. Yamagishi, and I. Echizen, *Capsule-forensics: Using capsule networks to detect forged images and videos*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 2307–2311.
- [170] Yuval Nirkin, Yosi Keller, and Tal Hassner, *FSGAN: Subject agnostic face swapping and reenactment*, Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 7184–7193.
- [171] Adrián Ochandio Fernández, Cristian Ariel Olguín Pinatti, Rafael Masot Peris, and

- Nicolás Laguarda-Miró, *Freeze-Damage Detection in Lemons Using Electrochemical Impedance Spectroscopy*, *Sensors* 19 (2019), no. 18, doi: 10.3390/s19184051.
- [172] Seon Ho Oh, Geon-Woo Kim, and Kyung-Soo Lim, *Compact deep learned feature-based face recognition for visual internet of things*, *Journal of Super Computing* 74 (2018), 6729–6741, doi: 10.1007/s11227-017-2198-0.
- [173] Mehmet Metin Ozguven and Kemal Adem, *Automatic detection and classification of leaf spot disease in sugar beet using deep learning algorithms*, *Physica A: statistical mechanics and its applications* 535 (2019), 122537.
- [174] Vishal Pallagani, Vedant Khandelwal, Bharath Chandra, Venkanna Udutalapally, Debanjan Das, and Saraju P Mohanty, *DCrop: A deep-learning based framework for accurate prediction of diseases of crops in smart agriculture*, *Proceedings of the IEEE International Symposium on Smart Electronic Systems (iSES)*, 2019, pp. 29–33.
- [175] Maria Pantopoulou and Nicolas Sklavos, *An fpga-implemented parallel system of face recognition, for digital forensics applications*, *Proceedings of the IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, 2020, doi: 10.1109/ICCE-Berlin50680.2020.9352182, pp. 1–6.
- [176] Aditya Parikh, Mehul S Raval, Chandrasinh Parmar, and Sanjay Chaudhary, *Disease detection and severity estimation in cotton plant from unconstrained images*, *Proceedings of the IEEE international conference on data science and advanced analytics (DSAA)*, 2016, pp. 594–601.
- [177] Tyler Phillips, Xukai Zou, Feng Li, and Ninghui Li, *Enhancing Biometric-Capsule-Based Authentication and Facial Recognition via Deep Learning*, *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies (New York, NY, USA), SACMAT '19, Association for Computing Machinery*, 2019, doi: 10.1145/3322431.3325417, p. 141–146.
- [178] Eduard Puig Garcia, Felipe Gonzalez, Grant Hamilton, and Paul Grundy, *Assessment of crop insect damage using unmanned aerial systems: A machine learning approach*,

- Proceedings of the 21st International Congress on Modelling and Simulation, MOD-SIM2015, 2015, pp. 1420–1426.
- [179] Alec Radford, Luke Metz, and Soumith Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks*, arXiv: 1511.06434 (2016).
- [180] N. K. Ratha, J. H. Connell, and R. M. Bolle, *Enhancing security and privacy in biometrics-based authentication systems*, IBM Systems Journal 40 (2001), no. 3, 614–634, doi: 10.1147/sj.403.0614.
- [181] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, doi: 10.1109/CVPR.2016.91., pp. 779–788.
- [182] I. S. Reed and G. Solomon, *Polynomial codes over certain finite fields*, Journal of the Society for Industrial and Applied Mathematics 8 (1960), 300–304.
- [183] Shannon Reid, *The deepfake dilemma: Reconciling privacy and first amendment protections*, University of Pennsylvania Journal of Constitutional Law, Forthcoming, SSRN: <https://ssrn.com/abstract=3636464> (June 26,2020).
- [184] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, Advances in Neural Information Processing Systems (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.
- [185] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, *FaceForensics++: Learning to detect manipulated facial images*, Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1–11.
- [186] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner, *Faceforensics: A large-scale video dataset for forgery detection in human faces*, ArXiv abs/1803.09179 (2018).
- [187] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake, *”GrabCut”: Interactive*

- Foreground Extraction Using Iterated Graph Cuts*, ACM Trans. Graph. 23 (2004), no. 3, 309–314, doi: 10.1145/1015706.1015720.
- [188] Anneleen Rutten, Jim Casaer, Marjolein F. A. Vogels, Elisabeth A. Addink, Jeroen Vanden Borre, and Herwig Leirs, *Assessing agricultural damage by wild boar using drones*, Wildlife Society Bulletin 42 (2018), no. 4, 568–576, doi: 10.1002/wsb.916.
- [189] E. Sabir, J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan, *Recurrent convolutional strategies for face manipulation detection in videos*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 80–87.
- [190] Muhammad Hammad Saleem, Sapna Khanchi, Johan Potgieter, and Khalid Mahmood Arif, *Image-Based Plant Disease Identification by Deep Learning Meta-Architectures*, Plants 9 (2020), no. 11, doi: 10.3390/plants9111451.
- [191] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, *Improved techniques for training gans*, Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, p. 2234–2242.
- [192] C. Sanderson and B.C. Lovell, *Multi-Region Probabilistic Histograms for Robust and Scalable Identity Inference*, Lecture Notes in Computer Science (LNCS), Vol. 5558, 2009, doi: 10.1007/978-3-642-01793-3_21, pp. 199–208.
- [193] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, *Mobilenetv2: Inverted residuals and linear bottlenecks*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, doi: 10.1109/CVPR.2018.00474, pp. 4510–4520.
- [194] Sayantan Sarkar, Vishal M. Patel, and Rama Chellappa, *Deep feature-based face detection on mobile devices*, Proceedings of the IEEE International Conference on Identity, Security and Behavior Analysis (ISBA), 2016, doi: 10.1109/ISBA.2016.7477230, pp. 1–8.
- [195] Suryakant Sawant, Jayantrao Mohite, Mariappan Sakkan, and Srinivasu Pappula, *Near Real Time Crop Loss Estimation using Remote Sensing Observations*, Proceedings of

- the 8th International Conference on Agro-Geoinformatics (Agro-Geoinformatics), 2019, doi: 10.1109/Agro-Geoinformatics.2019.8820217, pp. 1–5.
- [196] Florian Schroff, Dmitry Kalenichenko, and James Philbin, *FaceNet: A unified embedding for face recognition and clustering*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, doi: 10.1109/CVPR.2015.7298682, pp. 815–823.
- [197] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra, *Grad-cam: Visual explanations from deep networks via gradient-based localization*, Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, doi: 10.1109/ICCV.2017.74, pp. 618–626.
- [198] Lastika Sethi, Anay Dave, Raj Bhagwani, and Ameyaa Biwalkar, *Video security against deepfakes and other forgeries*, Journal of Discrete Mathematical Sciences and Cryptography 23 (2020), 349–363, doi: 10.1080/09720529.2020.1721866.
- [199] Zakariyyaa Siddiq, *Data Quality and Quantity for Machine Learning | Monolith AI*, Accessed on 29 June 2022.
- [200] Karen Simonyan and Andrew Zisserman, *Very deep convolutional networks for large-scale image recognition*, Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, 2015.
- [201] Vijai Singh and A.K. Misra, *Detection of plant leaf diseases using image segmentation and soft computing techniques*, Information Processing in Agriculture 4 (2017), no. 1, 41–49, doi: 10.1016/j.inpa.2016.10.005.
- [202] Leandro Sosa, Ana Justel, and I. Molina, *Detection of Crop Hail Damage with a Machine Learning Algorithm Using Time Series of Remote Sensing Data*, Agronomy 11 (2021), no. 10.
- [203] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, J. Mach. Learn. Res. 15 (2014), no. 1, 1929–1958.

- [204] John A. Stevenson, *Plant Pathology, an Advanced Treatise*, vol. 3, J. G. Horsfall and A. E. Dimond, Eds. Academic Press, New York, 1960, 1960.
- [205] Ethan L Stewart, Tyr Wiesner-Hanks, Nicholas Kaczmar, Chad DeChant, Harvey Wu, Hod Lipson, Rebecca J Nelson, and Michael A Gore, *Quantitative phenotyping of Northern Leaf Blight in UAV images using deep learning*, *Remote Sensing* 11 (2019), no. 19, 2209.
- [206] Jun Sun, Yu Yang, Xiaofei He, and Xiaohong Wu, *Northern maize leaf blight detection under complex field environment based on deep learning*, *IEEE Access* 8 (2020), 33679–33688.
- [207] Yi Sun, Xiaogang Wang, and Xiaoou Tang, *Deep Learning Face Representation from Predicting 10,000 Classes*, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, doi: 10.1109/CVPR.2014.244, pp. 1891–1898.
- [208] Yagiz Sutcu, Qiming Li, and Nasir Memon, *Secure biometric templates from fingerprint-face features*, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–6.
- [209] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, *Synthesizing obama: learning lip sync from audio*, *ACM Transactions on Graphics (TOG)* 36 (2017), no. 4.
- [210] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna, *Rethinking the inception architecture for computer vision*, *CoRR* abs/1512.00567 (2015).
- [211] Yaniv Taigman, Adam Polyak, and Lior Wolf, *Unsupervised cross-domain image generation*, *arXiv: abs/1611.02200* (2016).
- [212] Yaniv Taigman, Ming Yang, Marc’Aurelio Ranzato, and Lior Wolf, *DeepFace: Closing the Gap to Human-Level Performance in Face Verification*, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, doi: 10.1109/CVPR.2014.220, pp. 1701–1708.
- [213] Mingxing Tan and Quoc V. Le, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*, 2019, doi: 10.48550/ARXIV.1905.11946.

- [214] Qian Tao and Raymond Veldhuis, *Biometric Authentication System on Mobile Personal Devices*, IEEE Transactions on Instrumentation and Measurement 59 (2010), no. 4, 763–773, doi: 10.1109/TIM.2009.2037873.
- [215] Qian Tao and Raymond N.J. Veldhuis, *Biometric Authentication for a Mobile Personal Device*, Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems - Workshops, 2006, doi: 10.1109/MOBIQW.2006.361741, pp. 1–3.
- [216] Shahroz Tariq, Sangyup Lee, Hoyoung Kim, Youjin Shin, and Simon S. Woo, *Detecting both machine and human created fake face images in the wild*, Proceedings of the 2nd International Workshop on Multimedia Privacy and Security, 2018, p. 81–87.
- [217] Karl Tate, *History of A.I.: Artificial Intelligence (Infographic) | Live Science*, <https://www.livescience.com/47544-history-of-a-i-artificial-intelligence-infographic.html>, Aug 25 2014, Accessed on 29 June 2022.
- [218] A. Tewari, M. Zollhöfer, H. Kim, P. Garrido, F. Bernard, P. Pérez, and C. Theobalt, *MoFA: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction*, Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 3735–3744.
- [219] K Thenmozhi and U Srinivasulu Reddy, *Crop pest classification based on deep convolutional neural network and transfer learning*, Computers and Electronics in Agriculture 164 (2019), 104906.
- [220] J. Thies, M. Zollhofer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt, *Realtime expression transfer for facial reenactment*, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2015, vol. 34, Art No.183, no. 6, 2015.
- [221] Suramya Tomar, *Converting video formats with FFmpeg*, Linux Journal 2006 (2006), no. 146, 10.
- [222] Luan Tran, Xi Yin, and Xiaoming Liu, *Disentangled Representation Learning GAN for Pose-Invariant Face Recognition*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, doi: 10.1109/CVPR.2017.141, pp. 1415–1424.

- [223] Pradyumna K Tripathy, Ajaya K Tripathy, Aditi Agarwal, and Saraju P Mohanty, *MyGreen: An IoT-Enabled Smart Greenhouse for Sustainable Agriculture*, IEEE Consumer Electronics Magazine 10 (2021), no. 4, 57–62.
- [224] Matthew A Turk, Alex Pentland, et al., *Face recognition using eigenfaces.*, Proceedings of the IEEE Conf. Comp. Vision and Pattern Recognition, vol. 91, 1991, pp. 586–591.
- [225] Venkanna Udutalapally, Saraju P Mohanty, Vishal Pallagani, and Vedant Khandelwal, *sCrop: A novel device for sustainable automatic disease prediction, crop selection, and irrigation in internet-of-agro-things for smart agriculture*, IEEE Sensors Journal (2020).
- [226] National Agricultural Statistics Service USDA, *Farms and Land in Farms 2019 Summary*, February, 2020, Accessed on 10 January, 2022.
- [227] Viktor Varkarakis, Shabab Bazrafkan, Gabriel Costache, and Peter Corcoran, *Validating seed data samples for synthetic identities – methodology and uniqueness metrics*, IEEE Access 8 (2020), 152532–152550.
- [228] Viktor Varkarakis, Wang Yao, and Peter Corcoran, *Towards End-to-End Neural Face Authentication in the Wild - Quantifying and Compensating for Directional Lighting Effects*, CoRR abs/2104.03854 (2021).
- [229] P. Viola and M. Jones, *Rapid object detection using a boosted cascade of simple features*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1, 2001, doi: 10.1109/CVPR.2001.990517, pp. I–I.
- [230] Paul Viola and Michael J Jones, *Robust real-time face detection*, International journal of computer vision 57 (2004), no. 2, 137–154, doi: <https://doi.org/10.1023/B:VISI.0000013087.49260.fb>.
- [231] Bo Wang, *Identification of Crop Diseases and Insect Pests Based on Deep Learning*, Scientific Programming 2022, Article ID 9179998 (2022), 10 pages, doi: 10.1155/2022/9179998.
- [232] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng

- Li, and Wei Liu, *CosFace: Large Margin Cosine Loss for Deep Face Recognition*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, doi: 10.1109/CVPR.2018.00552, pp. 5265–5274.
- [233] Mei Wang and Weihong Deng, *Deep face recognition: A survey*, Neurocomputing 429 (2021), 215–244, doi: <https://doi.org/10.1016/j.neucom.2020.10.081>.
- [234] 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*, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (Christian Bessiere, ed.), 7 2020, pp. 3444–3451.
- [235] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro, *High-resolution image synthesis and semantic manipulation with conditional gans*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8798–8807.
- [236] Mika Westerlund, *The emergence of deepfake technology: A review*, Technology Innovation Management Review 9 (2019), 39–52, doi: 10.22215/timreview/1282.
- [237] Tyr Wiesner-Hanks, Harvey Wu, Ethan Stewart, Chad DeChant, Nicholas Kaczmar, Hod Lipson, Michael A Gore, and Rebecca J Nelson, *Millimeter-level plant disease detection from aerial photographs via deep learning and crowdsourced data*, Frontiers in Plant Science 10 (2019), 1550.
- [238] Deressa Wodajo and Solomon Atnafu, *Deepfake video detection using convolutional vision transformer*, arXiv preprint arXiv:2102.11126 (2021).
- [239] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma, *Robust Face Recognition via Sparse Representation*, IEEE transactions on pattern analysis and machine intelligence 31 (2008), no. 2, 210–227, doi: 10.1109/TPAMI.2008.79.
- [240] Lifang Wu, Xingsheng Liu, Songlong Yuan, and Peng Xiao, *A novel key generation cryptosystem based on face features*, Proceedings of the IEEE 10th International Conference on Signal Processing, 2010, doi: 10.1109/ICOSP.2010.5656719, pp. 1675–1678.
- [241] D.J. Wuebbles, D.W. Fahey, K.A. Hibbard, D.J. Dokken, B.C. Stewart, , and

- T.K. Maycock (eds.), *USGCRP, 2017:Climate Science Special Report: Fourth National Climate Assessment*, U.S. Global Change Research Program I (2017), 470.
- [242] Thomaz W. F. Xavier, Roberto N. V. Souto, Thiago Statella, Rafael Galbieri, Emerson S. Santos, George S. Suli, and Peter Zeilhofer, *Identification of Ramularia Leaf Blight Cotton Disease Infection Levels by Multispectral, Multiscale UAV Imagery*, *Drones* 3 (2019), no. 2, doi: 10.3390/drones3020033.
- [243] Weiye Xu, Jianwei Liu, Shimin Zhang, Yuanqing Zheng, Feng Lin, Jinsong Han, Fu Xiao, and Kui Ren, *RFace: Anti-Spoofing Facial Authentication Using COTS RFID*, *Proceedings of the IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, doi: 10.1109/INFOCOM42981.2021.9488737, pp. 1–10.
- [244] Chen-Zhao Yang, Jun Ma, Shilin Wang, and Alan Wee-Chung Liew, *Preventing DeepFake Attacks on Speaker Authentication by Dynamic Lip Movement Analysis*, *IEEE Transactions on Information Forensics and Security* 16 (2021), 1841–1854, doi: 10.1109/TIFS.2020.3045937.
- [245] Wei Yang, Ce Yang, Ziyuan Hao, Chuanqi Xie, and Minzan Li, *Diagnosis of Plant Cold Damage Based on Hyperspectral Imaging and Convolutional Neural Network*, *IEEE Access* 7 (2019), 118239–118248, doi: 10.1109/ACCESS.2019.2936892.
- [246] X. Yang, Y. Li, and S. Lyu, *Exposing deep fakes using inconsistent head poses*, *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 8261–8265.
- [247] Xin Yang, Yuezun Li, and Siwei Lyu, *Exposing deep fakes using inconsistent head poses*, 2018.
- [248] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong, *Dualgan: Unsupervised dual learning for image-to-image translation*, arXiv: abs/1704.02510 (2017).
- [249] N. Yu, L. Davis, and M. Fritz, *Attributing fake images to gans: Learning and analyzing gan fingerprints*, *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7555–7565.
- [250] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao, *Joint face detection and*

- alignment using multitask cascaded convolutional networks*, IEEE signal processing letters 23 (2016), no. 10, 1499–1503, doi: 10.1109/LSP.2016.2603342.
- [251] Wende Zhang, Yao-Jen Chang, and Tsuhan Chen, *Optimal thresholding for key generation based on biometrics*, Proceedings of the International Conference on Image Processing, 2004. ICIP'04., vol. 5, IEEE, 2004, pp. 3451–3454.
- [252] Hanqing Zhao, Wenbo Zhou, Dongdong Chen, Tianyi Wei, Weiming Zhang, and Nenghai Yu, *Multi-attentional deepfake detection*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2021, pp. 2185–2194.
- [253] Yang Zhao and Changyou Chen, *Unpaired image-to-image translation via latent energy transport*, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 16418–16427.
- [254] Peng Zhou, Xintong Han, Vlad I. Morariu, and Larry S. Davis, *Two-stream neural networks for tampered face detection*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, doi: 10.1109/CVPRW.2017.229, pp. 1831–1839.
- [255] J. Zhu, T. Park, P. Isola, and A. A. Efros, *Unpaired image-to-image translation using cycle-consistent adversarial networks*, Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, doi: 10.1109/ICCV.2017.244, pp. 2242–2251.
- [256] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman, *Toward multimodal image-to-image translation*, Advances in Neural Information Processing Systems 30 (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), Curran Associates, Inc., 2017, pp. 465–476.